SMARTS: Scalable Multi-Agent Reinforcement Learning Training School for Autonomous Driving

Ming Zhou[*†1,2], **Jun Luo**[*‡1], **Julian Villella**[*1], **Yaodong Yang**[*1,3], **David Rusu**[1], **Jiayu Miao**[†1,2]
Weinan Zhang[2], **Montgomery Alban**[1], **Iman Fadakar**[1], **Zheng Chen**[1], **Aurora C. Huang**[1]
Ying Wen[†1,2], **Kimia Hassanzadeh**[1], **Daniel Graves**[1], **Dong Chen**[1], **Zhengbang Zhu**[†1,2]
Nhat Nguyen[1], **Mohamed Elsayed**[†1], **Kun Shao**[1], **Sanjeevan Ahilan**[†1], **Baokuan Zhang**[1], **Jiannan Wu**[1]
Zhengang Fu[1], **Kasra Rezaee**[1], **Peyman Yadmellat**[1], **Mohsen Rohani**[1], **Nicolas Perez Nieves** [1]
Yihan Ni[†1], **Seyedershad Banijamali**[1], **Alexander I. Cowen-Rivers**[1], **Zheng Tian**[†1,3], **Daniel Palenicek**[†1]
Haitham bou Ammar[1,3], **Hongbo Zhang**[1], **Wulong Liu**[1], **Jianye Hao**[1], **Jun Wang**[1,3]

[1]Noah's Ark Lab, Huawei Technologies, [2]Shanghai Jiao Tong University, [3]University College London

Abstract: Multi-agent interaction is a fundamental aspect of autonomous driving in the real world. Despite more than a decade of research and development, the problem of how to competently interact with diverse road users in diverse scenarios remains largely unsolved. Learning methods have much to offer towards solving this problem. But they require a realistic multi-agent simulator that generates diverse and competent driving interactions. To meet this need, we develop a dedicated simulation platform called SMARTS (**S**calable **M**ulti-**A**gent **RL** **T**raining **S**chool). SMARTS supports the training, accumulation, and use of diverse behavior models of road users. These are in turn used to create increasingly more realistic and diverse interactions that enable deeper and broader research on multi-agent interaction. In this paper, we describe the design goals of SMARTS, explain its basic architecture and its key features, and illustrate its use through concrete multi-agent experiments on interactive scenarios. We open-source the SMARTS platform and the associated benchmark tasks and evaluation metrics to encourage and empower research on multi-agent learning for autonomous driving. Our code is available at https://github.com/huawei-noah/SMARTS.

Keywords: autonomous driving, simulation, multi-agent, reinforcement learning

1 Introduction

Sixteen years have passed since the DAPRA Grand Challenge [1]. More than two millions of autonomous miles have been driven on public roads by Waymo's cars alone [2]. In spite of such remarkable achievement, fundamental challenges remain underexplored. One is the weather: to date, most deployment of autonomous driving (AD) has been in fair weather [3]. Another challenge is realistic interaction with diverse road users. Current mainstream level-4 AD solutions tend to limit interaction rather than embrace it: when encountering complexly interactive scenarios, the autonomous car tends to slow down and wait rather than acting proactively to find another way through. While this strategy is practical enough to put autonomous cars on many streets without getting into accidents where they are at fault, its conservativeness also creates headaches to other road users and compromises rider experience. In California in 2018, 57% of autonomous car crashes are rear endings and 29% are sideswipes—all by other cars and thus attributable to the conservativeness of the autonomous car [4]. Consider Waymo, for example, whose leadership position and longer history also gave us more publicly available data points. Issues with interaction surfaced from their trials in Arizona and California, including difficulties with unprotected left turns, merging into highway traffic, and road users who do not follow traffic regulations [5]. It was reported that Waymo vehicles

[*]Equal contribution. † Work done as an intern at Huawei. ‡ Corresponding author: jun.luo1@huawei.com.

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

Level	Description	Possible MARL Approach illustrated with Double Merge
M0	Rule-based planning and control without learning	N/A
M1	Single-agent learning without coordination with other learning agents	An agent could learn to implicitly anticipate how other agents will react to its own actions but the learned solution will likely suffer from non-stationarity and lack generalizability [11].
M2	Decentralized multi-agent learning with opponent modeling	MARL to model other agents, e.g. "how likely are they to yield to me if I start changing to their lane given how they have been driving in the past few seconds?" [12] [13]
M3	Coordinated multi-agent learning and independent execution	Coordinated learning of what to expect of each other even if there is no explicit coordination during execution, e.g. "some of them will give me the gap." [14]
M4	Local (Nash) equilibrium oriented multi-agent learning	Learn as a group towards a certain equilibrium for the double merge such that the cars from the two sides will weave through each other without much trouble. [15, 16]
M5	Social welfare oriented multi-agent learning	Learn broader repercussions of our actions, e.g. "if I force to the right lane now, I will cause a congestion on the highway, due to the fast heavy traffic coming down to this double merge from there."

Table 1: Levels of multi-agent learning in autonomous driving.

tend to form a trail of other cars behind them on small roads, causing other drivers to illegally pull around them [6]. In addition, Waymo vehicles often brake excessively compared to human standard, sometimes making riders and passengers carsick [7]. Finally, the pick-up and drop-off locations are often limited, adding significant amount of total travel time [8].

To face up to the interaction challenge in AD, we take the view that driving in shared public space with diverse road users is fundamentally a *multi-agent problem* that requires *multi-agent learning* as a key part of the solution. To illustrate this perspective, we propose—with a nod to [9]—the "multi-agent learning levels", or "M-levels" for short. **M0** agents are designed to follow specific rules and stick to them regardless of how the environment dynamics may have shifted. After driving many times through a dense intersection, for example, the agents have no improvements except through tweaks by human engineers. **M1** agents can learn to adapt from its online experiences and such a learning process is expected to change their behaviors for future runs. **M2** agents not only learn to adapt, but also learn to model other road users. However, there is still no direct information exchange among the learning agents during such a decentralized learning process. **M3** further requires information exchange among the agents during training so as to coordinate their learning, but at execution time there is neither centralized control nor direct information exchange. At **M4**, agents are required to coordinate their learning at the local group level (e.g. at an intersection) in a way such that the Nash equilibrium or other equilibrium variant is ensured at execution time. At **M5**, agents start to focus on solving how their local actions in the scenario (e.g. "how I change into the other lane here and now") may impact global welfare (e.g. "may decide whether there is going to be a congestion on the highway five minutes later") so as to minimise *the price of anarchy* [10].

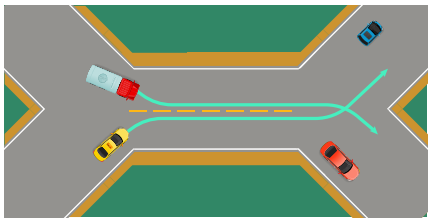


Figure 1: The double merge scenario.

Consider the so-called "double merge" (Figure 1) studied in Shalev-Shwartz et al. [12]. There are vehicles coming into a shared road section from one side and aiming to exit this section on the opposite side. In the process, they need to change lane by weaving through cars that may also be changing lane in the opposite direction. Failing that, they will be forced to either go out on the wrong road or wait somewhere before the road splits until a usable gap emerges.[1] Double merge is a sequential decision making setting that involves complex interaction among multiple agents. Questions one can ask about it include: Should a car drive further or wait here when looking for a gap? Should it force a lane change even when there is no adequate gap? Will the other cars give the gap? Is this other car coming over to my lane so that I can trade places with it? Furthermore, handwritten rules are unlikely to scale up to the full complexity of interactive scenarios such as the double merge [12]. We think learning with consideration of interaction is necessary and

[1] Such predicament does happen to autonomous cars. See a case in which an vehicle was forced to wait right before the split: https://youtu.be/HjtiiGCe1pE, and a related case: https://youtu.be/spw176TZ7-8?t=90.

multi-agent reinforcement learning (MARL) is especially promising. To show how MARL could be specifically relevant to autonomous driving, we have in Table 1 mapped the double merge scenario to MARL approaches at various M levels.

To date, AD research has mostly focused on M0 with highly limited attempts at M1 and M2 such as those noted in Table 1. We believe that a key reason is the lack of suitable AD simulation of interaction among heterogeneous traffic participants on the road.[2] When it comes to MARL for AD interaction, the behavior of the simulated traffic participants must be both realistic and diverse, especially for the *social vehicles* (i.e. vehicles sharing the driving environment with the autonomous vehicle) interacting with the *ego vehicles* (i.e. vehicles controlled by the AD software).

To be sure, there are some excellent open-source simulators (see Appendix A for a survey), such as CARLA [23] and SUMO [20], and even environments explicitly designed for RL and multi-agent research, such as Flow [18] and AIM4 [15]. Despite their respective successes, none of them comes close to addressing the specific need for in-depth MARL research on AD. The most specifically relevant efforts are probably highway-env [24], which is architected as a set of independently hand-crafted interaction scenarios, and BARK [25], which emphasizes the development of behavior models without explicit support for multi-agent research. Moreover, since there are only a small number of agent behavior models to choose from in the aforementioned solutions, it is still challenging to configure realistic interaction scenarios. Finally, the research community still lacks standard benchmarking suites that are comparable to MuJoCo [26] in how it brought physics simulation to RL research and to StarCraft II [27] in how it brought large scale games to multi-agent research, but remain true to the AD reality. Consequently, papers were published with ad-hoc, one-off small scale simulations that are typically concerned with just a single task and typically not maintained or reusable. In response to such a situation, we propose SMARTS (**S**calable **M**ulti-**A**gent **R**L **T**raining **S**chool). Our goal is to provide an industrial-strength platform that helps to bring MARL research for AD to the next level and both empowers and challenges it for many years to come.

2 Design Goals of the SMARTS Platform

Bootstrapping Realistic Interaction. Realistic and diverse interaction arise out of the confluence of some key contextual factors: (1) *physics*, (2) *behavior of road users*, (3) *road structure & regulations*, and (4) *background traffic flow*. The core of SMARTS covers all these aspects and is capable of continual growth to meet what realistic and diverse interaction requires. From the multi-agent perspective, the key to this continual growth is a bootstrapping strategy illustrated in Figure 2. At the heart is the *Social Agent Zoo*, which collects social agents that are used to control social vehicles in SMARTS simulations. These social agents could come from self play [28] or population-based training [29]. In order for the Social Agent Zoo to be grounded to the real world, however, at least some social agents will have to be also based on real-world driving data or domain knowledge. The process of growing the Social Agent Zoo is thus a multi-generational, iterative process that increases behavioral competence without sacrificing behavioral realism. As the Zoo grows, SMARTS will be able to simulate increasingly more complex but also increasingly more realistic multi-agent interaction. At the same time, given the open-source nature of SMARTS, we also expect that community contributions to the Social Agent Zoo will in turn benefit the global research community.

Heterogeneous Agent Computing. Under the bootstrapping strategy, social agents could be as sophisticated as a full AD stack, or as simple as a few lines of scripts. They could use the same observations and actions as the ego agent running the AD stack, or take shortcuts to use simulator states directly. They could use more neural network modules than what the on-board vehicle computer could run, or they could use no neural networks at all. For SMARTS to rely on such social agents for realistic and diverse simulated behavior, it naturally has to accommodate agents that are heterogeneous in design and implementation. Moreover, for scalability, we may not want to always simulate social vehicle behavior at the highest fidelity. Instead, we may use compute-intensive agents only when and where fine-grained interaction matters, such as for unprotected left turns or double merges. In scenarios where fidelity of interaction does not matter as much, such as highway cruising without lane change, much simpler classical control agents may be used.

[2] Interestingly, multi-agent learning research at M3 through M5 happens more often for *traffic management* [17, 18, 19], for which simulation has been under heavy development for about twenty years [20, 21, 22].

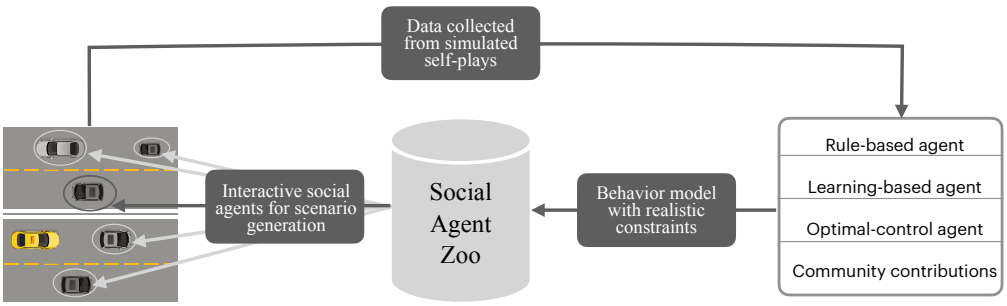


Figure 2: Bootstrapping interaction realism and diversity.

To retain the flexibility to deploy diverse social agents where they best fit the need, SMARTS needs to be able to dynamically associate social vehicles with social agents that diverge in terms of vehicle model, observation and action spaces, and computing needs. To support this, we have devised a mechanism called "bubbles", which is discussed in Section 3 below.

Key Features To fully support research usage and to cover the major factors that underpin realistic and diverse interaction, SMARTS must also have additional features.We have identified the features in Table 2 as important and have provided support for them in the initial release of SMARTS. Here, we can only highlight a few key features from this list.

Area	Features
Realistic Interaction	Realistic physics Heterogeneous ego and social agents Handwritten social agents Social agents trained with real-world data Social agents trained with RL Social agent zoo for crowd sourcing
Platform Integration	Multi-agent distribution Multi-simulation distribution RLlib integration for RL training SUMO integration for traffic simulation Built-in scenario composition
Research Friendliness	Gym APIs Headless mode Web-based visualization with recording Comprehensive observation & action options Multi-agent RL algorithm libraries Realistic benchmark suits

Table 2: SMARTS features.

- SMARTS is *natively multi-agent*, by which we mean that social agents could be as intelligent as ego agents and as heterogeneous as need be.
- For scalable integration, in addition to distributed training of ego agents supported through Ray [30], SMARTS also manages its own *distributed social agent computing*.
- To provide the best possible *researcher experience*, we follow the standard OpenAI Gym APIs, provide a web-streaming visualization solution, which allows ongoing simulation to be viewed from anywhere, such as on a smartphone, and offer a comprehensive set of observations and action spaces that can easily fit various research designs with minimum pre-processing and post-processing.
- Given our emphasis on support for MARL research, SMARTS integrates with popular libraries such as PyMARL [31] and MAlib [32], provides implementation of a few new algorithms, and supports a *comprehensive set of MARL algorithms*. (A most comprehensive set to our knowledge. See Table 7 for specifics.)
- SMARTS offers a *MARL benchmarking suite with AD-specific evaluation metrics*, which both poses challenging research questions and remains true to the AD reality. More benchmarking suites based on SMARTS are also in the workings.

3 Architecture and Implementation

To support our design goals, we have adopted a strongly *compositional* approach that reflects the diverse sources of the complexity of multi-agent interaction in real-world driving. We use compositionality to both scale out for the size of the simulation and scale up for interaction complexity. Here,

we can only comment on considerations about (1) how the simulation as a whole is composed, (2) how a specific interaction scenario is composed, and (3) how the various computational processes are orchestrated at runtime.

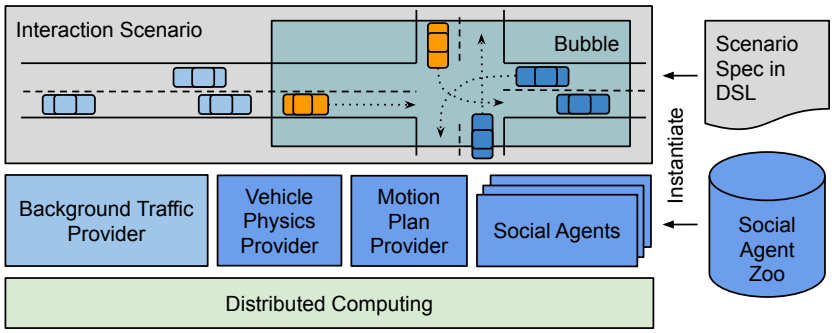


Figure 3: SMARTS Architecture. Interaction scenarios are defined with a domain-specific-language (DSL). Social agents are instantiated from the Social Agent Zoo. Orange vehicles are controlled by learning agents, dark blue vehicles by social agents, light blue ones by traffic provider. All providers and agents in principle run in their own processes, possibly remotely.

Simulation Providers. We adopt a *provider architecture*, which use limited co-simulators, or providers, for the major factors underpinning interaction. The overall simulation step orchestrates the stepping of these providers. *Background traffic flow* is provided everywhere in the map during the simulation. The map could be as big as a city and as small as an intersection. The background traffic provider is responsible for bringing the simulation session to the point where "foreground" interaction of interest could happen, at which point, the evolution of the traffic will be determined through the interaction of the social agents. SMARTS currently supports SUMO [20] as a background traffic provider.[3] *Physics* is supported through the *vehicle provider*, which simulates the specified physical vehicle model and exposes a control interface at different levels of abstraction: throttle-brake-steering, trajectory tracking, and lane following. With easily replaceable vehicle models and tire models, investigation on how physical changes, such as wet road surface or a flat tire, could affect multi-agent interaction may be readily studied. Our current implementation utilizes the *Bullet physics engine* [34] and supplements it with vehicle-specific dynamics and controllers. Finally, the *motion plan provider* can be used to implement social agents dedicated to highly specific maneuvers such as cut-in or U-turn. Our implementation includes three options: motion models available through SUMO, a simple Bezier-curve based motion planner [35], and an interactive motion planner based on OpEn [36].

Interaction Scenarios. To support large scale compositional construction of interaction scenarios, SMARTS comes with a Python-based domain specific language (DSL). At the base level, a scenario is composed of a map, a set of routes through the map, a set of vehicles with various driving characteristics, and "flows", which are assignments of the vehicles to the routes. As this level, we essentially just use the DSL to preprogram the background traffic. To integrate agents in the Social Agent Zoo into a scenario, we use "bubbles". A bubble is a spatiotemporal and conditionally specifiable region in which social vehicles are expected to be controlled by agents from the zoo. Different from *ObservedWorld* [25], which is used for planning or opponent modeling, bubble is for interaction management. As the background traffic (managed by the traffic provider) hits the outer membrane of the bubble, a transition happens wherein the control of the social vehicles could be handed over to specific social agents. To facilitate this handover, the bubble logic will need to instantiate the necessary vehicle and sensor models,[4] links these to the social agent's observation and action interfaces, starts the social agent processes, and switch control over from the traffic provider to the social agents. Imagine a large road network with many unprotected left turns and double merges, bubbles can be placed at these places wherein the multi-agent interaction can be configured and studied in detail.

[3] CARLA recently added trial integration with SUMO. This is a welcome move. Through our traffic provider interface, SMARTS may also be adapted to work with Aimsun Next [33] or Berkeley Flow [18].

[4] This is necessary because the background traffic provider typically models the social vehicles as mere rectangles with no articulated wheels or steering mechanism.

Accordingly, ego agents could be trained in a highly focused way by only collecting interaction trajectories in these bubbles, while the traffic provider continues to supply realistic traffic flow through the outer membrane of the bubble. This is how the bubble orchestrates the providers, the Social Agent Zoo, and the computing resource to create a milieu that is rich in multi-agent interaction.

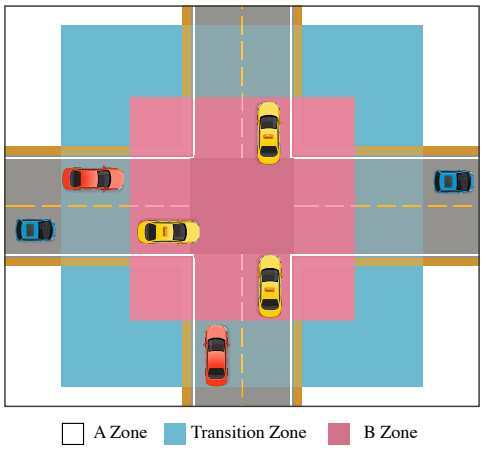


Figure 4: Bubble. Inside A Zone, social vehicles are controlled by the traffic provider; inside B Zone, by agents from the Social Agent Zoo.

Distributed Computing The bubble mechanism allows SMARTS to scale up without sacrificing interaction realism. Since the social agents may require a lot of compute, we cannot treat them as "non-character-players" (NPCs) run by simple scripts. Instead, we need to elastically assign computing resource according to the need of simulation fidelity. Another issue here is managing the dependencies of the social agents, especially their deep learning dependencies. Through the bootstrapping process, these agents are likely going to be based on more and more sophisticated deep models. SMARTS generalizes the approach Ray [30] takes to run scalable distributed training of ego agents by making social agents run in their own processes, possibly on remote machines, and provide options for running them with or without Ray and with or without TensorFlow and PyTorch dependencies. This way, we can run large scale, possibly city-scale, multi-agent simulations wherein one computer is dedicated to running the background traffic of the entire city, while many other computers are dedicated to running social agents inside the various bubbles.[5]

4 Support for Multi-Agent Reinforcement Learning

Using the scenario DSL of SMARTS, we can create numerous scenarios that vary in road structure and traffic. Figure 9 in Appendix E highlights some of them. These scenarios provide rich traffic flows and road conditions to help us study behavior and driving strategies. To train and evaluate the ego agents with specific parameters in these scenarios, we have implemented a benchmark runner based on Ray [30] and RLlib [37]. With the support of Ray, SMARTS enables users to conduct scalable and parallelizable training and evaluations. Below, we briefly describe other key elements of SMARTS for supporting MARL research.

Observation, Action and Reward. In SMARTS, the observation space for an agent is specified as a configurable subset of available sensor types that include dynamic object list, bird's-eye view occupancy grid maps and RGB images, ego vehicle states, and road structure etc. To make the use of deep models easier, we also provide built-in adapters to convert sensor data to tensors. An agent's action space is determined by the controller chosen for it. SMARTS supports four types of controllers: *Continuous*, *Actuator Dynamic*, *Trajectory Tracking* and *Lane Following*. The mapping from controllers to action spaces is listed in the Appendix C. For reward, SMARTS provides distance travelled along the mission route per time step as the raw reward signal, which may be used in reward customization that combines feedback on key events (e.g. collision, off-road, etc.) and other observations.

Algorithms & Baselines. SMARTS integrates three popular (MA)RL libraries—RLlib [37], PyMARL [31], and MAlib [32], allowing many algorithms to be used directly for AD research. While our experiments reported below used RLlib for high-concurrency training, RLlib's support for MARL is highly limited, because it focuses mainly on the implementation of single agent reinforcement learning. We therefore have provided implementations of more MARL algorithms under additional paradigms, such as centralized training & decentralized execution (CTDE) and networked agent learning [38]. Appendix F overviews the algorithms currently available in SMARTS.

[5]To get an intuitive sense of what SMARTS simulations are like when all these are up and running, please see sample screenshots in Appendix B.

To our knowledge, it is by far the most comprehensive set of MARL algorithms, covering all MARL paradigms (as show in Figure 10).

Metrics. Existing evaluation metrics in autonomous driving tend to be limited to specific tasks [25, 23]. Specifically, they focus on macro driving performance, such as *1) Whether the agent reached its goal; 2) Collision ratio; 3) Distance from the agent to its goal; 4) Ratio of driving the wrong way*. They hardly explain why an agent achieves a certain performance level. Under the multi-agent setting, an agent's performance is not only a matter of how it itself behaves, but also depends on how other agents interact with it. Thus, for multi-agent evaluation in SMARTS, we propose a more comprehensive set of metrics in Table 6 of Appendix D, covering three main aspects that include model performance, agent population behavior distribution, and game theoretic analysis.

5 Experiments & Results

To demonstrate how SMARTS may support MARL research on AD, we run a series of experiments that involve using seven MARL algorithms to tackle three increasingly more challenging driving scenarios that require non-trivial interactive capabilities to be learned.

- Two-Way traffic: As Figure 9e shows, agents start from either of two opposite lanes, then drive straight to the other end without colliding into each other or into social vehicles.
- Double Merge: As illustrated in Figure 1 and discussed in the Introduction.
- Unprotected Intersection: As Figure 9c shows, agents will start from any of the four roads linked to a junction, pass through the junction, and then continue on to other roads.

Each agent in each scenario has a different mission to complete: driving from a specific start to a specific goal. Traffic consists of the agents in training as well as the background traffic provided by SUMO [20]. The chosen baselines include two independent learning algorithms, DQN [39] and PPO [40], four centralized training methods, MAAC, MF-AC [41], MADDPG[42], and Networked Fitted-Q [43], and a fully centralized method, CommNet [44]. The observation, action, and reward functions are kept the same for all these baselines.

Observation & Action. The observation is a stack of three consecutive frames, which covers the dynamic objects and key events. For each frame, it contains: *1) relative position of goal; 2) distance to the center of lane; 3) speed; 4) steering; 5) a list of heading errors; 6) at most eight neighboring vehicles' driving states (relative distance, speed and position); 7) a bird's-eye view gray-scale image with the agent at the center.* The action used here is a four-dimensional vector of discrete values, for longitudinal control—*keep lane* and *slow down*—and lateral control—*turn right* and *turn left*.

Reward. The reward is a weighted sum of the reward components shaped according to ego vehicle states, interactions involving surrounding vehicles, and key traffic events. More details can be found in our implementation code.

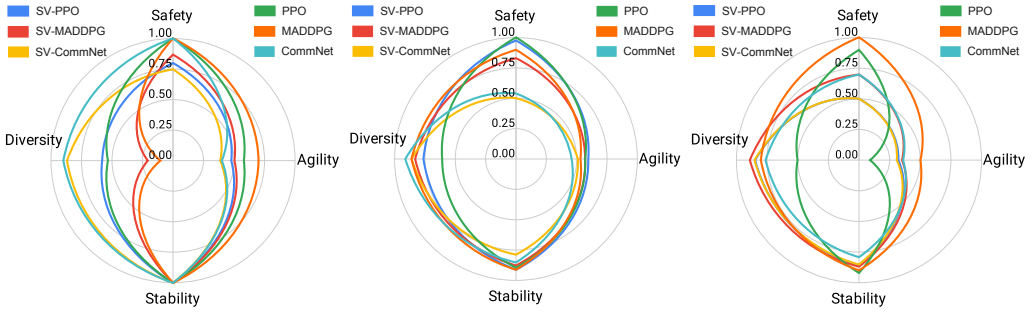
Performance. Table 3 shows the results of the baselines. It reports average *collision rate* and *completion rate* of the agent population on 10 episodes under two different background traffic settings. Unsurprisingly, all algorithms score higher under the No Social Vehicle setting. MADDPG outperforms the others in most tasks, especially in Intersection. Since the reward function and the observation are shared among all algorithms, independent learning agents could be at a disadvantage. For example, the better performance of MADDPG in the toughest setting of unprotected intersection, we believe, could be explained by the fact that agents learn to interact better by leveraging extra information from other agents. Since the overall interaction dynamics of the environment depends on all the agents, and the driving behavior of one agent will be influenced by other agents, extra information from other agents will mitigate the uncertainty for decision making, and lead the policy learning process into a more reasonable direction. (See Table 8 for additional results.)

Behavior Analysis. Figure 5 gives radar plots of four *behavior metrics*: Safety, Agility, Stability, and (Control) Diversity. These plots allow us to analyze behavioral difference of the algorithms in different scenarios. In the Two-Way scenario, behavioral difference concentrates on Diversity. In

Scenario	No Social Vehicle			Random Social Vehicle		
	PPO	CommNet	MADDPG	PPO	CommNet	MADDPG
Two-Way	**0/1**	0/0.96	**0/1**	0.25/0.75	0.25/0.65	**0.13/0.87**
Double Merge	**0/1**	0.7/0.25	0.1/0.9	**0.02/0.98**	0.5/0.5	0.17/0.8
Intersection	0.1/0.07	0.3/0.7	**0/1**	0.50/0.45	0.5/0.45	**0.30/0.7**

Table 3: Average Collision Rate / Completion Rate of three baselines on goal-directed navigation tasks in three scenarios. For collision rate, lower is better; for completion rate, higher is better. Collision and completion rates do not always sum up to 1, because agents may fail to reach destination before an episode ends. Examination of replay confirmed this for CommNet in Two-Way.

Double-Merge and Intersection, behavioral difference covers both Diversity and Safety. In Intersection, Agility also sees a fair amount of behavior difference. The overall trend here is that as the difficulty of the scenarios increases, interaction among traffic participants become more frequent and more complex, which forces the agents to perform more complex behavior and makes it harder for algorithms to score higher on some metrics compared to the easier scenarios. Change in Agility between Two-Way and Intersection is a case in point: as the scenario gets harder, max achieved agility drops and the spread of agility becomes wider, across the algorithms.



(a) Two-Way (b) Double Merge (c) Intersection

Figure 5: Results on behavior metrics. The larger the coverage, the more desirable the behavior. The wider scattered the curves, the more diverse the behaviors. "SV-" represents the algorithms interacting with social vehicles. See Figure 11 for additional plots.

6 Conclusion

We introduced SMARTS, an open-source platform that is dedicated to bringing together scalable multi-agent learning and scalable simulation of realistic driving interaction. Through our experiments and behavior analysis, we hope to show how the SMARTS platform as a whole can support investigations into how various MARL algorithms perform in the AD context. We hope this in turn can guide the development of new algorithms. Since its first internal release, SMARTS has successfully supported three autonomous driving competitions internationally,[6] wherein thousands of submissions of agent models have been automatically evaluated for the leaderboards. We believe the value of SMARTS will grow in the coming years, as it exposes the realistic driving interaction challenge to more researchers and enables them to further their research through trying to solve this challenge in a principle way.

Acknowledgments

The authors would like to thank the anonymous reviewers for their helpful comments and thank the many Huawei colleagues, SJTU students, competition participants in U.K. and China, and other early users of SMARTS for their valuable feedback. Weinan Zhang's contribution to this work is partly supported by "New Generation of AI 2030" Major Project (2018AAA0100900), National Natural Science Foundation of China (61632017) and Huawei Innovation Research Program.

[6] SMARTS-supported competitions can be found at https://drive-ml.com/.

References

[1] M. Buehler, K. Iagnemma, and S. Singh. *The 2005 DARPA grand challenge: the great robot race*, volume 36. Springer, 2007.

[2] Waymo. Waymo safety report. https://storage.googleapis.com/sdc-prod/v1/safety-report/2020-09-22-safety-report.pdf, 09 2020. (Accessed on 09/22/2020).

[3] C. Badue, R. Guidolini, R. V. Carneiro, P. Azevedo, V. B. Cardoso, A. Forechi, L. Jesus, R. Berriel, T. M. Paixão, F. Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[4] J. Stewart. Humans just can't stop rear-ending self-driving cars—let's figure out why. https://www.wired.com/story/self-driving-car-crashes-rear-endings-why-charts-statistics/. (Accessed on 09/23/2020).

[5] A. Efrati. Waymo's big ambitions slowed by tech trouble. https://www.theinformation.com/articles/waymos-big-ambitions-slowed-by-tech-trouble. (Accessed on 09/22/2020).

[6] F. Siddiqui. Some of the biggest critics of Waymo and other self-driving cars are the silicon valley residents who know how they work. https://wapo.st/30UAX6R. (Accessed on 09/22/2020).

[7] A. Efrati. Waymo riders describe experiences on the road, . URL https://www.theinformation.com/articles/waymo-riders-describe-experiences-on-the-road. (Accessed on 09/22/2020).

[8] A. Efrati. Waymo's backseat drivers: Confidential data reveals self-driving taxi hurdles, . URL https://www.theinformation.com/articles/waymos-backseat-drivers-confidential-data-reveals-self-driving-taxi-hurdles. (Accessed on 09/22/2020).

[9] S. International. Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. 2016.

[10] T. Roughgarden. *Selfish routing and the price of anarchy*, volume 174. MIT press Cambridge, 2005.

[11] A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah. Learning to drive in a day. In *ICRA*, pages 8248–8254. IEEE, 2019.

[12] S. Shalev-Shwartz, S. Shammah, and A. Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. 10 2016.

[13] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus. Social behavior for autonomous vehicles. *PNAS*, 116(50):24972–24978, 2019. ISSN 0027-8424. doi:10.1073/pnas.1820676116. URL https://www.pnas.org/content/116/50/24972.

[14] R. E. Wang, M. Everett, and J. P. How. R-MADDPG for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684*, 2020.

[15] AIM: Autonomous intersection management. http://www.cs.utexas.edu/~aim/. (Accessed on 09/22/2020).

[16] H. Zhang, W. Chen, Z. Huang, M. Li, Y. Yang, W. Zhang, and J. Wang. Bi-level actor-critic for multi-agent coordination. *arXiv preprint arXiv:1909.03510*, 2019.

[17] H. Zhang, S. Feng, C. Liu, Y. Ding, Y. Zhu, Z. Zhou, W. Zhang, Y. Yu, H. Jin, and Z. Li. CityFlow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *WWW*, page 3620–3624. ACM, 2019.

[18] C. Wu, A. Kreidieh, K. Parvate, E. Vinitsky, and A. M. Bayen. Flow: Architecture and benchmarking for reinforcement learning in traffic control. *ArXiv*, abs/1710.05465, 2017.

[19] H. Wei, N. Xu, H. Zhang, G. Zheng, X. Zang, C. Chen, W. Zhang, Y. Zhu, K. Xu, and Z. Li. Colight: Learning network-level cooperation for traffic signal control. In *CIKM*, pages 1913–1922, 2019.

[20] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner. SUMO (Simulation of Urban MObility)—an open-source traffic simulation. In *MESM*, pages 183–187, 2002.

[21] K. Dresner and P. Stone. Multiagent traffic management: a reservation-based intersection control mechanism. In *AAMAS 2004.*, pages 530–537, 2004.

[22] J. Barcelí, E. Codina, J. Casas, J. L. Ferrer, and D. García. Microscopic traffic simulation: A tool for the design, analysis and evaluation of intelligent transport systems. *J. Intell. Robotics Syst.*, 41(2–3):173–203, Jan. 2005. ISSN 0921-0296. doi:10.1007/s10846-005-3808-2. URL https://doi.org/10.1007/s10846-005-3808-2.

[23] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. Carla: An open urban driving simulator. *arXiv preprint arXiv:1711.03938*, 2017.

[24] E. Leurent. An environment for autonomous driving decision-making. https://github.com/eleurent/highway-env, 2018.

[25] J. Bernhard, K. Esterle, P. Hart, and T. Kessler. BARK: Open behavior benchmarking in multi-agent environments. *arXiv preprint arXiv:2003.02604*, 2020.

[26] E. Todorov, T. Erez, and Y. Tassa. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[27] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, et al. StarCraft II: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[28] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[29] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[30] P. Moritz, R. Nishihara, S. Wang, A. Tumanov, R. Liaw, E. Liang, M. Elibol, Z. Yang, W. Paul, M. I. Jordan, et al. Ray: A distributed framework for emerging AI applications. In *OSDI*, pages 561–577, 2018.

[31] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. Rudner, C.-M. Hung, P. H. Torr, J. Foerster, and S. Whiteson. The StarCraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[32] Y. Wen. Malib: A multi-agent learning framework. https://github.com/ying-wen/malib. (Accessed on 09/23/2020).

[33] J. Casas, J. L. Ferrer, D. Garcia, J. Perarnau, and A. Torday. Traffic simulation with Aimsun. In *Fundamentals of traffic simulation*, pages 173–232. Springer, 2010.

[34] Bullet real-time physics simulation. https://pybullet.org/wordpress/. (Accessed on 09/23/2020).

[35] C. Scheiderer, T. Thun, and T. Meisen. Bézier curve based continuous and smooth motion planning for self-learning industrial robots. *Procedia Manufacturing*, 38:423–430, 2019.

[36] P. Sopasakis, E. Fresk, and P. Patrinos. OpEn: Code generation for embedded nonconvex optimization. In *IFAC World Congress*, Berlin, 2020.

[37] E. Liang, R. Liaw, R. Nishihara, P. Moritz, R. Fox, K. Goldberg, J. Gonzalez, M. Jordan, and I. Stoica. Rllib: Abstractions for distributed reinforcement learning. In *ICML*, pages 3053–3062, 2018.

[38] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Basar. Fully decentralized multi-agent reinforcement learning with networked agents. In *ICML*, pages 5872–5881, 2018.

[39] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

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

[41] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang. Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5571–5580, 2018.

[42] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[43] K. Zhang, Z. Yang, and T. Basar. Networked multi-agent reinforcement learning in continuous spaces. In *2018 IEEE CDC*, pages 2771–2776. IEEE, 2018.

[44] S. Sukhbaatar, R. Fergus, et al. Learning multiagent communication with backpropagation. In *NIPS*, pages 2244–2252, 2016.

[45] Carmaker — IPG Automotive. https://ipg-automotive.com/products-services/simulation-software/carmaker/. (Accessed on 09/22/2020).

[46] Mechanical simulation. https://www.carsim.com/. (Accessed on 09/22/2020).

[47] Pro-SiVIC 2017.0 — myESI. https://myesi.esi-group.com/downloads/software-downloads/pro-sivic-2017.0. (Accessed on 09/22/2020).

[48] PreScan — TASS International. https://tass.plm.automation.siemens.com/prescan. (Accessed on 09/22/2020).

[49] Automotive-simulator – 4D-virtualiz. https://www.4d-virtualiz.com/en/automotive-simulator/. (Accessed on 09/22/2020).

[50] Robot simulator coppeliasim: create, compose, simulate, any robot - coppelia robotics. https://www.coppeliarobotics.com/. (Accessed on 09/22/2020).

[51] Gazebo. http://gazebosim.org/. (Accessed on 09/22/2020).

[52] Webots robot simulator. https://github.com/cyberbotics/webots. (Accessed on 09/22/2020).

[53] Rockstar Games. Grand Theft Auto V. https://www.rockstargames.com/games/V. (Accessed on 09/22/2020).

[54] D. Loiacono, L. Cardamone, and P. L. Lanzi. Simulated car racing championship: Competition software manual, 2013.

[55] Aimsun. Aimsun Next: Your personal mobility modeling lab. https://www.aimsun.com/aimsun-next/. (Accessed on 09/22/2020).

[56] MATSim.org. https://www.matsim.org/. (Accessed on 09/22/2020).

[57] MITSIMLab — INTELLIGENT TRANSPORTATION SYSTEMS LAB. https://its.mit.edu/software/mitsimlab. (Accessed on 09/22/2020).

[58] PTV Group. Traffic simulation software. https://www.ptvgroup.com/en/solutions/products/ptv-vissim/. (Accessed on 09/22/2020).

[59] S. Shah, D. Dey, C. Lovett, and A. Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *FSR*, 2017. URL https://arxiv.org/abs/1705.05065.

[60] Apollo. https://apollo.auto/platform/simulation.html. (Accessed on 09/22/2020).

[61] Driving simulation for autonomous driving, ADAS, vehicle dynamics and motorsport. http://www.rfpro.com/. (Accessed on 09/22/2020).

[62] Virtual Test Drive (VTD) – Complete Tool-Chain for Driving Simulation. https://www.mscsoftware.com/product/virtual-test-drive. (Accessed on 09/22/2020).

[63] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[64] M. Chevalier-Boisvert, F. Golemo, Y. Cao, B. Mehta, and L. Paull. Duckietown environments for openai gym. https://github.com/duckietown/gym-duckietown, 2018.

[65] Y. Naoto, K. Alexander, Z. Billy, and A. Kumar. Gym_torcs. https://github.com/ugo-nama-kun/gym_torcs, 2020.

[66] P. Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning, 2019.

[67] S. Anirban, R. Sohan, A. Buridi, and K. Meha. MADRaS: Multi-agent driving simulator. https://github.com/madras-simulator/MADRaS. (Accessed on 09/22/2020).

[68] T. Atlantic. Inside Waymo's secret world for training self-driving cars. https://www.theatlantic.com/technology/archive/2017/08/inside-waymos-secret-testing-and-simulation-facilities/537648/. (Accessed on 09/22/2020).

[69] K. Vogt. The disengagement myth. https://medium.com/cruise/the-disengagement-myth-1b5cbdf8e239. (Accessed on 09/22/2020).

[70] D. Anguelov. Presentation at MIT course on self-driving cars. https://www.youtube.com/watch?v=Q0nGo2-y0xY&feature=youtu.be&t=1920&ab_channel=LexFridman. (Accessed on 09/22/2020).

[71] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[72] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[73] R. S. Sutton, D. A. McAllester, S. P. Singh, Y. Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pages 1057–1063, 1999.

[74] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[75] Y. Wen, Y. Yang, R. Luo, J. Wang, and W. Pan. Probabilistic recursive reasoning for multi-agent reinforcement learning. In *International Conference on Learning Representations*, 2018.

[76] Z. Tian, Y. Wen, Z. Gong, F. Punakkath, S. Zou, and J. Wang. A regularized opponent model with maximum entropy objective. *arXiv preprint arXiv:1905.08087*, 2019.

[77] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson. Counterfactual multi-agent policy gradients. In S. A. McIlraith and K. Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*. AAAI Press, 2018.

[78] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. F. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, pages 2085–2087, 2018.

[79] T. Rashid, M. Samvelyan, C. Schroeder, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304, 2018.

[80] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896, 2019.

[81] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pages 7613–7624, 2019.

[82] T. Osogami and R. Raymond. Determinantal reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4659–4666, 2019.

[83] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar. Finite-sample analysis for decentralized batch multi-agent reinforcement learning with networked agents. *arXiv preprint arXiv:1812.02783*, 2018.

A Survey of AD Related Simulators

Table 4 categorizes simulators relevant to autonomous driving research and explains how they may be related to SMARTS. As is clear from the table, simulators in the AD and MA/RL categories are more similar to SMARTS in terms of either their focus on autonomous driving or their emphasis on interactive behavior. Moreover, among the AD and MA/RL simulators in Table 4, we can discern two approaches to simulation of interactive behavior: *data-replay simulators* and *interactive simulators*.

Category	Simulator	Purpose	Comment
Automotive	CarMaker[45]	Testing detailed designs regarding dynamics, safety, and performance of the individual physical vehicle.	Not concerned with interaction with other road users. Possible use as vehicle provider for SMARTS.
	CarSim [46]		
	Pro-SiVIC [47]		
	PreScan [48]		
Robotics	4DV-Sim [49]	General robotics simulation for full sensor-control loop with realistic models.	With suitable robot models, could provide SMARTS with varieties of road users beyond just cars and trucks.
	CoppeliaSim [50]		
	Gazebo [51]		
	Webots [52]		
Games	GTA V [53]	An action-adventure game involving driving.	Its vivid graphics inspired its use for generating AD training data.
	TORCS [54]	An open-source racing simulator.	Useful for testing RL algorithms but not AD due to racing focus.
Traffic	Aimsun Next [55]	Traffic simulation at the *microscopic* level, with motion and interaction of individual vehicles simulated.	Suitable for simulation of traffic flow but not interaction behavior. SMARTS currently uses SUMO as background traffic provider.
	MATSIM [56]		
	MITSIMLab [57]		
	SUMO [20]		
	VISSIM [58]		
AD	AirSim [59]	Dedicated AD simulators. Limited interaction through scripting or data-replay.	Focus is on testing ego vehicle AI, not on solving interaction. Could use SMARTS to supply high-quality interaction.
	Apollo [60]		
	CARLA [23]		
	rFpro [61]		
	VTD [62]		
MA/RL	AIM4 [15]	Multi-agent framework for managing autonomous vehicles at intersections.	Focused on intersection only and assuming all vehicles are autonomous.
	BARK [25]	Multi-agent envs with extensible social agent models.	Similar to SMARTS in emphasizing importance of social agents; no explicit support for multi-agent research.
	CarRacing [63]	An RL env in OpenAI Gym.	Car racing for testing RL.
	CityFlow [17]	Streamlined simulator for traffic optimization with RL.	Could provide background traffic in low-fidelity setting.
	Duckietown [64]	AD simulator for education.	For the Duckietown project.
	Flow [18]	Microscopic traffic simulation for deep RL.	Wraps SUMO and Aimsun Next.
	Gym-TORCS [65]	Gym-style env for TORCS	Shows need for standard envs.
	highway-env [24]	Hand-coded interaction envs.	Shows need for interaction envs.
	MACAD-Gym [66]	Multi-agent connected car simulation using CARLA.	Links V2V communication with multi-agent coordination.
	MADRaS [67]	TORCS multi-agent wrapper.	Limited to racing.

Table 4: Overview of Existing Autonomous Driving Related Simulators.

Data-replay simulators are extensively used in the industry for real-world autonomous driving R&D. Most simulated miles run by autonomous driving companies are based on replay of recorded data, against which the behavior of a new version of the ego vehicle AI may be assessed [68]. Because the recorded data capture how other road users actually reacted to the ego vehicle as well as to each other in real world trials, there is no question about the realism of their behavior. However, because the behavior of the new version of the ego vehicle AI is expected to diverge under at least some identical situations from the old version, in order to adequately assess the new version, the simulated road users must be able to reasonably react to the new behavior in a way that is not already reflected in the recorded data. This in turn requires intelligent agents that can control the

simulated road users, especially the social vehicles [69, 70]. However, technical specifics about how to add interactive intelligence to data-replay simulators are virtually never published.

Interactive simulators, instead of relying on recorded data, try to build the necessary interactive behaviors from ground up by providing intelligent behavior models for the simulated road users. CARLA [23], highway-env [24], BARK [25], and even SUMO [20] are such examples. In the fully multi-agent case (e.g. AIM4 [15]), there may not even be any real difference between social vehicle AI and ego vehicle AI. To date, however, the behavior of road users in these simulators is typically controlled by simple rules, scripts, or highly limited models. This results in overly simplistic and rigid behavior that falls far short of the complexity and diversity of real interactions on the road.

SMARTS is expected to be a platform that bootstraps realistic and diverse behavior models (Figure 2) that can be used to *make data-replay simulators more interactive* and *make interactive simulators more realistic*.

B SMARTS Simulations

To give an intuitive sense of what SMARTS is like when up and running, we have included several screenshots from a few sample simulation runs.

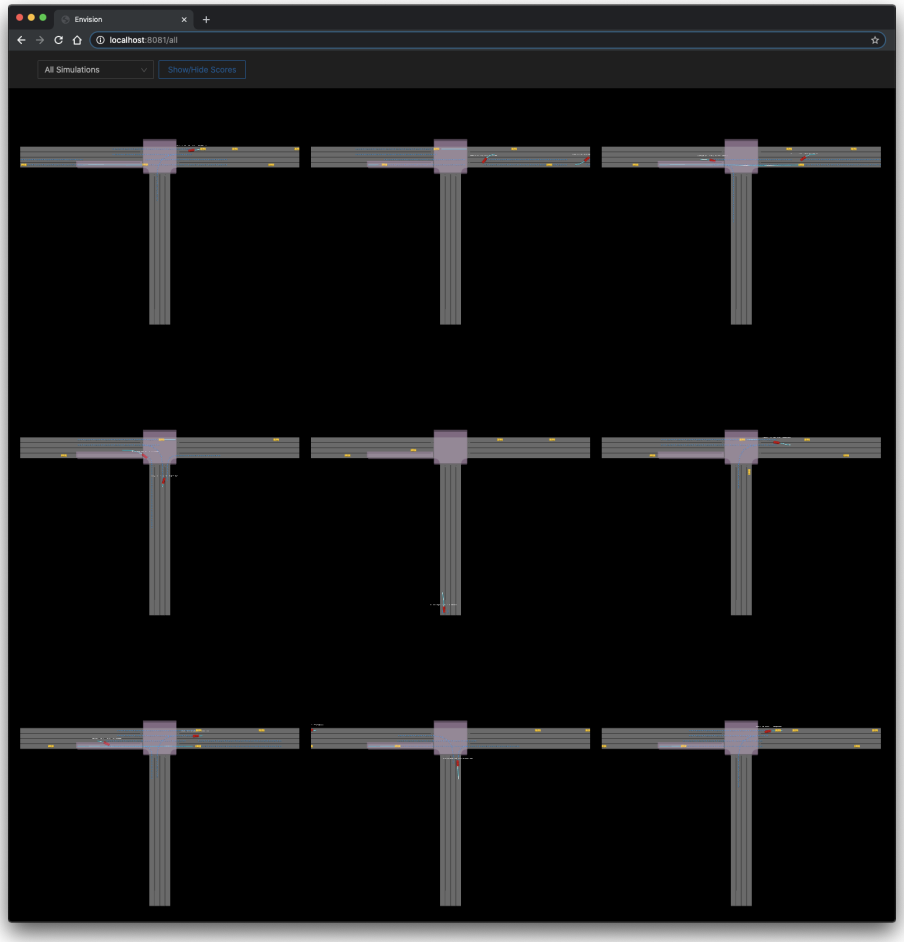


Figure 6: Multi-Instance and Web-Base Visualization. The SMARTS platform supports multiple instances running simultaneously. Instances are distributed across multiple processes or across networked machines. To collect diverse interaction data quickly, each instance can be configured to run its own set of multiple scenarios by sampling through them. In this example, 9 concurrent instances are simultaneously visualized in the browser.

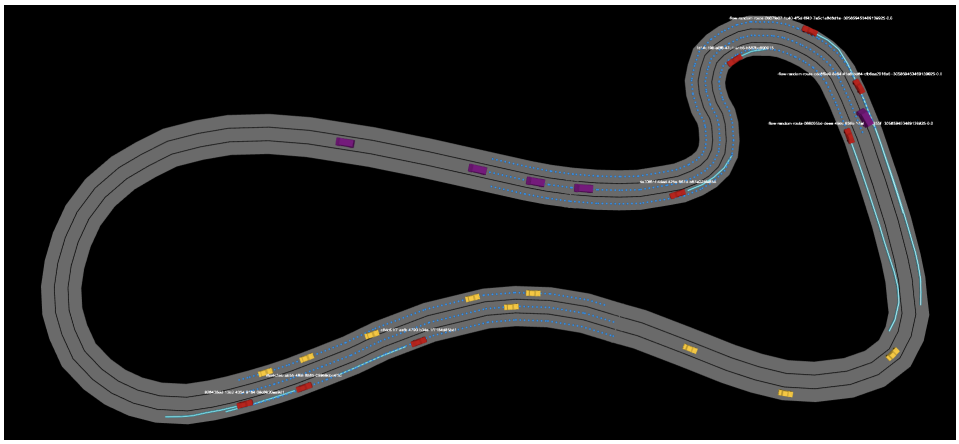
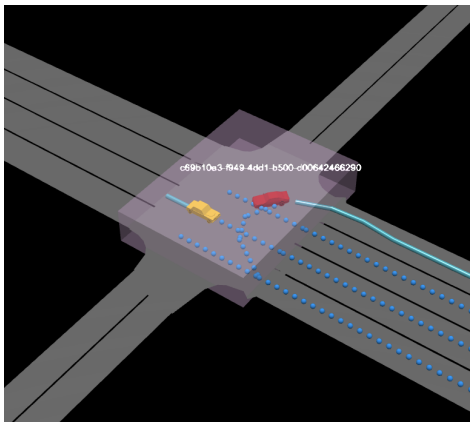
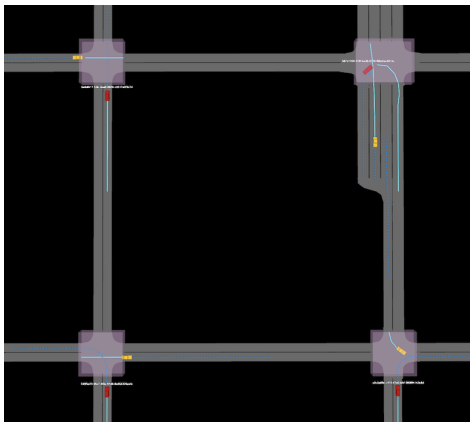


Figure 7: Multi-Agent. Multiple ego agents in training (red vehicles) can run simultaneously within a shared SMARTS instance. Each agent runs in a separate process and can also run remotely.



(a) Bubble



(b) Multiple Bubbles

Figure 8: Bubbles. Bubbles are regions in which social vehicles (yellow) may be controlled by agents from the Social Agent Zoo and interact meaningfully with ego vehicles (red).

C Controllers and Action Spaces

In SMARTS, agent's action space depends on the controller. Table 5 shows the mapping between controllers and action spaces.

Controller	Action Space Type	Control Command Dimensions
LaneFollowingController	Mixed	target speed
		lane change (+1 or 0 or -1)
TrajectoryTrackingController	-	trajectory
ActuatorDynamicController	Continuous	throttle
		brake
		steering rate in rad
ContinuousController	Continuous	throttle
		brake
		steering

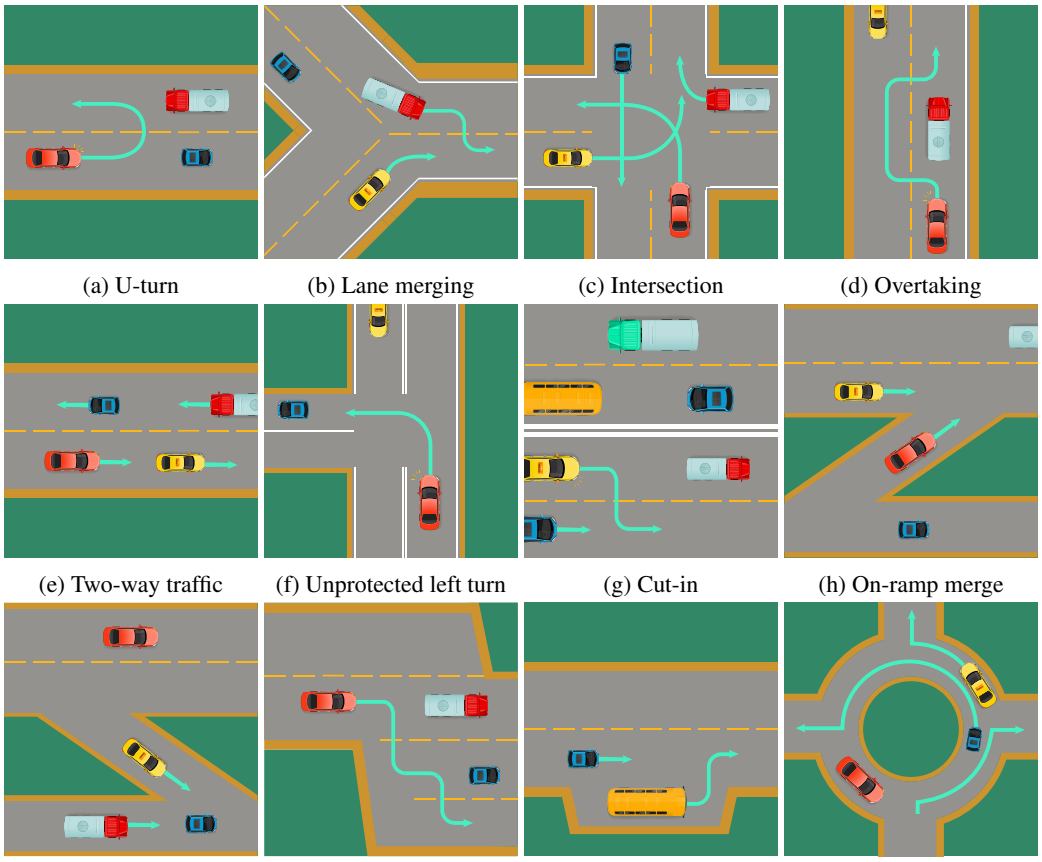
Table 5: Mapping from controllers to action spaces.

D Metrics

Metric	Type	Description
Collision Rate	Performance	# of collisions over # of episodes.
Completion Rate	Performance	# of missions completed over # of episodes.
Generalization	Performance	Robustness of algorithms to scenario variation.
Safety	Behavior	Integrated metrics, e.g. non-collision rate.
Agility	Behavior	Integrated metrics, e.g. speed.
Stability	Behavior	Integrated metrics for driving smoothness.
Control Diversity	Behavior	Preference for longitudinal or lateral control.
Cut-in Ratio	Behavior	Probability of cut-in in traffic flow.
Stochasticity	Behavior	Stochasticity of decision making.
Collaborative	Game theory	Compatible interests, e.g. ratio of giving way.
Competitive	Game theory	Conflicting interests, e.g. ratio of overtaking.

Table 6: Evaluation metrics for multi-agent autonomous driving. User can find the implementation of them in the `Metrics` class located in benchmarking library of SMARTS. Also, it is very easy for users to extend these metrics by inheriting from the `Metrics` class.

E Scenarios



(a) U-turn (b) Lane merging (c) Intersection (d) Overtaking

(e) Two-way traffic (f) Unprotected left turn (g) Cut-in (h) On-ramp merge

(i) Off-ramp merge (j) Cascading lane change (k) Bus stop merge (l) Roundabout

Figure 9: Scenarios of driving interaction specifiable in SMARTS

F Algorithm Library

We have integrated a large set of popular MARL algorithms. Table 7 gives an overview of these algorithms and tags them according to 1) paradigm, 2) communication or not, and 3) framework dependency. The four paradigms are fully centralized training, fully decentralized, centralized training & decentralized execution (CTDE) and networked agent learning. They are illustrated in Figure 10.

Paradigm	Algorithm	Communication	Framework
Fully centralized training	BiCNet [71] ⋆	Yes	malib
	CommNet[44] ⋆	Yes	RLlib/malib
Fully decentralized	Indepedent Q [72]	No	RLlib
	Independent PG [73]	No	RLlib
	Independent AC [74]	No	RLlib/malib
	PR2 [75]	No	malib
	ROMMEO [76]	No	malib
	Supervised Opponent Modeling	No	malib
CTDE	Centralized V	No	RLlib
	MAAC ⋆	No	RLlib
	MADDPG [42]	No	RLlib/malib
	MF-AC/Q [41] ⋆	No	RLlib/malib
	COMA [77]	No	PyMARL
	VDN [78]	No	PyMARL
	QMIX [79]	No	PyMARL
	QTRAN [80]	No	PyMARL
	MAVEN [81]	No	PyMARL
	Q-DPP [82] ⋆	No	PyMARL
Networked agent learning	Networked Fitted-Q [83] ⋆	graph	RLlib

Table 7: Algorithms available in SMARTS. Those tagged with ⋆ are our own implementations.

G Experiment Results

Table 8 and Figure 11 provide additional details about our experiment results.

Algorithm	Scenario - No Social Vehicle			Scenario - Random Social Vehicle		
	Two-Way	Double Merge	Intersection	Two-Way	Double Merge	Intersection
DQN	0/0.97	0.77/0.23	0.83/0.20	0.40/0.60	0.60/0.23	0.92/0.05
PPO	**0/1**	**0/1**	0.1/0.07	0.25/0.75	**0.02/0.98**	0.50/0.45
MAAC	**0/1**	0.42/0.58	**0/1**	0.25/0.75	0.42/0.6	0.32/0.68
MFAC	0/0.8	0.6/0.4	0.54/0.4	0.45/0.5	0.7/0.3	0.62/0.37
Net-Q	0/0.3	0.7/0.25	0.4/0.23	0.4/0.2	0.8/0.2	0.75/0.2
CommNet	0/0.96	0.46/0.45	0.3/0.7	0.25/0.65	0.5/0.5	0.5/0.45
MADDPG	**0/1**	0.1/0.9	**0/1**	**0.13/0.87**	0.17/0.8	**0.30/0.7**

Table 8: Average collision rate / completion rate of selected baselines. Net-Q: Networked Fitted-Q.

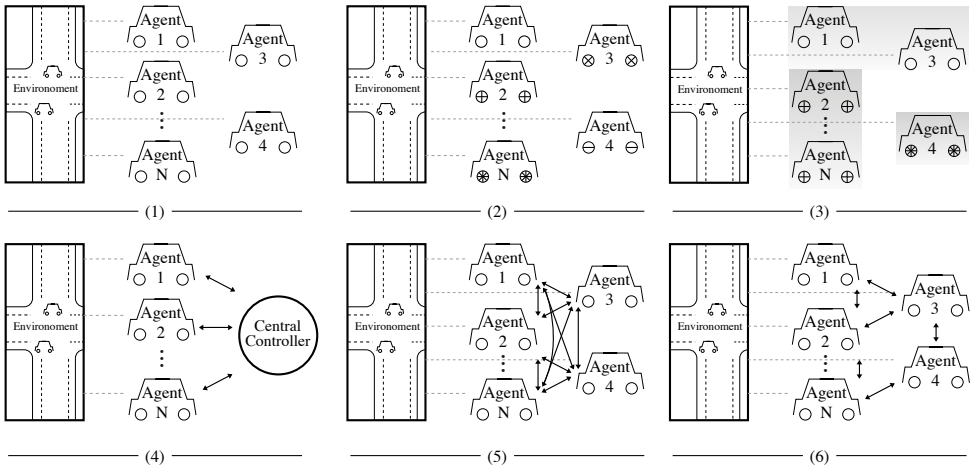


Figure 10: MARL Paradigms. The *fully decentralized paradigm* has three sub-settings: (1) parameter sharing, (2) non-parameter-sharing, and (3) group sharing, respectively. (4) is the *fully centralized training paradigm*, i.e., a central controller controls all agents to make decisions. (5) is the *CTDE paradigm*, wherein each agent shares its information with other agents globally, but makes decisions independently. (6) represents the *networked agent learning paradigm*, which differs from CTDE in that each agent shares information only with its networked neighbors.

H User Examples

H.1 Scenario Generation

In SMARTS, a scenario is composed of map, traffic flow, agent missions, and optionally but importantly bubble specification. Given the map, users can easily create scenarios with specific requirements by writing a scenario configuration file. An example of scenario configuration and generation is given below.

```
# Configuration: in 'scenario.py'
# 1. agent mission initialization and generation
agent_missions = [
    Mission(Route(begin=("top_left", 0, 10), end=("down_right", 0, 30))),
    ...
]

gen_missions(scenario=scenario, missions=agent_missions)

# 2. define social vehicle with diverse behavior
impatient_car = TrafficActor(
    name="car",
    speed=Distribution(sigma=0.2, mean=1.0),
    lane_changing_model=LaneChangingModel(impatience=1, cooperative=0.25),
    junction_model=JunctionModel(
        drive_after_red_time=1.5,
        drive_after_yellow_time=1.0,
        impatience=1.0
    ),
)

patient_car = TrafficActor(
    name="car",
    speed=Distribution(sigma=0.2, mean=0.8),
    lane_changing_model=LaneChangingModel(impatience=0, cooperative=0.5),
```

```
    junction_model=JunctionModel(drive_after_yellow_time=1.0, impatience=0.5)
)

# 3. traffic flow customization
traffic = Traffic(
    flows=[
        Flow(
            route=Route(begin=("down_left", 0, 30), end=("merge", 0, 150),),
            rate=1.,
            actors={impatient_car: 0.5, patient_car: 0.5},
        ),
        ...
    ]
)

gen_traffic(scenario, traffic, name="example")

# one command to do scenario generation
make build-scenario scenario=${SCENARIO_DIR}
```

H.2 Building Agents

SMARTS provide 10 built-in agent types that cover frequently used combinations of observation-sensor and action-controller combinations. These are available through the agent interface. The combination of agent interface, agent policy, and a set of adapters form the agent spec. Concrete agent instances are built according to the agent spec.

```
# AgentSpec for agent customization
class AgentSpec:
    ...
    policy_builder: Callable[..., AgentPolicy] = None
    policy_params: Optional[Any] = None
    observation_adapter: Callable = lambda obs: obs
    action_adapter: Callable = lambda act: act
    reward_adapter: Callable = lambda obs, reward: reward
    info_adapter: Callable = lambda obs, reward, info: info
    ...

# use case: build agents
agent_spec = AgentSpec(
    interface=AgentInterface.from_type(AgentType.Laner),
    policy_params={"policy_function": lambda _: "keep_lane"},
    policy_builder=AgentPolicy.from_function,
)

env = gym.make(
    "smarts.env:hiway-v0",
    scenarios=["scenarios/loop"],
    agent_specs={agent_id: agent_spec},
)

agent = agent_spec.build_agent()
```

H.3 Running Single-agent & Multi-agent Training

In SMARTS, it is easy to build a single-agent or multi-agent training experiment. The key steps include agent initialization, algorithm specification, and training scenario specification. Multiple training scenarios can be automatically loaded and used for multi-task learning.

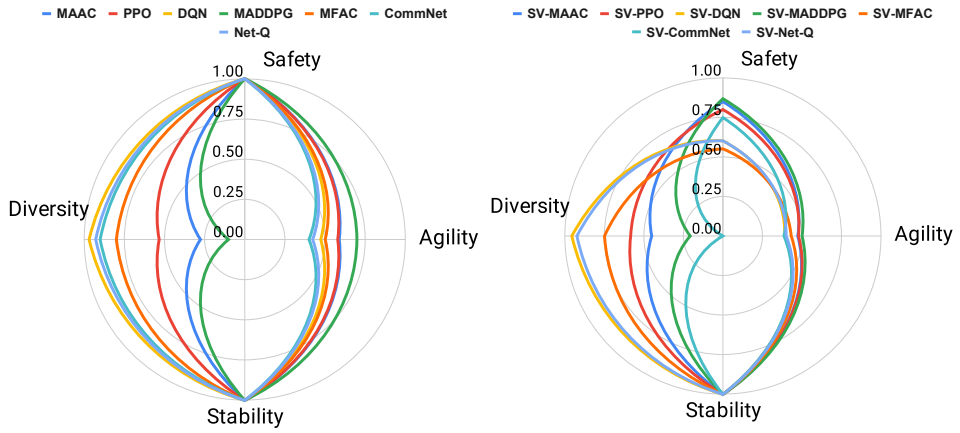
```
# initializing multiple agents
```

```
agents = {
    agent_id: AgentSpec(
        **config.agent, interface=AgentInterface(**config.interface)
    )
    for agent_id in agent_ids
}

# initializing environment with one or more scenarios
env_config = {
    "seed": 42,
    "scenarios": [str(scenario_path), ...],  # specifying scenarios
    "headless": args.headless,
    "agent_specs": agents,
}

policies = {k: config.policy for k in agents}

tune_config = {
    ...
    "env": RLlibHiWayEnv,
    "env_config": env_config,
    "multiagent": {
        "policies": policies,
        "policy_mapping_fn": lambda agent_id: agent_id,
    },
    ...
}

# run experiment
analysis = tune.run(
    ...
    config=tune_config,
)
```

H.4 Running Evaluation

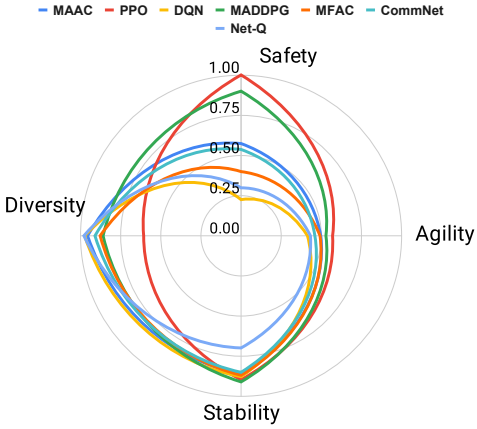
We implemented a benchmarking runner to do algorithm evaluation. The benchmarking runner's `Episode` recorder and `Metric` class allow us to record the steps and perform evaluation accordingly.

```
@dataclass
class Episode(EpisodeLog):
    def record_step(**kwargs):
        ...

class Metric:
    def log_step(self, observations, actions, rewards, dones,
                 infos, episode):
        ...

    def compute(self):
        """Evaluate algorithm based on records with the given metrics."""
        ...

metrics_obj = Metric(num_episodes=10)
while not done and step < num_steps:
    # ...
    metrics_obj.log_step(observations, actions, rewards,
                 dones, infos, episode=episode)
    # ...
_ = metrics_obj.compute()
```

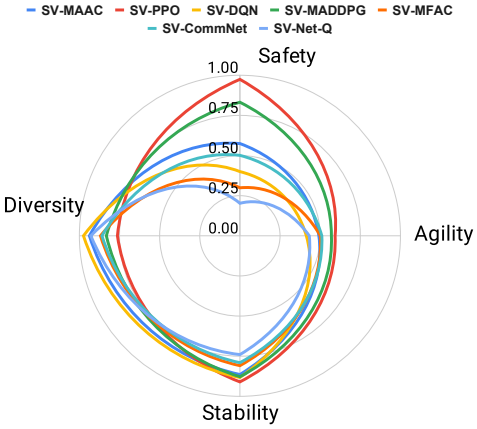


(a) Two-Way: No SV

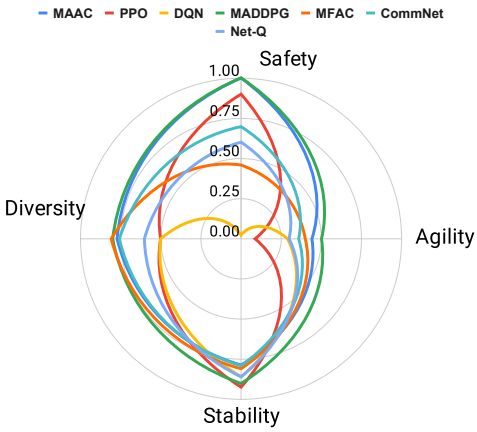
(b) Two-Way: SV



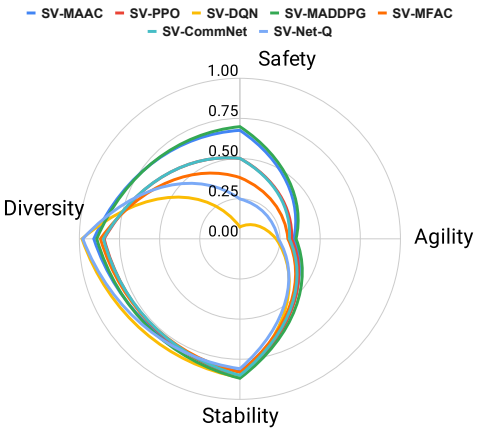
(c) Double Merge: No SV



(d) Double Merge: SV



(e) Intersection: No SV



(f) Intersection: SV

Figure 11: Results on behavior metrics. The larger the coverage, the more desirable the behavior. The wider scattered the curves, the more diverse the behaviors. "SV" represents the algorithms interacting with social vehicles. "No SV" means no social vehicles.