The Sidewalk Problem

David Isele and Kikuo Fujimura
Honda Research Institute-USA
{disele,kfujimura}@honda-ri.com

ABSTRACT

We propose a benchmark problem for cooperative stochastic games and interactive decision making. *The sidewalk problem* consists of two agents, each navigating toward their own goal, where the paths of the separate agents intersect. This requires one or both agents to deviate from the optimal path in order to avoid collision. Because both agents are dynamic, there is a need for prediction and/or replanning. We formulate the problem as a stochastic game, and provide details for implementing the problem in both discrete and continuous state and action spaces. We evaluate popular algorithms on variants of the problem, and we discuss generalizations and suggest other benchmark problems for cooperative stochastic games and interactive decision making.

KEYWORDS

Cooperative Stochastic Games; Interactive Decision Making; Adaptive Learning; Safe Reinforcement Learning

1 INTRODUCTION

Consider walking down the sidewalk when you encounter another pedestrian heading in the opposite direction. Both you and the other pedestrian adjust paths to prevent collision. Since both parties make decisions in real time, it is possible that both adjusted paths still conflict. This points out a difficulty *humans* have in inferring the intentions of other agents. Robots and artificial agents with slower response times, more limited dynamics, and less sophisticated reasoning capabilities are likely to find themselves even more disadvantaged when faced with interactive decision making.

The sidewalk problem, illustrated in Figure 1, isolates several key difficulties to adaptive learning. An agent must infer the intention of other agents. Since the agent is acting under uncertainty, and there is likely a necessity to re-plan, and the optimal plan in terms of efficiency may not be the optimal plan in terms of flexibility. It is likely that safer and better-in-expectation algorithms optimize for the latter, choosing plans that are easily adaptable to new optima in the event of a change.

This situation appears in and relates to various problems concerning robots in unstructured environments. While it directly relates to issues in crowd navigation [23, 24, 28] and is a contributing factor to the frozen robot problem, the more general aspect of interactive decision making also has important implications to autonomous driving, where negotiating merges and lane changes are heavily influenced by the highly variable behavior of other agents [1, 26].

This work does not propose a solution to the sidewalk problem, although we do show several existing algorithms perform reasonably well. We are instead interested in how the problem elegantly identifies a difficulty in interactive decision making, and propose it as a benchmark which any successful algorithm must be able to handle. We focus on formalizing and understanding the problem and its implementation details so that it might be readily used as a benchmark for people investigating issues related to cooperative stochastic games and decision making. It is our hope that by providing a readily available benchmark we can create a concrete problem to facilitate thinking about key issues in the field and allow a test bed for researchers to quickly verify their theories.

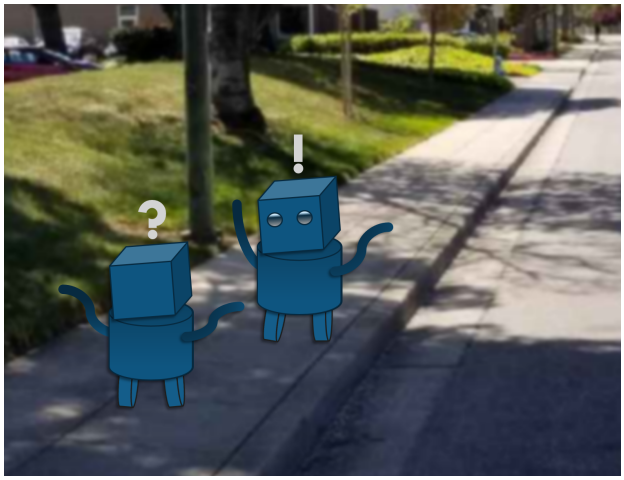


Figure 1: The Sidewalk Problem. Each agent tries to efficiently navigate to their respective goals, while avoiding collisions.

We note that the sidewalk problem is similar in spirit to the grid world problem presented by Hu and Wellman [13]. It also bares a likeness to the adversarial soccer problem [5, 19]. None-the-less we still see many papers on stochastic games that are purely theoretical, and contain no experiments [4, 6, 31], and we believe consistent benchmarks will help encourage empirical demonstrations of theoretical work, making research more comparable.

In this paper we formalize the sidewalk problem as a stochastic game. We pose two variants of the problem - one continuous and one discrete. We discuss design parameters of the problem *e.g.* setting the reward and noise. Additionally, we note that the sidewalk problem is suitable for investigations into safe RL, which we elaborate on. We then evaluate several existing algorithms on the sidewalk problem and discuss generalizations to the sidewalk problem, briefly sketching out other possible benchmark problems for stochastic games.

2 FORMULATION

Stochastic games extend the Markov Decision Process (MDP) formalism common to reinforcement learning (RL) to enable the handling of multiple agents, each of which influences the rewards and

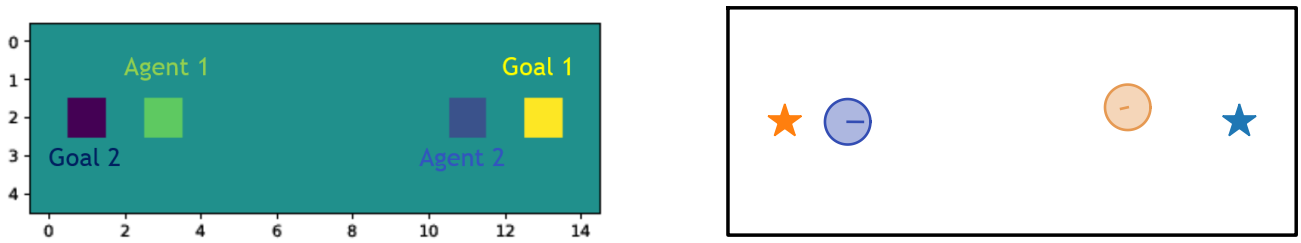


Figure 2: Two variants of the sidewalk problem. The goal is for each agent to navigate to its respective goal without colliding with the walls or other agent. Left: Discrete Domain. Right: Continuous Domain. The agents are modeled as unicycles, goals are depicted as stars.

successor states with their actions. In this document we use the subscript/superscript notation $variable_{time}^{agent,action}$. In a stochastic game, at time t each agent i in state s_t takes an action a_t^i according to the policy π^i. All the agents then transition to the state s_{t+1} and receive a reward r_t^i.

Stochastic games can be described as as tuple $\langle \mathcal{S}, \mathbf{A}, P, \mathbf{R} \rangle$, where $\mathcal{S}$ is the set of states, and $\mathbf{A} = \{\mathcal{A}^1, \ldots, \mathcal{A}^m\}$ is the joint action space consisting of the set of each agent's actions. Here m denotes the number of agents. The reward functions $\mathbf{R} = \{\mathcal{R}^1, \ldots, \mathcal{R}^m\}$ describe the reward for each agent $\mathbf{R} : \mathcal{S} \times \mathbf{A} \to \mathbb{R}$. The transition probability function $P : \mathcal{S} \times \mathbf{A} \times \mathcal{S} \to [0, 1]$ describes how the state evolves in response to all the agents' collective actions.

Given a single decentralized agent i, the goal is to learn a policy π^i that maximizes the expected return

$$R_i = \sum_{t=0}^{T-1} \gamma^t r_t^i \ , \tag{1}$$

where $0 < \gamma < 1$ is a discount factor that imposes a decaying credit assignment as time increases and allows for numerical stability in the case of infinite horizons.

Note that the definition has as different reward variable for each agent at each time step. This allows for the rewards to be independent. However in practice the rewards between agents are often correlated. For example, in an adversarial setting like a zero-sum game $\sum_{i=1}^{m} r^i = 0$, one agent receiving a reward results in a penalty to other agents. In domains like foraging tasks [15], penalties are often not explicitly encoded, but the limited resources often impose an adversarial setting since resources collected by one agent are not available for another. Similarly, a reward can be structured to encourage cooperation. For example [29] uses a reward

$$r^{i\prime} = r^i + \alpha r^j \ , \tag{2}$$

where α is a constant that adjusts the agent's *attitude* towards being cooperative. Just as domains might implicitly be adversarial, domains may also be implicitly cooperative. For example, in autonomous driving, collisions are bad for both agents, so both agents collaborate to avoid accidents.

The sidewalk problem is an example of the latter, each agent wants an efficient path that avoids collisions. The self-serving interest of taking the optimal path trades off against the collaborative interest of not colliding.

3 IMPLEMENTATION DETAILS

The sidewalk problem can be framed in a discrete or continuous setting as shown in Figure 2.

3.1 State and Action Spaces

In the discrete setting the problem is a multi-agent variant of grid world [27], agents occupy space in a 2D grid and navigate to a goal. For the state representation, we use a grayscale image of the grid world where each agent and goal have a unique fixed value to prevent aliasing, see the left image of Figure 2. In the continuous setting, agents have a position in 2D Cartesian coordinates. The state is represented as a vector with four real valued parameters per agent corresponding to the position, velocity, and rotation of the agent: $\{x, y, v, \theta\}$. We implement both versions in python as OpenAI gym environments [1].

Agents in the discrete setting have four actions: `UP`, `DOWN`, `LEFT`, and `RIGHT`. The agent stops moving when it reaches the goal.

In continuous space, the agent follows a unicycle model: the agent can move forward, backward, or rotate. The action space is two dimensional where the agent's control is an acceleration and a rotation. Rotations are bounded by $\pm\pi/8$ per time step and accelerations are bounded by $\pm 1.0 m/s^2$ per time step with a max velocity of $1.0 m/s$. The control is selected so that the agent cannot instantaneously stop.

3.2 Rewards

An agent is rewarded for reaching the goal and penalized for colliding with the other agent or the boundary of the world. A small negative step cost is added to encourage the agent to reach the goal efficiently.

In the sidewalk problem, there is a c_{step} step cost, a c_{goal} reward for reaching the goal, and $c_{collision}$ reward for colliding into a wall or other agent. When discussing a single agent in a multi-agent setting we refer to the single agent under consideration as the *ego* agent. To encourage cooperation, the ego agent can be rewarded for either agent reaching the goal *i.e.* each agent gets a combined reward of $c_{goal}^{ego} + c_{goal}^{other}$ if both agents reach the goal. Some care is needed to balance the step cost with the collision penalty to prevent the agents from getting stuck in the local optima

[1] https://gym.openai.com/

of remaining still for the entire trial. In our experiments we used $c_{step} = -0.01, c_{goal} = +0.5, c_{collision} = -0.5$. This gives a reward of +1 if both agents reach the goal.

This reward structure sets an explicit reward for collaboration. As mentioned, the domain is implicitly collaborative, and the agent should learn to avoid collisions without being rewarded for the other agent's success. We explore both explicit and implicit rewards in Section 6.

Additionally, we consider using reward shaping [3, 20, 22] to improve learning. In the discrete domain reward shaping is used to encourage the agents to follow the center line by penalizing lateral motions. This encourages the agent to deviate from the path as little as possible, and drives the agents to interact with each other. We can accomplish this with an $c_{center}d$ reward where d is the distance from the central axis. We use a value of $c_{center} = -0.01$ in our experiments.

The continuous domain is more difficult to learn. To direct the agents to their goal, potential-based reward shaping is used to make states further from the goal more negative. This provides a gradient signal that encourages the agent to move towards the goal and allows the agent to learn more quickly. We use a $c_{potential}d^2$ reward per time step where d is distance to the goal. A value of $c_{potential} = -0.0001$ is used for our experiments.

3.3 Randomness

We can add uncertainty to both the state and action to ensure the system has a robust policy. In the discrete setting agents take a random action with probability n. We run experiments with different amounts of randomness $n = \{0.0, 0.1, 0.2\}$.

In the continuous setting we add uniformly distributed random noise to the actions and observations. the noise ϵ is selected from the ranges $\epsilon = \{\pm 0.01, \pm 0.5, \pm 1.5\}$.

3.4 Timeouts

To bound the maximum length a given trial can run for, the environment resets after a set number of steps. In our experiments, the environments timeout at 300 time steps.

4 AGENTS

There is a chicken and egg problem developing interacting agents: in order to learn to interact with an intelligent agent we first need an intelligent agent for our ego agent to interact with. We use three different place-holder agents for the other agent.

4.1 Random

Random agents mostly act as obstacles. They are unlikely to reach the goal themselves, and their random behavior makes it so that a single fixed trajectory is unlikely to succeed in all trials.

4.2 Dijkstra and Probabilistic Road Map

The shortest path to the goal can be computed using Dijkstra's algorithm. The costmap of transitions is created by weighting pixels close to the ego agent as having high cost, so that the other agent seeks to navigate around our agent as shown in Figure 3. The path is recomputed at every time step. Since our grid world is relatively small, recomputing has a small computational cost, but if larger

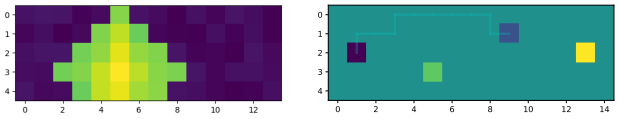


Figure 3: Costmap and a Dijkstra agent's resulting plan.

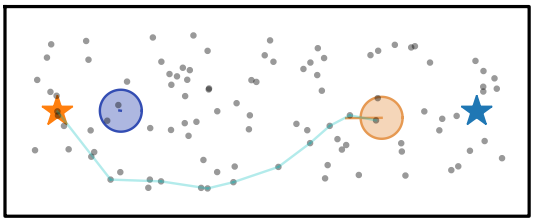


Figure 4: Probabilistic Roadmap

worlds are of interest, the search could be made more efficient using A* in place of Dijkstra and/or using algorithms that include anytime planning [17] or replanning [16].

In the continuous domain we use a probabilistic road map (PRM) [14] to discretize the search before running Dijkstra. An agent using a PRM plan often succeeds. Additionally the ego agent gets rewarded for getting out of the way, so it is often possible to outperform a domain with a random agent by simply getting out of the way and not crashing. A visualization of an agent following a PRM is shown in Figure 4.

4.3 Centralized

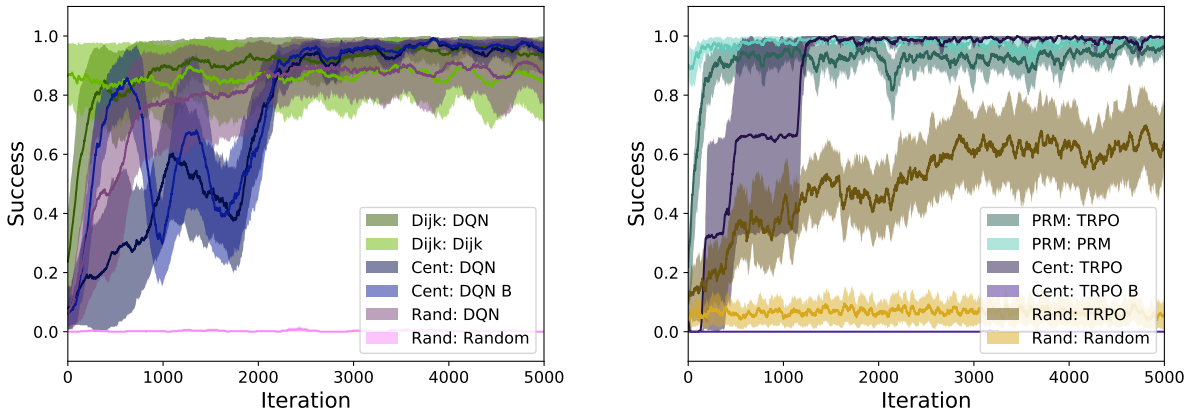
Our third control method gives control of both agents to the planning/learning algorithm[2]. This results in a centralized strategy.

A centralized strategy simplifies the problem. Since the system has control over both agents, there is no longer a need to interact or re-plan. It has been shown that decentralized strategies can also avoid interaction and replanning when the other agents' policies are known *e.g.* both agents move to their left.

5 SAFETY

An application we think the sidewalk domain is particularly well suited for is safe RL. The random exploration typical to RL methods is not appropriate to physical systems which act in the real world where failure cases result in real consequences. There have been numerous efforts at designing learning processes with safe exploration to mitigate the risks involved in training an RL system. Broadly speaking, approaches can be classified into approaches that modify the objective function [9, 11, 12, 18] and approaches that

[2]the specific learning algorithms we use are described in section 6.



(a) Discrete **(b) Continuous**

Figure 5: Success by Agent. In the Discrete setting (Left) we observe the most robust policy is learned by the centralized algorithm. This is expected since a centralized algorithm avoids the difficulty of agent interaction. In the continuous domain, we again see the centralized algorithm learns the most robust policy, but note that because the potential based reward shaping was only associated to one agent, the other agent learned to get out of the way, and did not learn to reach the goal.

constrain the search space [2, 10, 26, 32, 33]. A review of safe RL is presented by Garcia and Fernandez [8].

When safely learning the sidewalk problem, the agent should never collide with the other agent or the walls. The sidewalk problem presents both a discrete and continuous domain that can be used as a benchmark to explore safe RL.

6 EVALUATION

In this section we run experiments examining the performance of learning algorithms against different agent types. We look at the effects of noise on the agents, and make observations about different reward functions.

6.1 Experimental Setup

Open AI has released code for several different types of deep learning agents implemented in TensorFlow [7]. The code base provides a standardized test bed, allowing researchers to reproduce and build upon the state of the art. We use the deep Q network (DQN) and Trust Region Policy Optimization (TRPO) baseline implementations to train agents on the sidewalk problem.

For handling discrete state/action spaces we use a DQN [21, 30] which applies the non-linear function approximation of deep learning to boot-strapped procedures for learning a Q-function. The DQN we use is a convolutional neural network with with 32 4×4 filters with a stride of 2, 64 4×4 filters with a stride of 2, and 64 2×2 filters with a stride of 1. The final fully connected hidden layer has 256 nodes. The DQN uses a dueling architecture [30].

Trust Region Policy Optimization (TRPO) [25] is an approximate policy iteration technique which we use to learn a policy for our continuous state and action space sidewalk problem. The TRPO network uses a multi-layer perceptron architecture with two hidden layers, each with 32 nodes. This is the TRPO default in Open AI baselines.

The networks are trained with different levels of noise for 2000 iterations in the discrete domain, and 5000 iterations in the continuous domain. For comparing different agents we increase the discrete domain to 5000 iterations because the learning of the centralized agent had not stabilized. Results are averaged over three trials and the shaded regions depict the standard errors.

6.2 Results

In the discrete setting, the DQN agent is able to learn a reasonable policy in the presence of the three different agents. Results are shown in Figure 5.

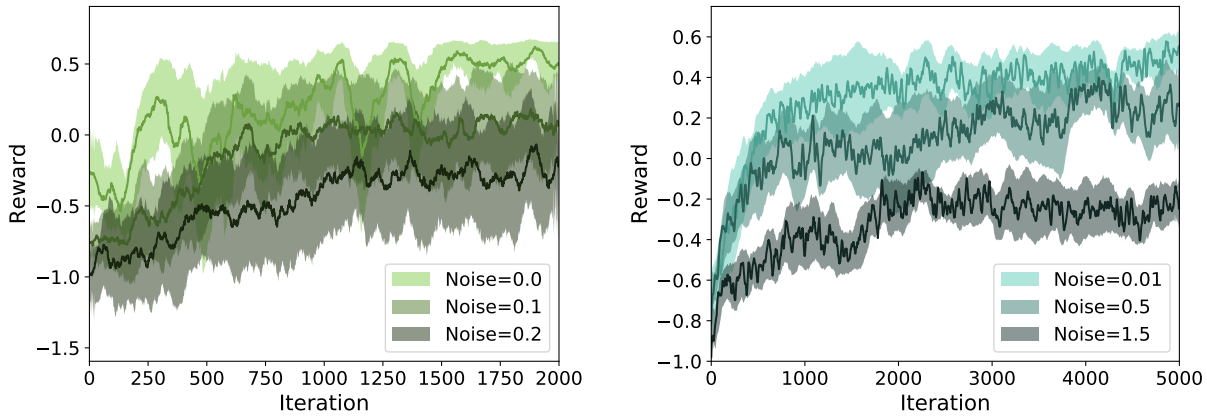
Typically the random agent (shown in light pink) blocks the goal before eventually colliding with a wall. The DQN agent (dark pink) is able to learn to a policy that is often able to avoid the random agent and reach the goal.

The Dijkstra agent (light green) is often successful at reaching the goal. The costmap often pushes the Dijkstra agent close to the wall to avoid the ego agent, which has some possibility of resulting in a collision as the randomness increases. The DQN agent (dark green) tends to head straight towards the goal, minimizing the lateral offset penalty, and forcing the Dijkstra agent to adjust.

The centralized DQN can learn to get both agents to their goals, but the system has a harder time learning a good policy as evidenced by requiring more training iterations and less monotonic learning curves.

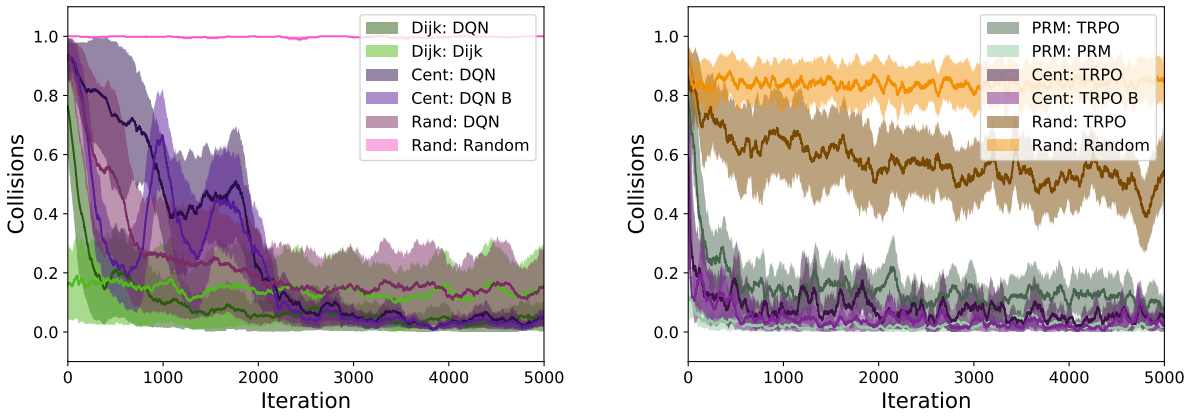
In the continuous setting, we again see the random agent (orange) block the goal, and often collide into walls. A visualization of a representative trajectory is shown in Figure 10d.

While the average success of TRPO agents trained with the centralized and PRM agents are similar, the performance is quite different. In the centralized domain, the potential based reward



(a) Discrete (b) Continuous

Figure 6: Varying Noise. This investigation was to determine how robust both the discrete and continuous agents are to noise.



(a) Discrete (b) Continuous

Figure 7: Collision by Agent. We see that the collisions of the centralized agents drop quickly compared to the learning curves realted to success. This indicates that first the agents learn to avoid collisions and then the agent with the potential reward learns to reach the goal.

shaping is relative to the ego agent only. We were interested in how the reward shaping affected the agent. We see that the reward shaping is important to learning in the continuous domain. The centralized policy learned to move the agent without reward shaping (light purple) safely out of the way, allowing the agent with reward shaping (dark purple) to move straight for the goal. This had the effect of minimizing the potential cost, but only one agent reached the goal. A sample trajectory is shown in Figure 10b. For comparison, a sample trajectory where both agents have reward shaping is shown in Figure 10c. It is interesting that the centralized algorithm that only uses reward shaping on one agent gets stuck in a local optima, even though we see that the random agent does occasionally reach the goal.

The TRPO agent (dark blue) trained with the PRM agent (light blue) gets to the goal in most trials but takes more circuitous paths, adjusting for the other agent. The simple control used for the PRM agent often successfully gets the agent to the goal, but can be a bit slow and sometimes the system times out before the agent reaches the goal. Also because the agent's action is harder to predict, there are more instances when the ego agent collides with the PRM agent. A sample trajectory is shown in Figure 10a.

We also plot collision ratios in Figure 7. The collisions mostly contain the same information as the successes, but comparing the two plots, it can be seen that centralized TRPO first learns to avoid collisions, and then starts moving toward the goal. Additionally, we see that TRPO with a random agent makes faster progress when

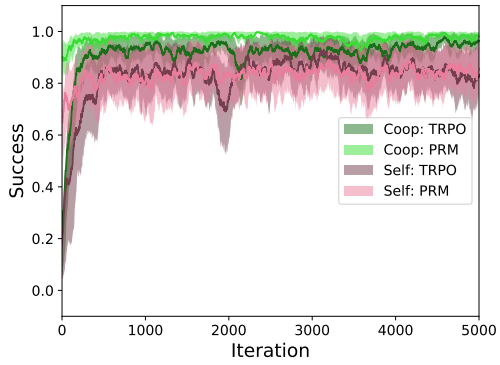


Figure 8: The difference in implicit vs. explicit reward. We see that even though the agent learns good behavior with an implicit reward, having an explicit reward appears to improve performance.

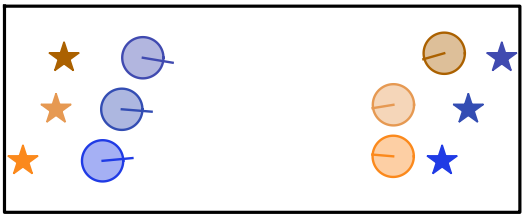


Figure 9: Extension to Multi-agent systems where instead of two agents there are two groups of agents. If there is not sufficient room for any agent to move to the side, it creates an increased need to coordinate and time movement through space.

it comes to reaching the goal, and comparatively slower progress when it comes to avoiding collisions.

The plots in Figure 6 show the effect of noise on the learned system. As expected more noise makes the problem more difficult. This test was to determine what amounts of the noise the agents can handle.

We also looked at the difference between explicitly rewarding an agent for the other agent's good performance, vs keeping the reward self-centered. Results are shown in Figure 8. We see that explicitly specifying the reward leads to improved performance. We hypothesize that this is because the explicit reward more strongly encourages safe interactions, and discourages paths that force the PRM agent to adjust in ways that might lead to a collision. Representative trajectories of each approach are shown in Figure 10.

7 GENERALIZATION AND OTHER BENCHMARKS

The sidewalk problem can trivially be extended to multiple agents. However, there are several possible strategies to layout the agents. One strategy is to randomly generate agents and goals. A more structured bottleneck can be created by positioning agents in a circle where goals are on the opposite side of the circle, thereby driving all agents to pass through the center of the circle. Alternatively, agents could be arranged in teams as shown in Figure 9. With multiple agents side by side there is an increased need to monitor space and time as some agents will have to wait to allow other agents to pass.

While the multi-agent setting introduces and amplifies important challenges, having many agents can make it more difficult to analyze what is happening. For this reason, we sketch the idea for two other baselines for cooperative decision making.

To incorporate the need to wait into temporal decision making problem, we propose a problem we call the three stooges problem. Two agents each wish to pass through a door to get to a goal on the other side. Rewards are based on the time it takes to reach the goal, however the door is too narrow for both agents to pass through and a collision results in a failure. As a result, agents must learn to appropriately wait or not wait, in order to prevent collisions and also not freeze.

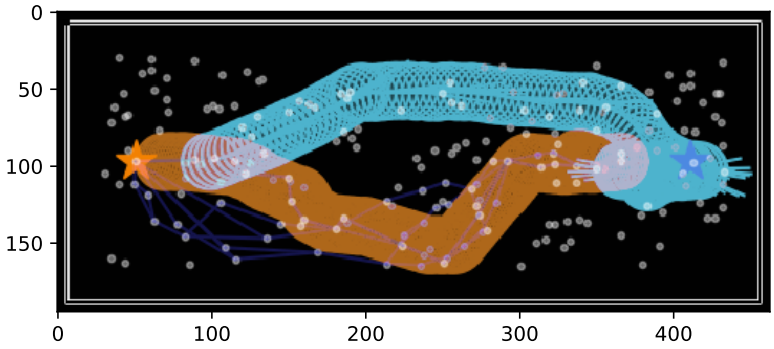
In the three stooges problem, a deadlock can occur if both agents wait at the door. More emphasis can be placed on resolving deadlocks in a third benchmark problem, which we call the zipper merge. In the zipper merge problem, two cars drive in adjacent lanes with each car wishing to navigate to the other cars lane.

8 CONCLUSION

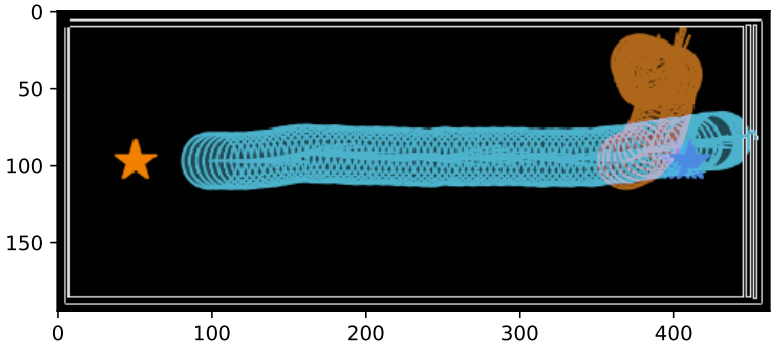
We proposed the sidewalk problem as a benchmark for cooperative decision making. We formalized the benchmark problem as a stochastic game, described in detail how the problem could be implemented as either a discrete or continuous domain, and explored several design choices related to the problem including the nature of the reward, the amount of noise affecting the agents, and the choice of algorithm to use for the opposing agent to create different situations. Additionally, we discussed how the sidewalk problem could be extended and sketched out other possible ideas for cooperative game benchmarks.

REFERENCES

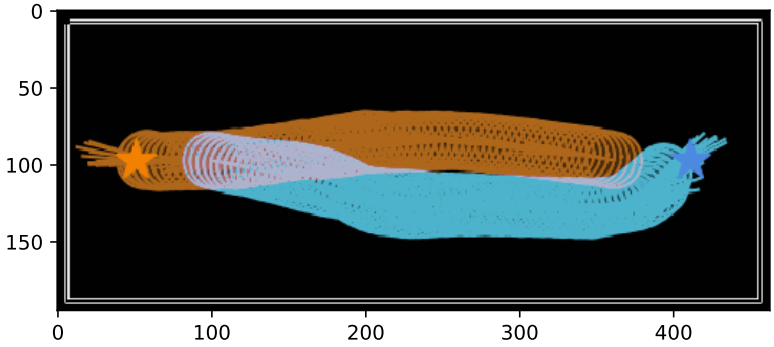
[1] Gabriel Agamennoni, Juan I Nieto, and Eduardo M Nebot. 2012. Estimation of multivehicle dynamics by considering contextual information. *IEEE Transactions on Robotics* 28, 4 (2012), 855–870.
[2] Mohammed Alshiekh, Roderick Bloem, Ruediger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. 2018. Safe reinforcement learning via shielding. *Proc. AAAI* (2018).
[3] John Asmuth, Michael L Littman, and Robert Zinkov. 2008. Potential-based Shaping in Model-based Reinforcement Learning.. In *AAAI*. 604–609.
[4] Michael Bowling and Manuela Veloso. 2000. *An analysis of stochastic game theory for multiagent reinforcement learning*. Technical Report. Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science.
[5] Michael Bowling and Manuela Veloso. 2001. Rational and convergent learning in stochastic games. In *International joint conference on artificial intelligence*, Vol. 17. Lawrence Erlbaum Associates Ltd, 1021–1026.
[6] Ronen I Brafman and Moshe Tennenholtz. 2002. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research* 3, Oct (2002), 213–231.
[7] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. 2017. OpenAI Baselines. https://github.com/openai/baselines. (2017).
[8] Javier Garcıa and Fernando Fernández. 2015. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* 16, 1 (2015), 1437–1480.
[9] Peter Geibel and Fritz Wysotzki. 2005. Risk-sensitive reinforcement learning applied to control under constraints. *J. Artif. Intell. Res.(JAIR)* 24 (2005), 81–108.
[10] Alexander Hans, Daniel Schneegaß, Anton Maximilian Schäfer, and Steffen Udluft. 2008. Safe exploration for reinforcement learning.. In *ESANN*. 143–148.



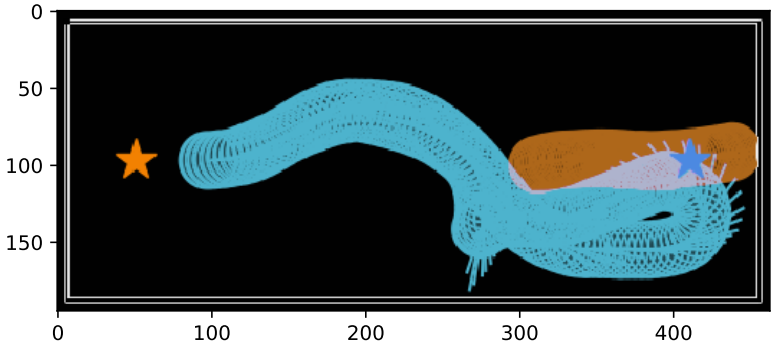
(a) TRPO agent (blue), PRM agent (orange)



(b) Centralized TRPO agents with (blue) and without reward shaping (orange).



(c) Centralized TRPO agents both with reward shaping.



(d) TRPO agent (blue), and Random agent (orange).

Figure 10: Representative trajectories.

[11] Matthias Heger. 1994. Consideration of risk in reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier, 105–111.

[12] Ronald A Howard and James E Matheson. 1972. Risk-sensitive Markov decision processes. *Management science* 18, 7 (1972), 356–369.

[13] Junling Hu and Michael P Wellman. 2003. Nash Q-learning for general-sum stochastic games. *Journal of machine learning research* 4, Nov (2003), 1039–1069.

[14] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation* 12, 4 (1996), 566–580.

[15] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. 2017. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 464–473.

[16] Maxim Likhachev, David I Ferguson, Geoffrey J Gordon, Anthony Stentz, and Sebastian Thrun. 2005. Anytime dynamic a*: An anytime, replanning algorithm.. In *ICAPS*. 262–271.

[17] Maxim Likhachev, Geoffrey J Gordon, and Sebastian Thrun. 2004. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in neural information processing systems*. 767–774.

[18] Zachary C Lipton, Jianfeng Gao, Lihong Li, Jianshu Chen, and Li Deng. 2016. Combating Reinforcement Learning's Sisyphean Curse with Intrinsic Fear. *arXiv:1611.01211* (2016).

[19] Michael L Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*. Elsevier, 157–163.

[20] Patrick Mannion, Sam Devlin, Karl Mason, Jim Duggan, and Enda Howley. 2017. Policy invariance under reward transformations for multi-objective reinforcement learning. *Neurocomputing* 263 (2017), 60–73.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.

[22] Andrew Y Ng, Daishi Harada, and Stuart Russell. 1999. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, Vol. 99. 278–287.

[23] Sébastien Paris, Julien Pettré, and Stéphane Donikian. 2007. Pedestrian reactive navigation for crowd simulation: a predictive approach. In *Computer Graphics Forum*, Vol. 26. Wiley Online Library, 665–674.

[24] Nuria Pelechano, Jan M Allbeck, and Norman I Badler. 2007. Controlling individual agents in high-density crowd simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 99–108.

[25] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

[26] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving. *arXiv:1610.03295* (2016).

[27] Richard S Sutton and Andrew G Barto. 1998. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge.

[28] Peter Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. 2013. Robot navigation in dense human crowds: the case for cooperation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2153–2160.

[29] Weixun Wang, Jianye Hao, Yixi Wang, and Matthew Taylor. 2018. Towards Cooperation in Sequential Prisoner's Dilemmas: a Deep Multiagent Reinforcement Learning Approach. *arXiv preprint arXiv:1803.00162* (2018).

[30] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581* (2015).

[31] Chen-Yu Wei, Yi-Te Hong, and Chi-Jen Lu. 2017. Online Reinforcement Learning in Stochastic Games. In *Advances in Neural Information Processing Systems*. 4994–5004.

[32] Min Wen, Rudiger Ehlers, and Ufuk Topcu. 2015. Correct-by-synthesis reinforcement learning with temporal logic constraints. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 4983–4990.

[33] Min Wen and Ufuk Topcu. 2016. Probably Approximately Correct Learning in Stochastic Games with Temporal Logic Specifications.. In *IJCAI*. 3630–3636.