Symmetry Agnostic Learning for Multi-Robot Zero-Shot Coordination*

Kegan J. Strawn[1] and Nora Ayanian[2]

Department of Computer Science, University of Southern California, Los Angeles, California, USA {kegan.j.strawn, ayanian}@usc.edu

Abstract. As reinforcement learning for robotics continues to grow, it is essential to consider the critical points that may fail. Robots collaborating with other robots (or humans) must be careful when optimizing the joint action reward in a multi-agent reinforcement learning solution. Recent work has shown that learning agents may differ in their decisions between symmetric joint actions. Our algorithm, Automatic Symmetry Tracking Other-Play (ASTOP), tracks the rewards the agent sees during learning to automatically avoid symmetric actions that lead to suboptimal performance in cooperative scenarios. We run multiple simulated and physical quadrotors in a location-selection game and evaluate the performance of ASTOP in the zero-shot coordination problem across graphs that vary in complexity and represent a wildfire response application. ASTOP policies never arbitrarily break ties between symmetric actions and correctly select the maximum symmetry-safe reward for all location-selection game graphs.

Keywords: Multi-Robot Systems · Multi-Agent Reinforcement Learning · Zero-Shot Coordination.

1 Introduction

Deep reinforcement learning has successfully produced beyond-human level policies for two-player zero-sum games such as Go, poker, chess, and more [5,24,34]. In the popular Self-Play (SP) reinforcement learning method, agents optimize the discounted expected reward by training against themselves [36]. In cases with *symmetries*, where multiple actions yield the same reward, it has been shown that SP-trained agents can arbitrarily converge on an optimal action when other equivalent actions exist [16]. For example, given two symmetric options with equal rewards, SP-trained policies will choose one option without considering that other agents might choose the other option. While many researchers in multi-agent reinforcement learning (MARL) apply SP learning to collaborative games [7, 12, 16, 20, 37], SP agents might disagree on which symmetric joint actions to choose. We propose a method to automatically find and avoid the joint actions that lead to arbitrary learned behaviors when symmetries are present to

* This work was supported by NSF grant 1724399.

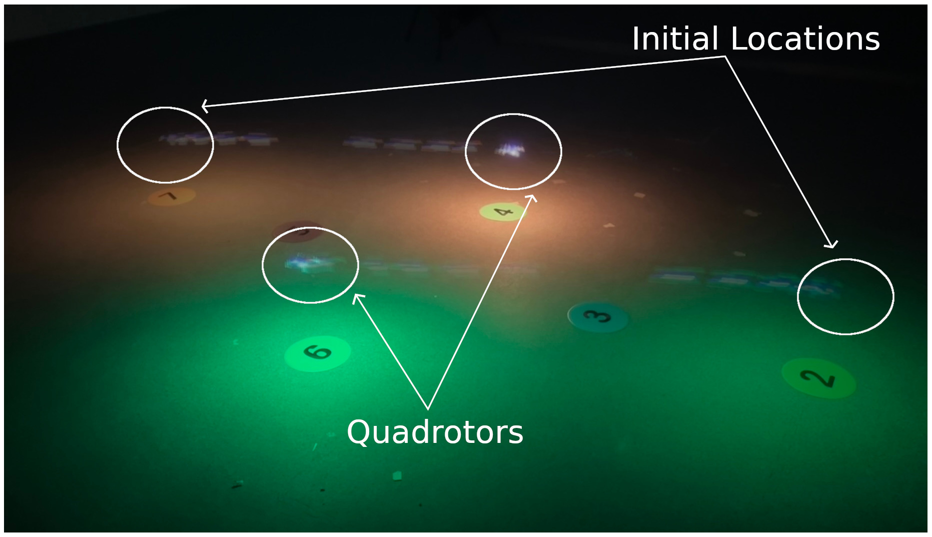

Fig. 1. Long exposure of 2 Crazyflie quadrotors flying to their chosen locations.

produce policies aware of the ambiguity. We test these policies in simulation and demonstrate them on quadrotors as shown in Fig. 1.

There are many challenges in applying reinforcement learning in collaborative scenarios such as autonomous vehicles [17,30,40], delivery robots in crowded pedestrian environments [8–10], or fire fighting robots collaborating with human firefighters and ground vehicles [1,18,27,29,31]. It is impossible to know exactly how all agents will behave, and anticipating the joint actions of all nearby agents depends on unobservable information (e.g., vehicles' or pedestrians' goals). Policies that seem optimal when training individual agents might in the cooperative setting choose actions that are at best well below optimal, and at worst, dangerous to others. Understanding where symmetries can occur and accounting for them helps reduce uncertainty in deployment if all agents are trained with this knowledge, increasing the predictability of a cooperative system, and thus, its safety.

In this work, we focus on scenarios with symmetries where all agents agree on the same task but do not know how the other agents will solve it. Inspired by previous work to improve SP learning [16] and the pursuit of deploying drones to help combat fires [27], we formulate an abstract location-selection game. In this game, each robot must choose then navigate to a location so as to maximize the joint reward without communication, as shown in Fig. 2. Each robot trains on the graph individually using the proposed algorithm, ASTOP, and has no information sharing ability with other agents.

The contribution of this work is an algorithm (ASTOP) that, through training, automatically finds and avoids equivalent joint actions that lead to a lower joint reward, and the successful application of this algorithm, including train-

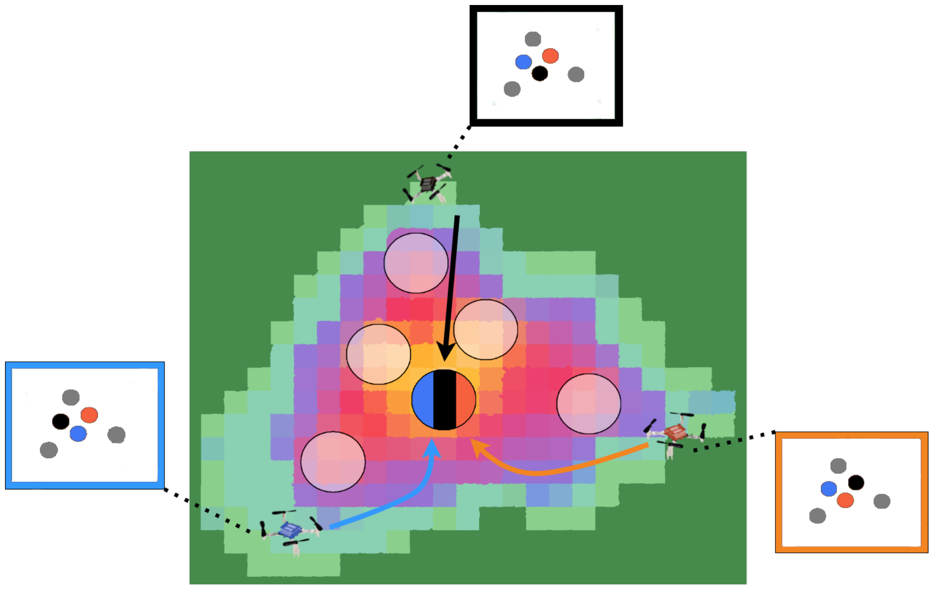

Fig. 2. A representation of three Crazyflie quadrotors running SP policies that incorrectly choose the same location. Each robot uses the estimated value of a location to learn a belief over the other robots' decisions. In the fire fighting example, this could be derived from the heatmap of the area or other context-specific reward information.

ing and transfer of the learned policies to simulated and physical agents, to a multi-robot game, which we call the location-selection game.

2 Background

We focus on a fully cooperative game where the joint action reward, conditioned on the joint action and state, is shared by all robots. The goal of MARL is to generate functions that map Markov decision processes to joint policies for each agent [26]. Most existing work on cooperative MARL trains agents together, but agents execute their learned policies independently during testing. Individual agents maximize the expected (discounted) joint reward. Maximizing this reward can create an unpredictable agent that develops arbitrary behaviors over equivalent, yet incompatible, actions during learning.

2.1 Self-Play

In SP MARL, the agents train against themselves before testing independently. During training, an SP agent will arbitrarily break ties between equally optimal actions [16]. For competitive zero-sum games, all optimal policies are interchangeable [25], so SP MARL performs well [36]. In a cooperative setting when symmetries are present, SP agents can fail to coordinate when trained separately from their testing collaborators [16]. Thus, an SP agent's assumption that other agents will behave similar to itself (or other policies they encountered during training) is not realistic when a robot is in an environment with unknown or unpredictable agents.

2.2 Zero-Shot Coordination

In reality, robots need to perform well in testing against robots trained separately. The zero-shot coordination (ZSC) problem focuses on achieving high returns with partners who are trained separately and cannot communicate their planned actions ahead of time. For robots planning trajectories from start to goal locations, they must take an action even though the goals of all other individual robots is unobservable. While they can agree upon the environment, the game, and the training algorithms used, they cannot agree upon policy-specific behaviors or use communication ahead of time.

2.3 Other-Play

Other-Play (OP) successfully enhanced SP by randomly permuting within a symmetry the equivalent actions an agent might choose to break the symmetry [16]. They define a symmetry as an arbitrary relabeling of the state/action space that leaves the resulting action sequences unchanged up to relabeling during training. In OP, when an agent attempts to take action as part of the joint action within a symmetry, the action in the joint action symmetry is permuted to return any of the resulting joint action rewards. OP policies may reach a suboptimal reward during training but perform better than SP policies during ZSC testing. Furthermore, Hu et al. [16] prove that an OP policy will not converge on one specific action in a given symmetry.

The OP algorithm uses given symmetries in the underlying Markov game. The authors applied OP to both a matrix game where the simulated agents choose levers and a simulated card game called Hanabi. It is difficult to apply OP to robotics problems because the algorithm requires the user to specify the symmetries a priori in the OP code. This requires complex knowledge of the underlying game and additional time to implement successfully.

2.4 Automatically Finding Symmetries

The more nodes and edges a graph contains, the more time it takes to search for and specify all potential symmetries prior to training. Users may also waste time searching for symmetries that do not exist or that agents will not encounter. We instead propose a method that does not require prior user specification of the symmetries. An automatic process for discovering symmetries does not rely on a user's experience or expertise for success, thus reduces uncertainty in the outcome. The proposed solution extends OP to automatically find these equivalent joint action symmetries and apply them during training to optimize the reward for ZSC testing, rather than relying on a user to specify them.

3 Related Work

Recent work showcases the possibility of finding quasi-equivalent policies to address the symmetry problem, presenting a communication problem setting for zero-shot coordination in referential games [6]. Referential games in communication require receivers to predict what senders are referring to through the messages sent. They discover a transformation matrix to identify equivalent policies in training, showing the importance of finding these equivalences. Similarly, our work automatically tracks the equivalent actions, but in a different, direct method applied to a novel location picking game for robotics.

OP and our approach are similar to domain randomization for reinforcement learning. Domain randomization attempts to produce a model invariant to features of the environment [38], and we produce a policy that is invariant to how an agent's collaborators break symmetries.

Deciding between symmetries is related to the problem of equilibrium selection in game theory [13]. Here, the equilibrium selection in the ZSC problem is choosing between different optimal symmetric policies in a fully cooperative game. A general solution for equilibrium selection is in the framework of standard-form games and stochastic games, with a large body of work in the theory of mind in collaboration and deciding between actions [13–15, 33]. ZSC and these frameworks share the belief that a solution should not depend on arbitrary labels [13]. However, these previous solutions do not scale well, and the standard-form games have less structure than full Dec-POMDPs.

There exists research on the coordination problem with and without joint learning. For example, agents randomize between optimal actions until they successfully coordinate [4], quickly exchange messages and employ probabilistic belief updating schemes [11], or perform other dynamic programming methods for fully observable and cooperative stochastic games [32]. In ad-hoc teamwork, methods produce an average response to a diverse set of agents [2, 3, 35]. Alternatively, some agents learn and imitate responses from observational data of other agents' behaviors [23,39]. There exist many approaches to solving decision-making under uncertainty in controls, search, and learning [21]. While all of these approaches seek to produce robust strategies, they require either specifying the other agents' behaviors or learning to respond to a specific set of agents rather than the more general problem that we focus on here of zero-shot coordination.

4 Approach

Our work extends OP to automatically find and apply symmetries during training to produce optimal action agnostic policies in testing on real robots in the location-selection game.

4.1 Problem Formulation

The Dec-POMDP model considers joint actions and observations. A Dec-POMDP [26] is defined as $M = \langle \mathbb{D}, \mathbb{S}, \mathbb{A}, \mathbb{O}, T, O, R, b_0 \rangle$ where $\mathbb{D}$ is the set of n agents and $\mathbb{S}$

is a finite set of states. Here, $\mathbb{A} = \times_{i \in \mathbb{D}} \mathbb{A}_i$ is the set of joint actions where $\mathbb{A}_i$ is the set of actions available to agent i. At time t, each agent i takes an action $a_{i,t}$, leading to a joint action $a = \langle a_1, \dots, a_n \rangle$[1]. Similar to the joint actions, $\mathbb{O} = \times_{i \in \mathbb{D}} \mathbb{O}_i$ is the set of joint observations where $\mathbb{O}_i$ is the set of observations available to agent i. At time t, the environment emits a joint observation $o = \langle o_1, \dots, o_n \rangle$ where each agent i only observes their component o_i. In M, T is the transition probability function $T(s'|s,a)$, O is the observation probability function $O(o|s',a)$, R is the immediate reward function, and $b_0 \in \Delta(\mathbb{S})$ is the initial state distribution at time $t = 0$. The joint action-observation history τ is all joint actions taken and joint observations seen up to time t.

The development follows closely that of Hu et al. [16]. We define a symmetry as an arbitrary relabeling of the state/action space that leaves the resulting action sequences unchanged up to relabeling during training. These symmetries occur when actions lead to equivalent rewards. A set of symmetries for a given Dec-POMDP is a set Φ where each element ϕ is a bijection of $\mathbb{S}$, $\mathbb{O}$, and $\mathbb{A}$ onto itself such that it leaves the Dec-POMDP unchanged[2]:

$$\begin{aligned} \phi \in \Phi \Leftrightarrow\; & T(\phi(s')|\phi(s), \phi(a)) = T(s'|s,a) \\ & \wedge R(\phi(s'), \phi(a), \phi(s)) = R(s', a, s) \\ & \wedge O(\phi(o)|\phi(s), \phi(a), \phi(i)) = O(o|s,a,i) \end{aligned}$$

$$\pi' = \phi(\pi) \Leftrightarrow \pi'(\phi(a)|\phi(\tau)) = \pi(a|\tau), \forall \tau, a.$$

In the two agent case, self-play optimization can optimize both policies by selecting a maximum over both policies:

$$\pi^* = \arg\max\; J(\pi_1, \pi_2)$$

By applying the mapping ϕ an agent can only select an equivalence class and not a specific action. Thus, we maximize over an equivalence class of policies, preventing the convergence on an individual action in a symmetry in the zero-shot coordination test setting. The optimization over the two-agent joint policy with the ϕ mapping applied to the second policy is:

$$\pi^* = \arg\max\; \mathbb{E}_{\phi \sim \Phi}\; J(\pi_1, \phi(\pi_2)).$$

Here, the expectation is taken with respect to a uniform distribution on Φ.

4.2 Location-Selection Game

Let $G = (V, E)$ be a connected graph consisting of locations V, edges connecting locations E, and let $V_{\mathbb{D}} \subset V$ where $V_{\mathbb{D}}$ is the set of initial agent locations. The

[1] We write a_i and o_i, the action and observation for agent i (when t is left unspecified).

[2] Note that, as in OP [16], the notation here is overloaded since ϕ can act on actions, states, observation functions, and ultimately, trajectories and policies.

agents try to maximize the joint reward by selecting the location to travel to, given three functions: C a path cost function, L a location reward function, and R the immediate joint action reward function. There exists at least one edge connecting every initial agent location $v_i \in V_{\mathbb{D}}$, to a non-agent location $v_j \in V - V_{\mathbb{D}}$. Two locations are adjacent if there exists an edge between them $e_{(v_i,v_u)} \in E$. For any path $p_{i,j} = \langle e_{(v_i,v_u)}, ..., e_{(v_w,v_j)} \rangle$ between two locations v_i and v_j where all edges in $p_{i,j}$ are adjacent pairs of $v_u, ..., v_w \in V - V_{\mathbb{D}}$, there is a path cost $C(p_{i,j})$ equal to the cumulative cost of edges in $p_{i,j}$. For every non-agent location there is a location reward $L(v_j)$. An agent k takes an action a_k, as part of a joint action a, to choose their goal location v_{a_k} from their initial location v_k. Every joint action has a joint reward $R(a) = \frac{1}{n} \sum_{k=1}^{n} L(v_{a_k}) - C(p_{k,a_k})$, however, $R(a) = 0$ if there is at least one duplicate location in the selected actions of the joint action, a.

4.3 Joint Action Tracking

We define a trajectory as a tuple of the joint actions that yield a reward and an equivalence class as a set of those trajectories leading to the same reward, collected as they are executed, in a hashmap. The rewards are the hashmap keys, and the trajectory sets are the hashmap values. To automatically track symmetries, when executing a joint action trajectory that we have not seen before, we insert the trajectory and its reward into the hashmap. In this way, all trajectories executed during training will be part of an equivalence class. All equivalence classes will contain one or more trajectories.

4.4 Joint Action Symmetry

Recall that a symmetry is an arbitrary relabeling of the state/action space that leaves the resulting action sequences unchanged up to relabeling. Where the joint action reward is the same, the assignment of agent-location pairings does not matter and is a symmetry. Permuting the different symmetric joint action trajectories will yield the same reward. Trajectories that lead to the same reward are interchangeable and can be stored to find symmetries. However, permuting the actions that an individual agent can select from within the symmetric trajectories might produce a different sub-optimal reward.

4.5 Symmetry Crossover

During training, when an agent is attempting to take action as part of a joint action trajectory, we first use the hashmap to get the equivalence class of symmetric trajectories found so far. Then, we sample a second trajectory from the same equivalence class to randomly mix each agent's selection between the two parent trajectories to produce a third trajectory[3].

[3] An equivalence class of size one would sample the same trajectory, resulting in no crossover and no changes to the expected equivalence class reward.

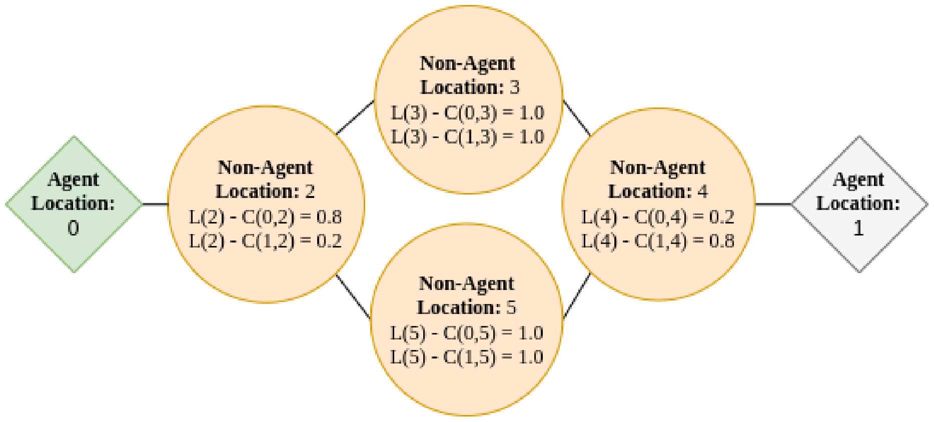

Fig. 3. Shown is a visual representation of a location-selection graph for two agents and four locations. The immediate joint action rewards are higher for the middle locations than those closer to the agents. A symmetry exists between the two middle locations for both agents.

This child trajectory is a crossover of two symmetric parent trajectories and is guaranteed to be feasible since all agents can reach all locations. The crossover method randomly swaps actions between parent trajectories. If the resulting child trajectory of the crossover method yields the same reward as the parent trajectories, the symmetry remains intact. However, if the child trajectory's reward is different from the parents' reward and not part of the same equivalence class, then the parents' equivalence class's expected reward will change. The agent will learn to avoid that class of joint action trajectories if, after updating, other higher reward equivalence classes exist.

For example, consider Fig. 3. Two possible joint-action trajectories are $b = \langle 3, 5\rangle$ and $c = \langle 5, 3\rangle$. Both trajectories yield $R(b) = R(c) = 1.0$ and therefore belong to the same equivalence class. If no crossover has yet to yield a lower value for the equivalence class containing b and c, then the expected reward for the class is 1.0. When agent 0 attempts to execute joint action b during training, performs the crossover method by sampling c from the equivalence class containing b and c, and randomly crosses the actions between b and c, it could produce a child trajectory $d = \langle 5, 5\rangle$. Agent 0 then takes this action and receives a reward of $R(d) = 0.0$. They then use $R(d)$ to update the expected value of the equivalence class for b and c. Doing so lowers the probability that agent 0 will choose locations 3 or 5 because the expected reward of the equivalence class will converge on 0.5.

We continue this tracking and use the equivalence classes during training to extend OP to produce policies aware of symmetries in the underlying problem and prevent a policy from converging onto an individual action within a symmetry. Note that this does not require specifying the symmetries ahead of time. An overview of the symmetry tracking and crossover methods is shown as pseudocode in Algorithm 1.

Algorithm 1: ASTOP Crossover

```
for each training iteration do
    $a \leftarrow \pi(\tau)$
    trackEquivalent($a$)
    $a^{other} \leftarrow$ sampleEquivalent($a$)
    newReward $\leftarrow$ crossover($a$, $a^{other}$)
    learningStep($\pi$, newReward)
```

4.6 Learning Method

We acknowledge that the action space in reinforcement learning can be inefficient at scale. However, work exists to narrow the scope of actions and efficiently prune irrelevant actions from planning and shape the action space [19, 22]. Future work is necessary to investigate and prove OP-based methods' efficiency and scalability. Deep reinforcement learning can apply function approximation and gradient-based optimization to solve the Dec-POMDP [16]. We use simple joint action learning in our work and are agnostic to the precise learning method used.

5 Experiments

We evaluate the performance of ASTOP in ZSC across location-selection games that range in the complexity of the underlying graph and the number of robots playing. We test robot policies trained separately with SP, OP, and ASTOP algorithms for each experiment. We do not hand-tune symmetries for OP on graphs with more than two robots. For these more challenging graphs, we can only compare SP and ASTOP.

5.1 Two Robots

All policies learn on the same constant graph (with two robots, four locations, and a set of edges as shown in Fig. 3). We show the expected reward when training and ZSC testing.

In Fig. 4 we see that SP policies achieve the maximum reward in training (1.0) but drop to a lower average reward in cross-play ZSC testing (below 0.6). SP performs worse because the policies arbitrarily break ties between the two symmetric trajectories. OP and our ASTOP algorithm converge on a lower reward in training (0.8), but outperform SP in the cross-play ZSC testing. OP and ASTOP policies cannot arbitrarily break ties between the two symmetric trajectories. OP and ASTOP correctly converge on 0.8, the maximum expected equivalence class reward for this specific graph.

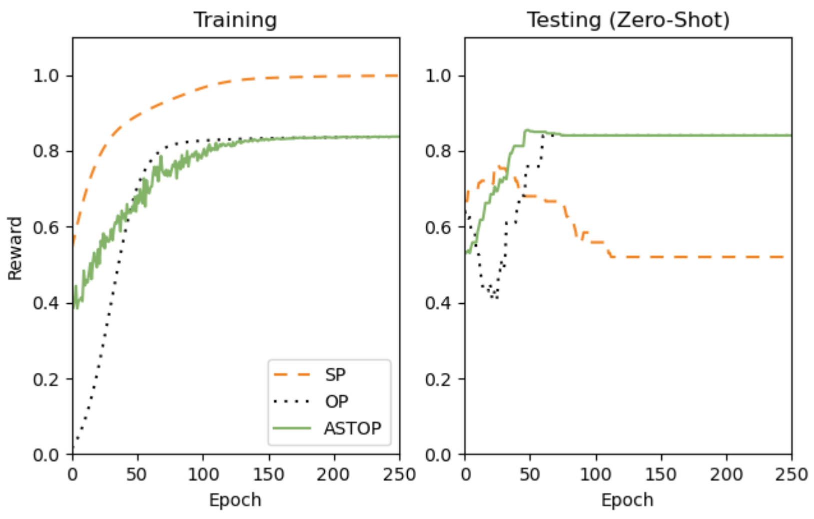

Fig. 4. Training and ZSC testing performance of SP, OP, and ASTOP algorithms on the location-selection graph in Fig. 3.

5.2 Three Robots

For the three robot case, the same constant graph is used with three robots, six locations, and a set of edges as shown in Fig. 5.

In Fig. 6 we see that SP policies again achieve the maximum reward in training (1.0) but drop to a lower average reward in cross-play ZSC testing (below 0.3). Our ASTOP algorithm converges on a lower reward than SP in training (above 0.7) but does not drop to a lower average reward in cross-play ZSC testing. Again, ASTOP policies correctly optimize between equivalence classes to outperform SP in testing.

5.3 Policy Performance

To evaluate performance against SP, we take 30 policies of each algorithm type and run them against SP. Shown in Fig. 7 are the cross-play performances of each pairing. We can see that ASTOP outperform SP in the number of symmetry collisions. Our algorithm ASTOP chooses the maximum expected equivalence class rather than the optimal joint action to make safer decisions consistently across all individual policies without being given the symmetries ahead of time.

5.4 Complex Graphs

The graphs robots might encounter can be composed of various robots, locations, edges, and the values of the edges and locations. To estimate how well ASTOP

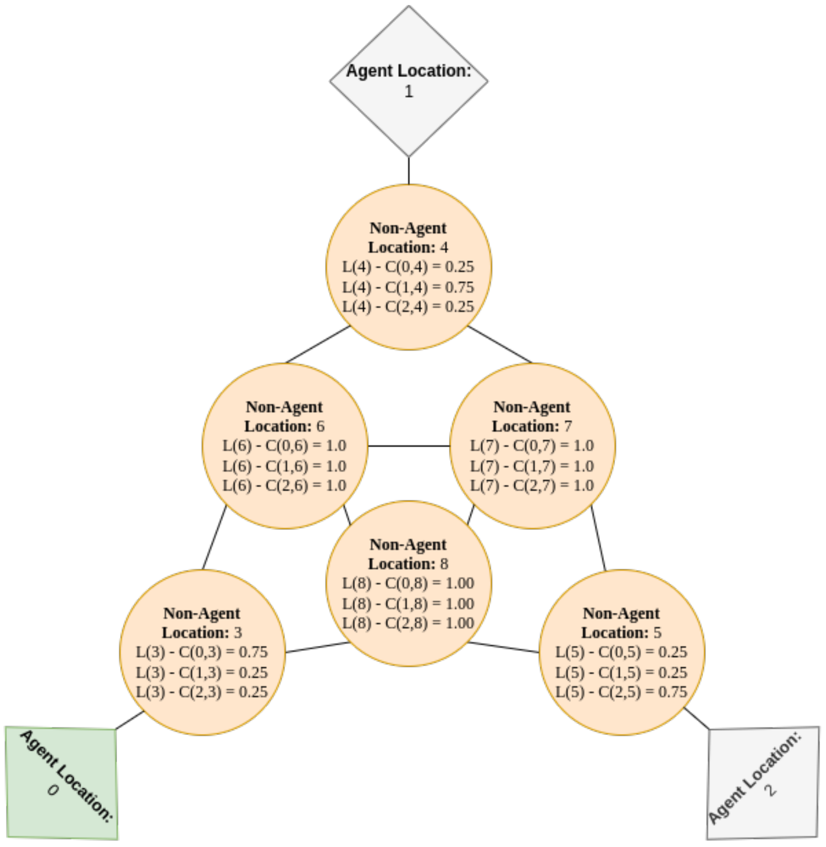

Fig. 5. Shown is a visual representation of a location-selection graph for three robots and six locations.

performs across a wide array of graphs, we run ASTOP across different combinations in Table 1. We ran 1200 randomly composed graphs with randomly generated location and edge values for each combination of robots and locations. Some random graphs may not contain symmetries; in such cases, ASTOP and SP would both find the optimal joint actions. The RANDOM policies randomly select a joint action to help compare ASTOP and SP to random behavior. We can see that both algorithms outperform random selection and that ASTOP outperforms SP in every case.

5.5 Quadrotor Experiments

We ran multiple physical robot experiments to demonstrate the applicability of the location-selection game and its relevance to robotics. We compare quadrotors

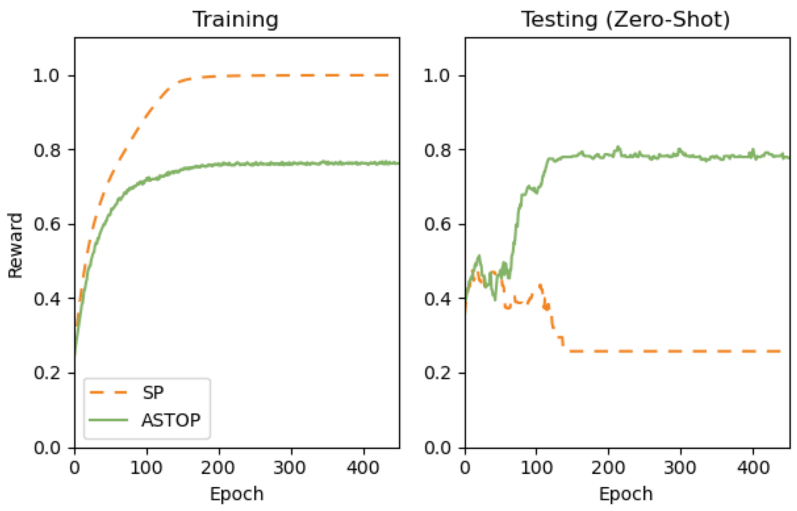

Fig. 6. Training and ZSC testing performance of SP and ASTOP algorithms on the location-selection graph in Fig. 5.

running SP or ASTOP trained policies for location-selection. The quadrotors fly at a preset height from their initial locations to their selected goal locations.

We use Crazyflie 2.0 nano-quadrotors to execute the flight trajectories in a space with virtual locations marked with numbered and colored circles for physical visualization. We use a 24-camera VICON motion capture system for localization. Every circular location has a diameter of 9in. The length of edges between locations is approximately 3.5 feet. The policies are uploaded to the quadrotors before takeoff and use the Crazyswarm infrastructure to execute the flight trajectories [28]. The control and state estimation runs onboard the quadrotors and the motion capture system broadcasts the localization. Snapshots of one flight execution are shown in Fig. 8. In Tables 2 and 3, the quadrotor cross-play tests obtain a higher reward for policy pairings with ASTOP policies.

6 Conclusions

Collaborative scenarios necessitate protecting policies against erratic behaviors and finding problematic symmetries. As learning methods for robotics continue to grow, it is vital to address these critical failure points. In future work, we investigate how to efficiently search for equivalences classes in larger, continuous, and more diverse action spaces. Here, we present a novel learning algorithm, Automatic Symmetry Tracking Other-Play (ASTOP), that builds on the OP algorithm to exploit the presence of unknown symmetries in the underlying problem. We discuss the performance of ASTOP in the zero-shot coordination problem

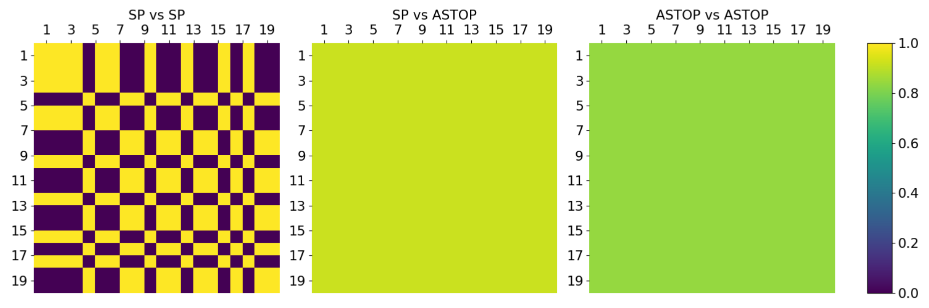

Fig. 7. Cross-play matrix visualization of paired policies trained under SP and ASTOP algorithms. The y-axis represents the index of robot one and the x-axis represents the index of robot two. Each block on the grid is obtained by evaluating the pair on the graph in Fig. 3.

Table 1. Algorithm Reward Over 1,200 Randomly Generated Graphs

(Number of Robots, Number of Locations)				
	(2,5)	(3,7)	(4,9)	(5,11)
SP	0.847	0.667	0.676	0.654
ASTOP	**0.906**	**0.902**	**0.915**	**0.934**
RANDOM	0.512	0.637	0.49	0.44
	(3,7)	(3,9)	(3,11)	(3,13)
SP	0.817	0.693	0.7	0.586
ASTOP	**0.861**	**0.889**	**0.921**	**0.885**
RANDOM	0.56	0.58	0.476	0.35
	(2,7)	(3,7)	(4,7)	(5,7)
SP	0.817	0.693	0.7	0.586
ASTOP	**0.861**	**0.889**	**0.921**	**0.885**
RANDOM	0.56	0.58	0.476	0.35

across graphs that vary in complexity and show multiple physical quadrotor experiments to demonstrate the applicability of the location-selection game and its relevance to robotics. Our ASTOP policies never arbitrarily break ties between symmetric trajectories and consistently outperform SP across all games played to improve collaboration and zero-shot coordination.

References

1. Akhloufi, M., Couturier, A., Castro, N.: Unmanned aerial vehicles for wildland fires: Sensing, perception, cooperation and assistance. Drones **5**, 15 (02 2021). https://doi.org/10.3390/drones5010015
2. Barrett, S., Stone, P.: Cooperating with unknown teammates in complex domains: A robot soccer case study of ad hoc teamwork. In: Proc. of the Twenty-Ninth AAAI Conf. on AI (January 2015)

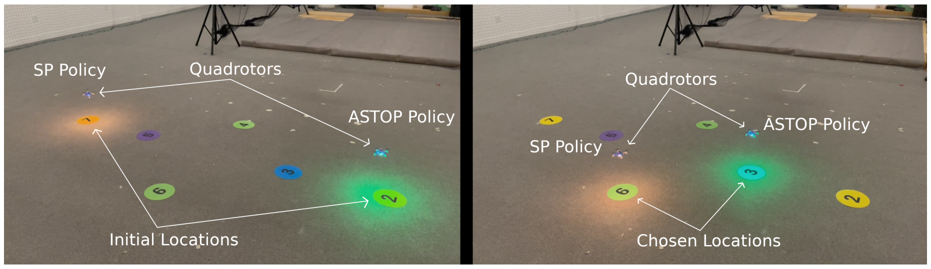

Fig. 8. Two Crazyflie 2.0 quadrotors execute their flight trajectories to reach their chosen locations. The orange quadrotor (left) used a self-play policy and the green quadrotor (right) used an ASTOP policy. The two do not collide as ASTOP chooses the safer location: number three.

Table 2. 2 Quadrotors Cross-Play Reward

Algorithm Pairing	Reward
(SP, SP)	0.52
(SP, ASTOP)	**0.92**
(ASTOP, ASTOP)	0.84

Table 3. 3 Quadrotors Cross-Play Reward

Algorithm Pairing	Reward
(SP, SP, SP)	0.438
(SP, SP, ASTOP)	0.636
(SP, ASTOP, ASTOP)	0.746
(ASTOP, ASTOP, ASTOP)	**0.758**

3. Barrett, S., Stone, P., Kraus, S.: Empirical evaluation of ad hoc teamwork in the pursuit domain. In: The 10th Int. Conf. on Autonomous Agents and Multiagent Systems - Vol. 2. p. 567–574. AAMAS 2011, Richland, SC (2011)
4. Boutilier, C.: Sequential optimality and coordination in multiagent systems. In: Proc. of the 16th Int. Joint Conf. on Artifical Intelligence - Vol. 1. p. 478–485. IJCAI'99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1999)
5. Brown, N., Sandholm, T.: Superhuman ai for heads-up no-limit poker: Libratus beats top professionals. Science **359**, 418–424 (2018)
6. Bullard, K., Kiela, D., Pineau, J., Foerster, J.N.: Quasi-equivalence discovery for zero-shot emergent communication. CoRR **abs/2103.08067** (2021), https://arxiv.org/abs/2103.08067
7. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Trans. on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **38**(2), 156–172 (2008)
8. Everett, M., Chen, Y.F., How, J.: Collision avoidance in pedestrian-rich environments with deep reinforcement learning. IEEE Access **PP**, 1–1 (01 2021). https://doi.org/10.1109/ACCESS.2021.3050338
9. Everett, M., Chen, Y.F., How, J.P.: Motion planning among dynamic, decision-making agents with deep reinforcement learning. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). Madrid, Spain (Sep 2018)
10. Everett, M., Chen, Y.F., How, J.P.: Collision avoidance in pedestrian-rich environments with deep reinforcement learning (2020)
11. Goldman, C.V., Allen, M.W., Zilberstein, S.: Learning to communicate in a decentralized environment. Autonomous Agents and Multi-Agent Systems **15**, 47–90 (2006)

12. Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. AI Review **55**(2), 895–943 (2022)
13. Harsanyi, J.C., Selten, R.: A general theory of equilibrium selection in games: A review article. The MIT Press, Cambridge, MA (1988)
14. Harsanyi, J.C.: The tracing procedure: A bayesian approach to defining a solution for n-person noncooperative games. Int. Journal of Game Theory **4**, 61–94 (1975)
15. Herings, P.J.J., Peeters, R.J.A.P.: Equilibrium selection in stochastic games. Int. Game Theory Review **05**(04), 307–326 (2003). https://doi.org/10.1142/S0219198903001082
16. Hu, H., Lerer, A., Peysakhovich, A., Foerster, J.N.: Other-play for zero-shot coordination. In: ICML (2020)
17. Isele, D., Rahimi, R., Cosgun, A., Subramanian, K., Fujimura, K.: Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In: 2018 IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 2034–2039 (2018). https://doi.org/10.1109/ICRA.2018.8461233
18. Islam, S., Razi, A.: A path planning algorithm for collective monitoring using autonomous drones. In: 2019 53rd Annual Conf. on Information Sciences and Systems (CISS). pp. 1–6 (2019). https://doi.org/10.1109/CISS.2019.8693023
19. Kanervisto, A., Scheller, C., Hautamäki, V.: Action space shaping in deep reinforcement learning. In: 2020 IEEE Conf. on Games (CoG). pp. 479–486 (2020). https://doi.org/10.1109/CoG47356.2020.9231687
20. Kleiman-Weiner, M., Ho, M., Austerweil, J., Littman, M., Tenenbaum, J.: Coordinate to cooperate or compete: Abstract goals and joint intentions in social interaction. In: COGSCI (01 2016)
21. Kochenderfer, M.J., Amato, C., Chowdhary, G., How, J.P., Reynolds, H.J.D., Thornton, J.R., Torres-Carrasquillo, P.A., Üre, N.K., Vian, J.: Decision Making Under Uncertainty: Theory and Application. The MIT Press, 1st edn. (2015)
22. Kumar, N., Fishman, M., Danas, N., Tellex, S., Littman, M., Konidaris, G.: Task scoping for efficient planning in open worlds (student abstract). Proc. of the AAAI Conf. on AI **34**(10), 13845–13846 (Apr 2020). https://doi.org/10.1609/aaai.v34i10.7195
23. Lerer, A., Peysakhovich, A.: Learning existing social conventions via observationally augmented self-play. In: Proc. of the 2019 AAAI/ACM Conf. on AI, Ethics, and Society. p. 107–114. AIES '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3306618.3314268
24. M. Campbell, A. J. Hoane, and F.-h. Hsu: Deep blue. AI **134**(1), 57–83 (2002). https://doi.org/https://doi.org/10.1016/S0004-3702(01)00129-1
25. Nash, J.: Non-cooperative games. Annals of Mathematics **54**(2), 286–295 (1951)
26. Oliehoek, F.A., Amato, C.: A Concise Introduction to Decentralized POMDPs. Springer Publishing Company, Incorporated, 1st edn. (2016)
27. Pham, H.X., La, H.M., Feil-Seifer, D., Deans, M.: A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking. In: 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). pp. 6648–6653 (2017). https://doi.org/10.1109/IROS.2017.8206579
28. Preiss, J.A., Hönig, W., Sukhatme, G.S., Ayanian, N.: Crazyswarm: A large nano-quadcopter swarm. In: IEEE Int. Conf. on Robotics and Automation (ICRA). pp. 3299–3304. IEEE (2017). https://doi.org/10.1109/ICRA.2017.7989376
29. Sadhu, V., Salles-Loustau, G., Pompili, D., Zonouz, S., Sritapan, V.: Argus: Smartphone-enabled human cooperation via multi-agent reinforcement learning for disaster situational awareness. In: 2016 IEEE Int. Conf. on Autonomic Computing (ICAC). pp. 251–256 (2016). https://doi.org/10.1109/ICAC.2016.43

30. Sallab, A.E., Abdou, M., Perot, E., Yogamani, S.: Deep reinforcement learning framework for autonomous driving. Electronic Imaging **2017**(19), 70–76 (2017). https://doi.org/doi:10.2352/ISSN.2470-1173.2017.19.AVM-023
31. Seraj, E., Gombolay, M.: Coordinated control of uavs for human-centered active sensing of wildfires. In: ACC (06 2020)
32. Shapley, L.S.: Stochastic games. Proc. of the National Academy of Sciences **39**(10), 1095–1100 (1953)
33. Shum, M., Kleiman-Weiner, M., Littman, M., Tenenbaum, J.: Theory of minds: Understanding behavior in groups through inverse planning. Proc. of the AAAI Conf. on AI **33**, 6163–6170 (07 2019). https://doi.org/10.1609/aaai.v33i01.33016163
34. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G., Graepel, T., Hassabis, D.: Mastering the game of go without human knowledge. Nature **550**, 354–359 (10 2017). https://doi.org/10.1038/nature24270
35. Stone, P., Kaminka, G.A., Kraus, S., Rosenschein, J.S.: Ad hoc autonomous agent teams: Collaboration without pre-coordination. In: AAAI (2010)
36. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)
37. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proc. of the tenth Int Conf. on Machine Learning. pp. 330–337 (1993)
38. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: 2017 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). pp. 23–30 (2017). https://doi.org/10.1109/IROS.2017.8202133
39. Tucker, M., Zhou, Y., Shah, J.: Adversarially guided self-play for adopting social conventions. ArXiv **abs/2001.05994** (2020)
40. You, C., Lu, J., Filev, D., Tsiotras, P.: Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. Robotics and Autonomous Systems **114**, 1–18 (2019). https://doi.org/https://doi.org/10.1016/j.robot.2019.01.003