Reward Shaping for Intelligent Swarm Coordination

Everardo Gonzalez

---

1. Survey prior work on reward shaping for multiagent systems, with a focus on problems with sparse and/or multiple rewards.
2. Survey prior work on the impact of learning on leader/follower systems for large multiagent systems and swarms.
3. Provide three research directions where reward shaping can be used in large leader/follower systems. At least one research direction should focus on algorithmic contributions and one on implementation/hardware contributions.
4. Discuss the broader societal (e.g. ethical, economic, policy, social, regulatory) implications of multi-robot systems operating under uncertainty in terms of future impact. Note that addressing this question requires including benefits, challenges, and downsides of the proposed work.

Contents

1 Introduction

Intelligent swarm systems offer advantages in warfare [116, 123], transportation systems [5, 29, 124], and addressing the climate crisis [49, 67], where large numbers of agents must coordinate to achieve complex coordination on tasks with non-trivial solutions. This is especially helpful when a system must perform multiple actions in multiple locations simultaneously. Swarm algorithms focus on getting intelligent swarm behaviors through hand-designed simple rules for local swarm interactions [83, 111], making it difficult and time-consuming to design algorithms to achieve a desired overall swarm behavior. Reward shaping in multiagent learning is focused on getting individual agents in a system to *learn* to achieve a system objective [3, 4], which starkly contrasts the approach of these hand-designed agent behaviors in swarms. Unfortunately, reward shaping methods do not scale well to coordinating large numbers of agents on complex tasks [107]. We survey literature from both of these areas in order to synthesize research directions that can leverage both the intelligence learned through reward shaping in multiagent learning as well as the large collective capabilities of a swarm. We conclude by discussing how these systems might impact warfare and surveillance, transportation systems, and the climate crisis.

2 Background

2.1 Intelligent Swarms

Swarms are composed of a large number of agents taking independent actions based on local observations and simple rules [83, 111]. The intelligence in these systems emerges from many local interactions, but we consider cases where the swarm has an explicit system objective that must be achieved. We refer to these systems as "intelligent swarms" because we are going beyond defining simple rules that result in emergent intelligence, and instead giving the swarm an explicit objective.

Emergent intelligence is advantageous in its robustness to individual agents failing [45], and previous studies have found local rules that result in effective swarm behaviors for various domains [15, 83, 137]. However, this approach requires meticulously designing and testing hand-coded rules to achieve the desired behavior. To best leverage the swarm's collective capabilities, a subset of the swarm can interact based on simple rules described by prior work, and another subset of the swarm can learn how to influence the rest of the swarm to collectively achieve complex tasks. This gives us the benefit of a large collective without the necessity of hand-coding every agent's behavior.

2.2 Multiagent Learning

In multiagent learning, multiple agents operate in a shared environment with the goal of optimizing system feedback [17–19, 23, 76, 102]. To motivate the learning methods we explore, consider an underwater observation task where multiple robot fish must coordinate to collect observations on scientific points of interest (POIs): coral and seaweed. This task is illustrated in Figure 1. Each robot fish receives its own local observation of the environment, maps that to an action using its individual policy, and takes that action in the environment. We can consider the collective state of all the agents in the system as a "joint-state" or "system state", and collective actions as a "joint-action". Each agent receives the system feedback, or system "reward", as feedback.

Multiagent learning can be formalized as learning a joint-policy for a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [73, 89, 120], as opposed to single agent learning which is usually formalized as learning a policy for a Markov Decision Process (MDP) [39, 65, 140]. A Markov Decision Process is a 4-tuple composed of states, actions, transition probabilities, and

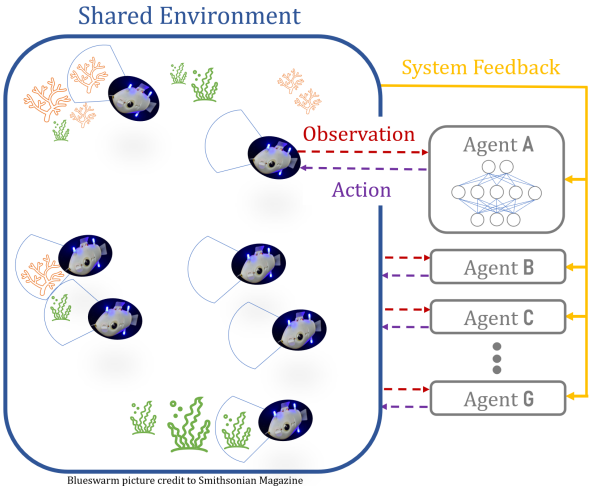

Figure 1: Multiple robot fish must coordinate to collect observations on coral and seaweed. Aspects relevant to multiagent learning are explicitly called out.

rewards. Each state is Markov, meaning that the next state is only dependent on the current state, rather than a history of states. In a Dec-POMDP, the underlying system state is Markov, but it is decentralized among several agents where each one can only observe a portion of the system state, and must act independently from other agents. This makes it so that many concepts from single agent learning partially transfer into multiagent learning, but not entirely.

There are distinct challenges that arise in multiagent learning. Primarily, there are the challenges of other agents' learning exacerbating environment stochasticity [57–59], agent credit assignment [55, 106, 119, 145, 150], reward sparsity [42, 74, 107], and balancing multiple reward signals [68, 101]. From the perspective of a single agent, the exploratory actions of other agents cause unpredictable changes in the environment. Additionally, each agent only observes a portion of the system state, making it difficult to map an observation to the best action to take as one observation can represent various distinct states. The agent credit assignment problem is concerned with how to distill system feedback into feedback for individual agents based on an individual agent's contribution to the overall system's performance. Since the system feedback captures the performance of all agents in the system, this signal is quite noisy in capturing the contribution of any particular agent – in fact, most of this feedback is capturing the performance of the *other* agents.

Reward sparsity arises from tasks that require tight coordination [107], where agents must discover a sequence of joint-actions together before receiving any feedback. Rewards may also be inherently sparse based on the system objective [142, 143]. In the underwater observation task, the objective could instead be to find a stingray, in which case the system would only receive positive feedback once a stingray was found, and no feedback otherwise, meaning that most actions taken by the robots would have no associated instantaneous reward. A multiagent system may also have multiple objectives, meaning that the system must learn to make tradeoffs between multiple reward signals on top of associating actions with these different signals. In the underwater observation task, these reward signals could each give information about different POIs, power efficiency, swimming speed, and so forth depending on what was important to the system.

Different learning algorithms can be used to learn a joint-policy. Multiagent Reinforcement Learning (MARL) learns to associate observation-action pairs with expected values, and uses this to learn a joint-policy [6, 53, 72, 73, 98, 119, 146]. Cooperative Coevolutionary Algorithms (CCEAs) aggregate the system rewards into a system fitness or system "evaluation", and evolve each agent's

policy alongside other agents' policies in order to learn a joint-policy [4, 48, 69, 110, 148].

Some multiagent systems are competitive, in that different agents seek to maximize an individual reward signal and must learn to compromise with other agents [76, 121]. Learning in these systems focuses on finding a Nash Equilibria, which is an outcome in which no agent benefits from changing its individual policy, rather than focusing on maximizing a system reward signal. We will focus primarily on cooperative multiagent systems as they are the most relevant to our work.

2.3 Multireward Learning

A swarm might have objectives that are not well captured by a single reward signal, especially if these systems are being used in long term deployments. Decomposing the reward signal into multiple rewards necessitates the use of multi-objective optimization methods [68, 127]. These methods focus on learning a Pareto front, which in this case is a set of policies that represent the best tradeoffs between multiple objectives [85, 86, 144]. A policy is on the Pareto front if a change to that policy would make it less optimal with respect to at least one objective. Consider the underwater observation task in Figure 1. If these robot fish are deployed with a single reward signal, then we must assign a value to observing coral and a value to observing seaweed in order to combine feedback for both types of observations into a single reward signal, inherently baking a tradeoff directly into our reward signal. In a long term deployment, the mission objective may change, and a virus killing off coral reefs could make coral observations far more valuable than seaweed observations. If we had taken a multiobjective approach, then we could simply change where along the Pareto front we are selecting policies. In the single reward case, we would have to retrain our system entirely with a new reward function that has the new tradeoff baked in.

2.4 Counterfactuals

Counterfactuals are used quite heavily in reward shaping in multiagent systems. A counterfactual is simply something that is counter to the facts [47, 71]. These are useful in multiagent systems for computing counterfactual system feedback to "what if" scenarios [3, 4]. Consider an example from the underwater observation task. An agent may have taken a sequence of actions to observe a seaweed POI, but this may not have been the most useful for the system objective. We can rerun the simulation with a counterfactual agent that swims to the nearest coral POI in place of this agent, and receive counterfactual system feedback. By comparing the actual feedback with the counterfactual feedback, we can determine whether swimming to the nearest coral would have actually been more helpful for the overall system. The use of counterfactuals improves the performance and adaptivity of learned joint-policies when leveraged appropriately [3, 4, 119, 148].

3 Reward Shaping For Multiagent Systems

Reward shaping in a multiagent learning context focuses on methods that take the system reward signal and modify it for individual agents, either adding information or removing noise in order to make this reward signal better suited for learning without modifying the target task [3, 4, 7, 31, 33, 85]. Reward shaping is related to, but distinct from reward design and reward approximation. Reward design, sometimes also called "reward shaping" [40, 66], focuses on how to turn the requirements for a target task into a reward signal that makes this task learnable. Reward approximation focuses on how to approximate the reward signal to get a denser reward signal or to speed up the reward calculation via the approximation.

While most reward shaping methods are designed to work specifically with system rewards, some shaping methods have equivalent fitness shaping techniques that extend them to cooperative coevolutionary algorithms [4, 25]. These techniques generally help agents learn better policies, but are especially useful when agents learn with sparse rewards, or multiple rewards. The pains from reward sparsity can be partially alleviated through reward shaping methods that add additional information to the reward signal. Similarly, the trade-offs between multiple reward signals can be made more apparent when those reward signals have been cleaned of noise.

Shaped rewards take the form of the following equation, where G, our system feedback, is modified with the addition of a shaping term F in order to calculate the shaped reward.

$$r_{shaped} = G(s, a, s') + F(s, s') \tag{1}$$

Equation 1 is specifically focused on shaping rewards, so G and F are functions of states and actions. Similarly, we could think of G and F in terms of sequences of states and actions for evolutionary algorithms. We explore different reward shaping techniques, which focus on adding information, removing noise, or a combination of both in order to create a better learning signal.

3.1 Adding Information

Adding information through reward shaping is particularly well suited to address reward sparsity in multiagent systems [32, 112]. By adding information through the shaping term, we can drastically reduce the time agents in the system spend trying random actions with no feedback, and instead provide agents with "stepping stone" rewards that eventually lead to the system reward.

When adding information to an agent's reward signal through reward shaping, it must be done with caution so as not to modify the actual task as represented by the original reward function. Unchecked reward shaping could lead to a sort of malicious compliance from a learning agent [108]. Consider if we wanted to add a shaping term to the underwater observation task rewarding agents for staying near one another to better meet coupling requirements for POIs. Agents could now exploit that shaped reward and learn to just find other agents rather than going to POIs.

3.1.1 Potential Based Reward Shaping Framework

Potential Based Reward Shaping (PBRS) addresses the malicious compliance pitfall [7, 31], and acts as a framework for many of the following methods that add information to the reward signal. This framework allows the injection of information into an agent's reward while ensuring that the resulting shaped reward is policy invariant. Policy invariance means that the optimal policy for the shaped reward function will also be the optimal policy for the original reward function. The accompanying theory shows that policy invariance is guaranteed for a single agent MDP, and that the Nash Equilibria does not change for competitive multiagent MDPs [7]. However, further theory shows that a multiagent system can converge on a different joint-policy with PBRS, as the additional reward alters agents' explorations, which affect the experiences of other agents [33]. In the multi-objective case, the theory shows that PBRS does not alter the true Pareto front [85]. Despite the lack of a guarantee for joint-policy invariance, plenty of experiments empirically demonstrate the efficacy of the PBRS framework applied to multiagent learning [16, 32, 35, 50, 88, 101, 113].

Potential based reward shaping requires a potential function that calculates the potential values of different system states or agent observations. This function can be expressed as $\Phi(s)$ if the potential function is static [7], and $\Phi(s, t)$ if the potential function is dynamic. This potential function can be engineered using knowledge about the domain and target task [113, 141], generated through counterfactuals [113], or generated from solutions gained from conventional planners [32,

38, 50, 88]. While the potential function can essentially be arbitrary, the design of the potential function does influence what policy or joint-policy is actually learned [16], even if it doesn't change the optimal policy [7], so this function must be designed with care.

The actual shaping function is defined as the difference of potentials between a new state and old state. Following the example of a shaped term for keeping underwater agents near one another, this approach would prevent agents from exploiting those proximity rewards. If an agent got close to a second agent, each one would gain a positive reward from F, but if they remained close to one another, they would receive no additional reward from F, as there would be no difference in potentials. Potential Based Reward Shaping is defined mathematically through the following equations, with Equation 2 representing Static PBRS, and Equation 3 representing Dynamic PBRS.

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \tag{2}$$

$$F(s, t, s', t') = \gamma\Phi(s', t') - \Phi(s, t) \tag{3}$$

$F(s, s')$ and $F(s, t, s', t')$ are the shaping terms, γ is a discount factor, $\Phi(s')$ and $\Phi(s', t')$ are the new state potentials, and $\Phi(s)$ and $\Phi(s, t)$ are the old state potentials. Notably, the rewards collected through F from reaching a terminal state in the MDP must sum to zero to maintain the guarantees from PBRS [51].

3.1.2 Knowledge Based Reward Shaping Methods

Knowledge based reward shaping seeks to encourage useful behaviors during the learning process by incorporating rewards for these behaviors into the shaping term. Proximity rewards for our agents to stick together for more efficient observations are a good example of this type of shaping.

Many works experiment with incorporating knowledge based hueristics into the potential function for PBRS [34, 37, 101, 112, 113, 141]. When used effectively, PBRS with a knowledge based potential function can lead to learning a better joint-policy than using just the system feedback G. However, PBRS must be used with caution, as while the multiagent system learns a joint-policy from the same set of joint-policies that would be learned otherwise through G, that joint-policy will not actually be better if the potential function is not well designed [34, 101, 112]. Notably, knowledge based shaping can be done outside of the PBRS framework and still improve performance on the original task [138]. This is an effective method for reward shaping when we have access to useful heuristics for solving the target task, both for providing additional signals with reward sparsity, and making multiple reward signals more informative in multi-reward learning.

3.1.3 Plan Based Reward Shaping Methods

Plan Based Reward Shaping uses conventional planners to generate potential functions for use in the Potential Based Reward Shaping Framework [32, 38, 50, 88], with some works incorporating these planners into multiagent reward shaping [32, 88]. In our underwater observation example, we might plan a path from each agent to a POI, and use this plan to inform the shaping term used in the shaped reward, encouraging our agents to follow the planned paths. STRIPS [41] is a popular planner for these methods [32, 38, 50], but requires a well-defined goal state to be applied effectively [50]. Alternatively, an abstract MDP can be learned from the original MDP and solved as a means of generating a plan based potential function [88]. Using plan based reward shaping for F directly, rather than in a PBRS framework, can also lead to better coordination [74].

This approach is effective in addressing reward sparsity if we are able to generate a plan, but this comes with caveats. STRIPS creates local optima when there many valid plans [50], and

the solution to an abstract MDP is not guaranteed to solve the original MDP [88]. Additionally, these planners don't consider multiple entities, causing conflicts in individual plans [32]. While this approach has been shown to help in single-reward settings, there is little to no work exploring this in multi-reward settings, where the target task might be more complex than hitting a particular goal state. Perhaps these approaches could be further explored in a multiagent setting with the help of multi-robot path planners [54, 56, 70] and multi-objective multi-robot path planners [82, 135].

3.1.4 Counterfactual as Potential

We can leverage counterfactuals to compute a potential function for multiagent learning. In Counterfactual as Potential (CaP) [113], the potential function is defined as shown in Equation 4.

$$\Phi(s) = G(s_{-i}) \tag{4}$$

$\Phi(s)$ is the potential function for agent i, and $G(s_{-i})$ is a system reward with agent i counterfactually removed from the system. This encourages agents to seek out tasks that are not already being done by other agents, as pushing another agent away from a task it was already doing would reduce $G(s_{-i})$. In the underwater observation example, this would encourage agents to seek out POIs that have not already been observed by other agents.

CaP can be approximated to alleviate the potentially high computational cost of calls to G, and so agents do not require access the functional form of G [84]. In sparse reward settings, CaP will not help because it adds information based on a slightly cleaned G, and if the original G does not contain any useful information, then the counterfactual G will not either. CaP has not been tested in multi-reward settings, but would likely benefit learning similarly to other methods that add information to the system reward so long as rewards are not sparse [85, 86].

3.2 Removing Noise

Noise in the system reward for multiagent learning comes primarily from G capturing feedback for the entire system. From a single agent's perspective, G is mostly noise from other agents operating in the environment. By cleaning this signal, we can better isolate an individual agent's contribution.

3.2.1 Difference Rewards

A difference reward is specifically designed to have a strong alignment with G and have a high sensitivity to an individual agent's actions [3–5, 20, 26, 113, 139, 141, 144]. By comparing the actual system reward with a counterfactual system reward where that agent is removed from the system, we can calculate a difference reward that captures that individual agent's contribution to the system reward. In the underwater observation example, we could remove an agent entirely from the system to compute a counterfactual reward that would help us determine whether that agent's current observation is helping the system. This is formalized in Equation 5.

$$D_i = G(z) - G(z_{-i} \cup c_i) \tag{5}$$

D_i represents the difference reward for agent i. z represents either a joint-action or sequence of joint-actions, depending on whether this is used for MARL or a CCEA. $G(z)$ represents the original system reward, and $G(z_{-i} \cup c_i)$ represents the system reward where agent i took counterfactual action(s) c_i. While this approach does not address reward sparsity, it does help clean reward signals in a multi-reward setting [101, 144]. Additionally D can be approximated using information local

to each agent [104, 105, 124], and difference rewards have been shown to assist multiagent learning in a variety of domains [4, 5], offering a popular solution to the credit assignment problem.

3.2.2 CLEAN Rewards

Coordinated Learning without Exploratory Action Noise (CLEAN) rewards drastically reduce the noise in G [57–59]. With CLEAN, agents always take their best action in the shared environment. To estimate the value of an exploratory action, an agent computes a counterfactual system reward where that agent took the exploratory action, and compares this reward to the actual system reward where the agent behaved according to its best policy. In the underwater observation example, this would involve each agent having a full copy of the environment and other agents' best policies that the agent can take exploratory actions in. CLEAN is formalized in Equation 6.

$$C_i(a) = G(a_{a_i \leftarrow a_i'}) - G(a) \tag{6}$$

$C_i(a)$ represents the CLEAN reward, $G(a_{a_i \leftarrow a_i'})$ represents the system reward from the exploratory action in the copied environment, and $G(a)$ represents the actual system reward with the agent's actual action. CLEAN rewards have been shown to outperform difference rewards in learning high performing joint-policies [58], but have yet to be tested in multireward settings. As CLEAN is a noise reduction method, it would help a system clean multiple reward signals, but not with learning from sparse rewards. The main limitation is that there is no approximation for CLEAN, so it requires many calls to G and access to the functional form of G for each agent.

3.3 Combination Methods

3.3.1 D++: An Extension to Difference Rewards

D++ extends difference rewards to tightly coupled tasks, where multiple agents are required to take a joint-action together in order to receive a reward [107]. Feedback for these tasks is inherently sparse as agents are unlikely to stumble upon the necessary joint-action to receive a reward. In the underwater observation example, coral observations might have a tight coupling requirement of 3, indicating 3 agents must observe a coral POI simultaneously in order to receive a system reward for the observations. In this case, we would want to encourage individual agents to observe the coral to maximize the chance that 3 agents hit the coupling requirement, rather than hoping that 3 agents will randomly stumble upon the correct joint-action simultaneously to observe the coral. D++ gives us the stepping stone reward necessary to do this.

To calculate a D++ stepping stone reward for an agent, we consider a counterfactual system reward where this agent has the "weight" of n additional agents and compare this to the actual system reward to determine if this agent's behavior would be beneficial with the help of other agents. This is normalized by n to ensure this reward is a "stepping stone", and doesn't accidentally distract agents from actually meeting the coupling requirement. This is formalized in Equation 7.

$$D^n_{++}(i) = \frac{G(z_{+(\cup_{i=1,\ldots,n})i}) - G(z)}{n} \tag{7}$$

This is calculated iteratively across a range of n additional agents to give an agent the maximum stepping stone reward possible. $D^n_{++}(i)$ represents the stepping stone reward, $G(z_{+(\cup_{i=1,\ldots,n})i}$ represents the system reward with the counterfactual agent(s), and $G(z)$ represents the actual system reward. The primary use case of D++ is in tackling tightly coupled tasks, but if an environment does not have tightly coupled tasks, then D++ will not help. This means D++ is not automatically useful for learning with sparse rewards or multiple rewards.

3.3.2 Difference Rewards incorporating Potential-Based Reward Shaping

Difference Rewards incorporating Potential-Based Reward Shaping (DRiP) gives an agent a difference reward plus a knowledge based potential reward [113]. In the underwater observation example, this could mean calculating D for an agent's observations by removing that agent, and then adding a knowledge based potential reward based on spreading out to cover the most POIs. DRiP is formalized in Equation 8.

$$r_{shaped} = D(s, a, s') + \gamma\Phi(s') - \Phi(s) \tag{8}$$

All terms are the same as defined for difference rewards and PBRS. This method has been shown to outperform both D and PBRS on their own in a multiagent setting [113], so it suffices to say this approach would likely perform quite well in addressing sparse rewards (which PBRS addresses well) and cleaning multiple noisy rewards (which D addresses well), but there is little or no follow-up on this method beyond the original paper.

4 Impact of Learning on Leader-Follower Swarms

The swarm shepherding problem involves leaders that must guide a swarm of followers, and followers that interact with each other based on simple local interactions. Many works in swarm shepherding focus on a subset of this problem: learning in small leader-follower systems. We focus on the swarm shepherding problem and how learning has been leveraged to solve this problem. The intersection of learning, leader-follower systems, and swarms has incredible potential [99, 129] and offers many interesting research directions that remain unexplored. A few works at this intersection focus on expanding learning to large leader-follower systems, but many of the small scale experiments are meant to demonstrate a proof of concept for swarm shepherding in larger systems. We explore both learning-based and conventional methods for swarm shepherding to better understand the impact learning has had on solving this problem, as well as hardware implementations and hardware-focused work as relevant to generating new research directions.

4.1 The Swarm Shepherding Problem

Swarm shepherding is a specific problem where a swarm is split into two agents types: sheepdog agents and sheep agents [80], which we consider as leaders and followers, respectively. The followers follow a prescribed behavior based on simple rules to guide interactions with their neighbors. Leaders must guide the followers from a starting location to a goal location, and do this through local interactions with the followers, oftentimes repulsing followers to push them to the goal.

This problem in the Artificial Intelligence (AI) and Robotics literature takes inspiration from the real life shepherding problem of instructing a sheepdog or team of sheepdogs to guide sheep to a goal location. In the context of AI, this problem takes the form of simplified sheepdog agents guiding sheep agents in order to explore and move around in an environment to accomplish spatial tasks. Many works in the AI and Robotics literature focus on this problem in the context of a small "swarm", looking at sub-problems as small as one leader and one follower [117], but the idea is to generate approaches that generalize to larger swarms [21, 129]. The objective is oftentimes to get as many sheep agents to the goal location as possible, minimize the total distance from each sheep agent to the goal location, or maximize the success rate of bringing all the sheep agents into a goal location given some noise in the system. The solution to the problem is a control policy for the sheepdog agents, which takes as an input either global or local information about the flock, and outputs an action for the sheepdog agent to take towards guiding the sheep.

These algorithms have been primarily tested in 2D particle environments. In these environments, each agent has an (x,y) position, and moves according to a (dx,dy) at each timestep. In particle domain experiments, sheep have policies based on two behaviors: attraction and repulsion. If other sheep or shepherd agents are too close, then a sheep agent will be repulsed by those agents. Otherwise the sheep is attracted to either agents within an observable radius or the n nearest sheep agents in the flock. A repulsion vector and attraction vector are summed together to determine a sheep agent's action. The shepherds usually have access to precise global information about the flock and the goal location, though there are studies that explore this problem with shepherds using local rather than global flock information. Most hardware tests are performed with two-wheeled differential-drive mobile robots, and the algorithms tested on hardware are simpler than algorithms that have been tested in these simulated particle environments.

Many early approaches to the swarm shepherding problem rely on hand coding different behaviors for the sheepdog agents and specifying conditions for when to switch between these different behaviors [80]. These approaches have been quite popular from the late 1990's through the late 2010's [77, 78, 131, 132]. However, more studies in swarm shepherding algorithms started using learning based approaches in the late 2010's and early 2020's [1, 63, 149]. These approaches experiment with many learning algorithms, from Q-learning and deep reinforcement learning to evolutionary algorithms and differential evolution, as well as different learning methodologies such as curriculum learning and inverse learning. We explore how early works attempted to hand-code good shepherding policies for shepherd agents, how learning based approaches expanded the capabilities of these shepherd agents, and different attempts at solving this problem in hardware. Throughout this review, we address the limitations of the different approaches.

4.2 Hand-Coded Sheepdog Policies

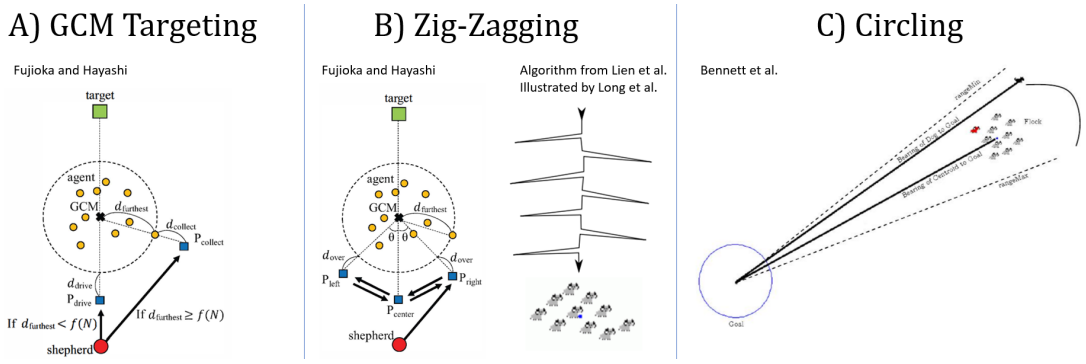

Figure 2: Different approaches to swarm shepherding, with credit to authors for illustrations.

Many conventional approaches to the swarm shepherding problem take direct inspiration from the real-life problem of sheepdogs guiding a flock of sheep to a goal location. Humans have been using sheepdogs for hundreds of years, and have documented what behaviors are useful for assessing the effectiveness of a sheepdog in shepherding [125], making this an intuitive starting point for generating good leader policies. These behaviors have been developed over the course of many generations of people documenting what has worked effectively for shepherding with sheepdogs, making this quite a rich starting point for solving the more general swarm shepherding problem.

There are three types of approaches to shepherding based on hand-coded control policies, as illustrated in Figure 2. The sheepdog agent (or agents) first positions itself behind both the sheep agents' global center of mass (GCM) and goal location.

The first type of approach then has the sheepdogs move directly towards the GCM in order to push the sheep towards the goal location [44, 92, 132]. These approaches come with the limitation that if the flock does not have strong attraction vectors between the sheep agents, then the flock disperses when the sheepdogs approach [14]. In some of these approaches, the sheepdog policy includes a mechanism for switching from *driving* the flock to *gathering* individual sheep agents if they do begin to disperse [44, 92]. The second type of approach addresses the problem of poor flock cohesion by pushing the sheep agents forward by zig-zagging side to side behind the flock or in a v-shape partially encircling the flock [43, 44, 78]. This helps to keep the flock together by pushing any sheep agents that start to stray from the flock back towards the center of the flock. However, these approaches can still fail to maintain flock cohesion depending on how far the sheepdogs move side to side and how far the v-shape extends around the flock. The third type of approach is for the sheepdogs to alternate between clockwise and counter-clockwise circles around the flock while simultaneously drifting towards a goal location [14]. This better addresses flock cohesion by ensuring coverage of the entire flock, but does require an assumption of how big those circles should be.

The major limitation of these conventional approaches is they all require quite specific and extensive domain knowledge in actually designing a control policy for the shepherd agent, requiring extensive parameter tuning to work properly [36, 61, 62]. They are also quite slow to develop and improve as there are years between publications that tweak previous approaches in order to maintain better flock cohesion, speed up shepherding, or achieve higher success rates in shepherding. Additionally, approaches need to be tweaked in order to accommodate different numbers of sheepdogs [77, 78]. The creativity and complexity of these algorithms is quite limited because even the simpler implementations require extensive tuning to work properly, and any complexity must be explicitly known and encoded into the sheepdog agents' behaviors by a person. As we will see with learning based methods, neural networks do not have this limitation and can capture plenty of complexity that can even solve more challenging variations of the swarm shepherding problem that these hand-coded approaches cannot.

4.3 Learning Sheepdog Policies

There are three main approaches to solving this problem with learning. There is an important distinction that needs to be made here that none of these approaches are multiagent learning, with the exception of one paper which we discuss separately. These approaches use a single learner that learns how to map a single sheepdog observation to a single sheepdog action. This learner learns one policy that is copied and tested across all of the sheepdog agents. Different approaches experiment with Q-learning, deep reinforcement learning, and evolutionary algorithms, but the real delineating factor between the approaches is the methodology behind what the algorithm is learning and how the learning is structured.

4.3.1 Machine Teaching

The first approach stems from Abbass's work in "machine teaching" [2], which is curriculum learning with elaborate and intentional curriculum design. The idea is to break down the overall task of shepherding into sub-tasks and skills [24, 46, 63]. Instead of learning the full control policy at once, the learner learns how to accomplish sub-tasks that are designed to hone certain skills associated with good shepherding, such as *driving* and *gathering* as mentioned earlier. This involves the learner learning to associate observations with the best actions by interactions between the learner and the environment. Whereas hand-coded sheepdog policies require adjustments to be made in

scaling up the policies to multiple sheepdogs, this approach has been demonstrated to be successful in learning a good sheepdog policy for problems involving multiple sheepdog agents without needing adjustments based on the number of sheepdogs [63]. However, there is a fundamental limitation to this: while this machine teaching approach does make it possible to learn a control policy, this approach does require curriculum design that inherently encodes biases about how the learner should solve the problem, potentially limiting the creativity of the learned solution. This is more like learning to do what we've already been doing more efficiently, rather than trying to learn a fundamentally better control policy.

4.3.2 Apprenticeship Bootstrapping

The second approach builds on machine teaching, but instead of the learner learning through interactions with the environment, this approach uses "Apprenticeship Bootstrapping" (AB) [95, 97]. AB is quite similar to inverse reinforcement learning [10], where the learner learns to mimic an existing control policy rather than generating one from scratch. This approach still breaks down the overall task into sub-tasks and skills, but a person then controls a sheepdog agent in the 2D particle environment to accomplish these different sub-tasks [96]. This is used to generate training examples for the learner to learn from in machine teaching. Rather than learning each sub-task from scratch, the learner learns to mimic the control policy of a person solving the sub-task. While this approach does speed up learning by not requiring random exploration from the learner in the environment, this approach does further constrain the creativity of the learner by requiring that the learner mimic an underlying control policy generated by a human teacher.

4.3.3 No Machine Teaching or Apprenticeship Bootstrapping

The third approach does not use any kind of curriculum or human generated examples. This is learning in its simplest form where the learner is simply released upon the target task and must learn to solve it through trial and error [13, 30, 129, 149]. Studies using this approach are just as successful as the previous more complex approaches, but have only been explored more recently. This is likely because Abbass introduced machine teaching [2], and went on to publish seminal work in learning-based swarm shepherding [24, 46, 63], creating a strong bias towards machine teaching for these early learning methods. The learning methods which do not require a curriculum or human examples have been shown to learn better control policies than hand-coded methods in the presence of increased noise in agent actions [30], and have even been able to solve variations of the swarm shepherding problem previous approaches have not explored. Zhi et al. recently published work using a deep reinforcement learning approach to learning how to guide the flock around complex obstacles that required nontrivial navigation [149]. The big advantage of this approach is that it does not require any kind of domain expertise or curriculum design in order to be successful, and it actually calls into question whether the other approaches are necessary in the first place. Perhaps a synthesis of these different methods would enable sheepdog agents to perform even more complex variations of the swarm shepherding task, but this has yet to be seen.

4.3.4 Multiagent Learning

Looking at swarm shepherding through the lens of multiagent learning, as far as we are aware, has only been addressed in Nguyen's work [99], which provides a rich gap for further exploration with multiagent learning based methods. Nguyen et al. define individual reward functions for five shepherd agents so each agent learns a unique policy. These functions are designed depending on assigned positional roles. Each shepherd is responsible for covering a specific portion of the flock:

upper-left, lower-left, lower-middle, lower-right, and upper-right. This creates a v-shape around the flock, similar to earlier methods that trace a v behind the flock, but this covers several portions of the v at once. Particle swarm optimization is used to solve for the weights of the neural networks as opposed to an evolutionary or reinforcement learning algorithm, and each network only has a single hidden layer. There is plenty of room to expand this work. This does not leverage the advances in the multiagent learning literature that focus on aligning individual reward functions with the system reward, like difference rewards. Additionally, this method is limited to five roles, and the authors claim this could be expanded to larger swarms by assigning multiple sheepdogs to each role, but this would not be as effective as allowing further specialization amongst sheepdogs. While this does demonstrate that multiagent learning has a place in the swarm shepherding problem, this work stops short of showing how multiagent learning might be leveraged in order to tackle more challenging variations of the swarm shepherding problem.

4.4 Hardware Implementations

While many approaches have been validated in simulation with simple particle agents, few have been validated in hardware. Hardware implementations of swarm shepherding are tested on simple versions of the swarm shepherding problem, and they use far simpler algorithms than the ones tested in the particle domain [21, 103, 117, 132]. One study from 1999 by Shultz et al. incorporates learning into their hardware implementation [117]. This study only used one sheepdog robot and one sheep robot. While this is excellent pioneering work, we now have far more sophisticated learning algorithms, swarm shepherding algorithms, and robotics hardware. A similar study conducted with modern hardware and learning algorithms would address a large gap in research literature, and may even be successful with far larger numbers of sheepdog and sheep robots.

A notable hardware study by Çelikkanat et al. [21] used 7 kobots (two-wheeled differential-drive mobile robots) for a leader-follower task, with up to 100 in simulation. Each kobot has 8 IR sensors placed radially around the base, and a magnetic compass sensing heading. Each kobot (both leaders and followers) repulsed the nearest kobots, aligned heading with nearby kobots, and was attracted to far away kobots. These repulsion, alignment, and attraction forces were summed to obtain the high-level control command for each kobot. However, leader kobots had an additional force pulling them towards the goal location, and Çelikkanat et al. showed that only a minority of informed leaders can guide the entire swarm to a goal location, which is consistent with similar studies [129]. Their unique contribution was in showing how their algorithm would be deployed on hardware. Note that the algorithms used by Çelikkanat et al. were quite simple compared to what we discussed above, so there is plenty of room here to test whether the more complex shepherding algorithms would work on hardware, or what gaps would need to be addressed to scale them up to hardware tests.

Several studies look into tackling more hardware specific problems in developing these algorithms, but stop short of hardware testing. This includes designing algorithms that can work with limited local information as opposed to global swarm information [100], that use computer vision for sheepdog observations [109, 128], and that are implemented on simulated UAVs (Unpersoned Aerial Vehicles) guiding UGVs (Unpersoned Ground Vehicles) [22, 79, 81, 96]. There is plenty of potential here for addressing gaps for hardware implementations, especially in using local limited information or computer vision for sheepdog observations, as oftentimes hardware implementations involve working with limited information based on sensor readings, and computer vision for high-level perception information. The gap here is testing these approaches on hardware to analyze the robustness and pitfalls of these approaches in the real world.

5 Synthesis/Research Directions

All of these research directions apply reward shaping to a multiagent learning approach to the swarm shepherding problem, where each sheepdog, or "leader", learns its own unique control policy. Direction 1 focuses on credit assignment, direction 2 focuses on hardware analysis, and direction 3 focuses on learning with local information.

5.1 Difference Rewards For Multiagent Swarm Leaders

Neither reward shaping alone nor swarm shepherding can coordinate a swarm to achieve a complex objective that requires tight coordination on multiple tasks simultaneously. A purely multiagent learning approach, even with shaped rewards, fails to scale up to the large number of agents in a swarm. On the other hand, a swarm shepherding approach only works to guide the entire swarm to achieve one task, and does not have any way of splitting up the swarm to tackle multiple tasks.

This research direction investigates how to integrate difference rewards into each leader's individual reward signal. The challenge here is to define a suitable counterfactual for the difference reward to not only capture the leader's direct contribution to the system (which is what the standard counterfactual does), but also capture how that leader influenced followers' contributions to the system. This will make it possible for leaders to learn to specialize in different aspects of the swarm's objective, making it possible to intelligently split the swarm to investigate various points of interest simultaneously.

5.2 Hardware Analysis of Complex Shepherding Algorithms

There is a large gap in the swarm shepherding literature in testing the more complex shepherding algorithms on hardware, bringing into question how effective these algorithms would be on real-world robots. Many algorithms do not consider the complexities of robot dynamics in agents' motion models nor the noise from the inherent uncertainty in real-world sensing and actuation. A flock of turtlebots would serve as an excellent testing platform, with an overhead camera looking down to gather global flock information for shepherding. Turtlebots come with the added benefit of strong support for simulation for algorithm validation before the big leap to hardware. The assumption from conventional methods that an agent can move holonomically by a (dx,dy) breaks down on these differential drive robots, so there will be inefficiencies and possibly failures from these methods even without considering real-world uncertainty. We would use this platform to analyze how conventional and learning-based methods compare on actual hardware, with a focus on shepherding success rates, and time to shepherding completion for these different approaches.

This research direction explores how to integrate probabilistic estimates into reward shaping in order to address the shortcomings of these algorithms faced with real world uncertainty. With plan based reward shaping, we could integrate a plan generated from a conventional shepherding approach into a learning-based approach. This plan would have to be probabilistic to reflect the system's uncertainty, representing the *probability* that following a conventional approach will result in successful shepherding. The learning-based approach would use experiences from the real world to determine when following the conventional approach is actually the best action versus when it is best to perform a non-trivial learned action.

5.3 Learning with Local Information

Many swarm shepherding approaches assume that the leader agents have access to global information about the swarm, but this will not be the case in many shepherding applications. Instead,

leader agents will need to be able to guide the swarm only with information they can gather from local observations.

This research direction investigates how Potential Based Reward Shaping could help leaders identify which followers have the most *potential* to impact the system. For instance, a leader could generate a null counterfactual for a specific follower within its observation radius to determine how valuable it is for the leader to guide that follower. This is similar to Counterfactual as Potential, but instead of trying to capture the potential of the entire system with one agent removed, we are concerned with capturing the potential of one non-learning follower agent, which is not trivial. This potential might be estimated by a leader counterfactually treating that follower as a copy of itself and computing what that follower's contribution would be in that case. Estimating this potential correctly would make leaders more efficient by incentivizing leaders to guide specific followers rather than all nearby followers.

6 Societal Impact

In considering the societal impact of intelligent swarm systems, it is critical to consider different political and economic incentives, and how these systems might be used to tackle global problems.

6.1 Automation of Warfare and Mass Surveillance

Intelligent swarms currently have a spot in automated warfare in performing reconnaissance [75, 134]. A swarm of UAVs constitutes a practical system for performing reconnaissance with little risk to any people operating the swarm, as the individual UAVs would be the targets of any resistance. This gives a military advantage to a country that can effectively deploy these swarms, offering a political incentive for country leaders to develop these systems [123]. While this may in the short-term remove some soldiers out of harm's way in warfare, this also promotes warfare in the long term. The more warfare is automated, the less the leaders of a country have to give up political power in order to gain support from their constituents, as explained in Selectorate Theory from political science [52]. They are not endangering people; they are endangering robots. This brings down the political cost of warfare, and sways the balance of power towards countries with many automated systems for warfare. These countries could enter conflicts with greater triviality. Intelligent swarms could later be equipped with weapons as the technology is developed further, which brings into question whether an automated swarm of killing robots is under any circumstances ethically permissible.

While swarm literature tends to stray away from discussions of automated warfare, there is no shortage of papers discussing improvements to swarm algorithms for better surveillance [115, 146]. Even with the proposed research directions focused on moving swarms to "goal locations" or "points of interest", one could simply replace these with "people of interest" and these research directions become relevant to surveillance. A surveillance system leveraging a swarm of UAVs could be deployed to monitor large populations by a government with a strong interest in keeping its population under control. This would make quite an effective tool in an oppressor's toolbox for suppressing dissent by stopping any resistance before it can even be organized.

6.2 More Efficient Transportation

Not all intelligent swarm systems are poised to create a dystopian world - some applications are simply mundane. There are strong economic incentives for intelligent swarms to be used for better coordination of air traffic control and autonomous vehicles [11, 87, 122]. Leader-follower systems are

especially applicable here, as one air traffic control tower (a leader) may not have any control over what other air traffic control towers (followers) actually do, but it may be possible to intelligently predict how to influence air traffic given what these other towers are doing. Similarly, a fleet of autonomous vehicles or intelligent traffic lights (leaders) may not have control over how human drivers (followers) drive, but could influence these drivers to bring down traffic congestion and improve driving efficiency. Such systems have the potential to make our daily lives more efficient, though not necessarily easier, as productivity expectations will rise to meet that increased efficiency. Additionally, there is the question of whether improving these systems is even a good idea in the first place. In the case of autonomous vehicles, we may be much better off with better public transportation that supports walkable cities rather than systems that force transportation to be car-dependent [27, 90, 136]. There is a glimmering hope here that these intelligent swarms could be used to better coordinate healthcare services as the same leader-follower ideas apply to a distributed network of healthcare providers [118], but the formulation for this is a bit more nebulous.

6.3 Addressing the Climate Crisis

There is hope for an indisputably positive societal impact from intelligent swarms. While there are not yet strong economic or political incentives for this technology to be deployed in order to address the climate crisis, this may change as this problem becomes more pressing [94, 126].

Consider the problem of wildfires that have become more prevalent. Intelligent swarms have the unique capability of autonomously performing complex coordination tasks in many places simultaneously. In the case of wildfires, a swarm of UAVs could provide valuable intel on which areas require immediate attention, and make intelligent decisions about where to focus observations [114, 130]. Similarly, in search and rescue operations arising from fires, floods, earthquakes, and winter storms, such a swarm would be capable of covering a large space quickly [9, 28], and could even be retrofitted to deliver supplies to those in immediate need [12].

We can have hope that these systems could also be used to better understand and clean our oceans in order to preserve and heal them [8, 64, 91, 147]. An intelligent swarm could cover a far larger area than a single robot exploring oceans or tracking natural phenomena. These swarms would also be effective in collecting trash from our oceans [93, 133], for which there are already existing robots capable of collecting ocean trash [60]. Similarly, a swarm could help with cleaning an oil spill [147], where many robots autonomously cleaning a large space in the ocean would be far more effective than trying to perform it all manually. Admittedly, these systems are best suited for providing band-aid solutions to these problems rather than tackling the root problem of building more sustainable practices for life on Earth, but these problems do need to be addressed.

7 Conclusions

This work outlined the state-of-the-art for reward shaping in multiagent systems, as well as the impact of learning in the leader-follower system of swarm shepherding. By pulling inspiration from both of these fields, we propose three novel research directions focused on addressing specific gaps at the intersection of both fields. The first is to use credit assignment from reward shaping to split a leader-follower swarm across various tasks. The second is to analyze state-of-the-art algorithms on hardware and explore probabilistic reward shaping as a means of better adapting these algorithms for the real world. The third looks at how reward shaping might improve the state-of-the-art for coordinating a leader-follower swarm with only local information. Intelligent swarms have a strong potential to do plenty of harm, but there is hope these systems could be deployed to address existential challenges.

References

[1] Hussein Abbass, John Harvey, and Kate Yaxley. Lifelong Testing of Smart Autonomous Systems by Shepherding a Swarm of Watchdog Artificial Intelligence Agents. *arXiv e-prints*, art. arXiv:1812.08960, December 2018. doi: 10.48550/arXiv.1812.08960.

[2] Hussein A. Abbass, Sondoss Elsawah, Eleni Petraki, and Robert Hunjet. Machine education: Designing semantically ordered and ontologically guided modular neural networks. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 948–955, 2019. doi: 10.1109/SSCI44817.2019.9003083.

[3] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic domains. *Autonomous Agents and Multi-Agent Systems*, 17, 2008.

[4] Adrian Agogino and Kagan Tumer. Efficient evaluation functions for multi-rover systems. In *Genetic and Evolutionary Computation–GECCO 2004*, pages 1–11. Springer, 2004.

[5] Adrian K Agogino and Kagan Tumer. A multiagent approach to managing air traffic flow. *Autonomous Agents and Multi-Agent Systems*, 24:1–25, 2012.

[6] Mehdi Ahrarinouri, Mohammad Rastegar, and Ali Reza Seifi. Multiagent reinforcement learning for energy management in residential buildings. *IEEE Transactions on Industrial Informatics*, 17(1):659–666, 2021. doi: 10.1109/TII.2020.2977104.

[7] Stuart Russell Andrew Y. Ng, Daishi Harada. Policy invariance under reward transformations: Theory and application to reward shaping. *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287, 1999.

[8] Ali Arjan. Carbon based microrobotics & swarm microrobotics for environmental cleaning. 2022.

[9] Ross Arnold, Jonathan Jablonski, Benjamin Abruzzo, and Elizabeth Mezzacappa. Heterogeneous uav multi-role swarming behaviors for search and rescue. In *2020 IEEE Conference on Cognitive and Computational Aspects of Situation Management (CogSIMA)*, pages 122–128, 2020. doi: 10.1109/CogSIMA49017.2020.9215994.

[10] Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

[11] Michael O Ball, Chien-Yu Chen, Robert Hoffman, and Thomas Vossen. *Collaborative decision making in air traffic management: Current and future research directions*. Springer, 2001.

[12] Debapriya Banik, Niamat Ullah Ibne Hossain, Kannan Govindan, Farjana Nur, and Kari Babski-Reeves. A decision support model for selecting unmanned aerial vehicle for medical supplies: Context of covid-19 pandemic. *The International Journal of Logistics Management*, 2022.

[13] Michael Baumann. *Learning Shepherding Behavior*. PhD thesis, University of Paderborn, 2015.

[14] Brandon Bennett and Matthew Trafankowski. A comparative investigation of herding algorithms. In *Proc. Symp. on Understanding and Modelling Collective Phenomena (UMoCoP)*, pages 33–38, 2012.

[15] Florian Berlinger, Melvin Gauci, and Radhika Nagpal. Implicit coordination for 3d underwater collective behaviors in a fish-inspired robot swarm. *Science Robotics*, 6(50):eabd8668, 2021.

[16] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'99, page 478–485. Morgan Kaufmann Publishers Inc., 1999.

[17] Lucian Busoniu, Robert Babuska, and Bart De Schutter. Multi-agent reinforcement learning: A survey. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6, 2006. doi: 10.1109/ICARCV.2006.345353.

[18] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008. doi: 10.1109/TSMCC.2007.913919.

[19] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pages 183–221, 2010.

[20] Jacopo Castellini, Sam Devlin, Frans A Oliehoek, and Rahul Savani. Difference rewards policy gradients. *Neural Computing and Applications*, pages 1–24, 2022.

[21] Hande Çelikkanat, Ali Emre Turgut, and Erol Şahin. *Guiding a Robot Flock via Informed Robots*, pages 215–225. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ISBN 978-3-642-00644-9. doi: 10.1007/978-3-642-00644-9_19. URL `https://doi.org/10.1007/978-3-642-00644-9_19`.

[22] Luiz Chaimowicz and Vijay Kumar. Aerial shepherds: Coordination among uavs and swarms of robots. In *Distributed Autonomous Robotic Systems 6*, pages 243–252. Springer, 2007.

[23] Young-Cheol Choi and Hyo-Sung Ahn. A survey on multi-agent reinforcement learning: Coordination problems. In *Proceedings of 2010 IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications*, pages 81–86, 2010. doi: 10.1109/MESA.2010.5552089.

[24] Nicholas R. Clayton and Hussein Abbass. Machine teaching in hierarchical genetic reinforcement learning: Curriculum design of reward functions for swarm shepherding. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 1259–1266, 2019. doi: 10.1109/CEC.2019.8790157.

[25] Mitchell Colby and Kagan Tumer. Shaping fitness functions for coevolving cooperative multiagent systems. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, page 425–432, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 0981738117.

[26] Mitchell K Colby and Kagan Tumer. Shaping fitness functions for coevolving cooperative multiagent systems. In *AAMAS*, volume 1, pages 425–432. Citeseer, 2012.

[27] Elisa Conticelli, Athanasios Maimaris, George Papageorgiou, and Simona Tondelli. Planning and designing walkable cities: a smart approach. *Smart planning: Sustainability and mobility in the age of change*, pages 251–269, 2018.

[28] Micael Santos Couceiro. An overview of swarm robotics for search and rescue applications. *Artificial Intelligence: Concepts, Methodologies, Tools, and Applications*, pages 1522–1561, 2017.

[29] William Curran. Using rubi to partition agents in air traffic problems with hard constraints and reward shaping. 2013.

[30] Essam Debie, Hemant Singh, Saber Elsayed, Anthony Perry, Robert Hunjet, and Hussein Abbass. A neuro-evolution approach to shepherding swarm guidance in the face of uncertainty. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2634–2641, 2021. doi: 10.1109/SMC52423.2021.9659082.

[31] S. Devlin and D. Kudenko. Dynamic potential-based reward shaping. *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440, 06 2012.

[32] S. Devlin and D. Kudenko. Plan-based reward shaping for multi-agent reinforcement learning. *The Knowledge Engineering Review*, 31(1):44–58, 01 2016. Copyright - © Cambridge University Press, 2016; Last updated - 2022-11-12.

[33] Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th international conference on autonomous agents and multiagent systems*, pages 225–232. ACM, 2011.

[34] Sam Devlin, Marek Grześ, and Daniel Kudenko. Multi-agent, reward shaping for robocup keepaway. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1227–1228, 2011.

[35] Sam Devlin, Marek Grześ, and Daniel Kudenko. Multi-agent, reward shaping for robocup keepaway. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1227–1228, 2011.

[36] Noel E. Du Toit and Joel W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2012. doi: 10.1109/TRO.2011.2166435.

[37] Adam Eck, Leen-Kiat Soh, Sam Devlin, and Daniel Kudenko. Potential-based reward shaping for pomdps. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1123–1124, 2013.

[38] K. Efthymiadis and D. Kudenko. Using plan-based reward shaping to learn strategies in starcraft: Broodwar. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8, 2013. doi: 10.1109/CIG.2013.6633622.

[39] Sara El Hamdani, Salaheddine Loudari, Stanislav Novotny, Petr Bouchner, and Nabil Benamar. A markov decision process model for a reinforcement learning-based autonomous pedestrian crossing protocol. In *2021 3rd IEEE Middle East and North Africa COMMunications Conference (MENACOMM)*, pages 147–151, 2021. doi: 10.1109/MENACOMM50742.2021.9678310.

[40] Tom Erez and William D. Smart. What does shaping mean for computational reinforcement learning? In *2008 7th IEEE International Conference on Development and Learning*, pages 215–219, 2008. doi: 10.1109/DEVLRN.2008.4640832.

[41] Richard E. Fikes and Nils J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 2(3):189–208, 1971. ISSN 0004-3702.

[42] Qingxu Fu, Tenghai Qiu, Zhiqiang Pu, Jianqiang Yi, and Wanmai Yuan. A cooperation graph approach for multiagent sparse reward reinforcement learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2022. doi: 10.1109/IJCNN55064.2022.9891991.

[43] Kaoru Fujioka. Effective herding in shepherding problem in v-formation control. *Transactions of the Institute of Systems, Control and Information Engineers*, 31(1):21–27, 2018.

[44] Kaoru Fujioka and Sakiko Hayashi. Effective shepherding behaviours using multi-agent systems. In *2016 IEEE Region 10 Conference (TENCON)*, pages 3179–3182, 2016. doi: 10.1109/TENCON.2016.7848636.

[45] Melvin Gauci, Monica E Ortiz, Michael Rubenstein, and Radhika Nagpal. Error cascades in collective behavior: a case study of the gradient algorithm on 1000 physical agents. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 1404–1412, 2017.

[46] Alexander Gee and Hussein Abbass. Transparent machine education of neural networks for swarm shepherding using curriculum design. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2019. doi: 10.1109/IJCNN.2019.8852209.

[47] Matthew L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30(1):35–79, 1986. ISSN 0004-3702. doi: https://doi.org/10.1016/0004-3702(86)90067-6. URL `https://www.sciencedirect.com/science/article/pii/0004370286900676`.

[48] Jorge Gomes, Pedro Mariano, and Anders Lyhne Christensen. Dynamic team heterogeneity in cooperative coevolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 22(6):934–948, 2018. doi: 10.1109/TEVC.2017.2779840.

[49] Nicolás Gómez, Néstor Peña, Sergio Rincón, Sindy Amaya, and Juan Calderon. Leader-follower behavior in multi-agent systems for search and rescue based on pso approach. In *SoutheastCon 2022*, pages 413–420. IEEE, 2022.

[50] M. Grzes and D. Kudenko. Plan-based reward shaping for reinforcement learning. In *2008 4th International IEEE Conference Intelligent Systems*, volume 2, pages 10–22–10–29, 2008. doi: 10.1109/IS.2008.4670492.

[51] Marek Grzeundefined. Reward shaping in episodic reinforcement learning. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, page 565–573, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

[52] Aldric Hama. The dictator's handbook: Why bad behavior is almost always good politics. *The Journal of Social, Political, and Economic Studies*, 38(1):126, 2013.

[53] Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–21, 2023. doi: 10.1109/TNNLS.2023.3236361.

[54] Wei Hao and Xinying Xu. Immune ant colony optimization network algorithm for multi-robot path planning. In *2014 IEEE 5th International Conference on Software Engineering and Service Science*, pages 1118–1121, 2014. doi: 10.1109/ICSESS.2014.6933762.

[55] Ahad Harati, Majid Nili Ahmadabadi, and Babak Nadjar Araabi. Knowledge-based multiagent credit assignment: A study on task type and critic information. *IEEE Systems Journal*, 1(1):55–67, 2007. doi: 10.1109/JSYST.2007.901641.

[56] Shin nyeong Heo, Sheng yu Lu, Ji-sun Shin, and Hee-hyol Lee. Multi-robot-multi-target path planning and position estimation for disaster area. In *2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT)*, pages 1–4, 2018. doi: 10.1109/ICT-ROBOT.2018.8549910.

[57] Chris HolmesParker, Adrian Agogino, and Kagan Tumer. Clean rewards for improving multiagent coordination in the presence of exploration. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1113–1114, 2013.

[58] Chris Holmesparker, Mathew E. Taylor, Adrian K. Agogino, and Kagan Tumer. Clean rewards to improve coordination by removing exploratory action noise. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, volume 3, pages 127–134, 2014. doi: 10.1109/WI-IAT.2014.159.

[59] Chris HolmesParker, Mathew E Taylor, Kagan Tumer, and Adrian Agogino. Cleaning the reward: counterfactual actions to remove exploratory action noise in multiagent learning. In *International Conference on Autonomous Agents and Multiagent Systems*, number ARC-E-DAA-TN13699, 2014.

[60] Rozemarijn Roland Holst. The netherlands: The 2018 agreement between the ocean cleanup and the netherlands. *The International Journal of Marine and Coastal Law*, 34(2):351–371, 2019.

[61] Hiroyuki Hoshi, Ichiro Iimura, Shigeru Nakayama, Yoshifumi Moriyama, and Ken Ishibashi. Computer simulation based robustness comparison regarding agents' moving-speeds in two- and three-dimensional herding algorithms. In *2018 joint 10th international conference on soft computing and intelligent systems (scis) and 19th international symposium on advanced intelligent systems (isis)*, pages 1307–1314. IEEE, 2018.

[62] Hiroyuki Hoshi, Ichiro Iimura, Shigeru Nakayama, Yoshifumi Moriyama, and Ken Ishibashi. Robustness of herding algorithm with a single shepherd regarding agents' moving speeds. *Journal of Signal Processing*, 22(6):327–335, 2018.

[63] Aya Hussein, Eleni Petraki, Sondoss Elsawah, and Hussein A Abbass. Autonomous swarm shepherding using curriculum-based reinforcement learning. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 633–641, 2022.

[64] Jules S Jaffe, Peter JS Franks, Paul LD Roberts, Diba Mirza, Curt Schurgers, Ryan Kastner, and Adrien Boch. A swarm of autonomous miniature underwater robot drifters for exploring submesoscale ocean dynamics. *Nature communications*, 8(1):14189, 2017.

[65] Shengde Jia, Lincheng Shen, and Hongtao Xue. Continuous-time markov decision process with average reward: Using reinforcement learning method. In *2015 34th Chinese Control Conference (CCC)*, pages 3097–3100, 2015. doi: 10.1109/ChiCC.2015.7260117.

[66] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos, and Miroslav Pajic. Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning. *arXiv e-prints*, art. arXiv:1909.07299, September 2019. doi: 10.48550/arXiv.1909.07299.

[67] Athanasios Ch Kapoutsis, Savvas A Chatzichristofis, Lefteris Doitsidis, Joao Borges De Sousa, Jose Pinto, Jose Braga, and Elias B Kosmatopoulos. Real-time adaptive multi-robot exploration with application to underwater map construction. *Autonomous robots*, 40:987–1015, 2016.

[68] Adarsh Kesireddy, Wanliang Shan, and Hao Xu. Global optimal path planning for multi-agent flocking: A multi-objective optimization approach with nsga-iii. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 64–71, 2019. doi: 10.1109/SSCI44817.2019.9002956.

[69] Seung-Mok Lee, Hanguen Kim, Hyun Myung, and Xin Yao. Cooperative coevolutionary algorithm-based model predictive control guaranteeing stability of multirobot formation. *IEEE Transactions on Control Systems Technology*, 23(1):37–51, 2015. doi: 10.1109/TCST.2014.2312324.

[70] SeungHwan Lee. A multi-robot balanced coverage path planning strategy for patrol missions. In *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pages 1567–1569, 2021. doi: 10.23919/ICCAS52745.2021.9649836.

[71] David Lewis. *Counterfactuals and Comparative Possibility*, pages 57–85. Springer Netherlands, Dordrecht, 1981. ISBN 978-94-009-9117-0. doi: 10.1007/978-94-009-9117-0_3. URL `https://doi.org/10.1007/978-94-009-9117-0_3`.

[72] Chun-Gui Li, Meng Wang, and Qing-Neng Yuan. A multi-agent reinforcement learning using actor-critic methods. In *2008 International Conference on Machine Learning and Cybernetics*, volume 2, pages 878–882, 2008. doi: 10.1109/ICMLC.2008.4620528.

[73] Li Li, Jun Wang, Wei Li, Qihang Peng, Xiaonan Chen, and Shaoqian Li. Decentralized decision for multi-band sensing: A deep reinforcement learning approach. *IEEE Wireless Communications Letters*, 10(12):2674–2677, 2021. doi: 10.1109/LWC.2021.3111750.

[74] Mengyuan Li, Bin Guo, Jiangshan Zhang, Jiaqi Liu, Sicong Liu, Zhiwen Yu, Zhetao Li, and Liyao Xiang. Decentralized multi-agv task allocation based on multi-agent reinforcement learning with information potential field rewards. In *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 482–489, 2021. doi: 10.1109/MASS52906.2021.00066.

[75] Rui Li and Hongzhong Ma. Research on uav swarm cooperative reconnaissance and combat technology. In *2020 3rd International Conference on Unmanned Systems (ICUS)*, pages 996–999. IEEE, 2020.

[76] Tianxu Li, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Qihui Wu, Yang Zhang, and Bing Chen. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys Tutorials*, 24(2):1240–1279, 2022. doi: 10.1109/COMST.2022.3160697.

[77] Jyh-Ming Lien, O.B. Bayazit, R.T. Sowell, S. Rodriguez, and N.M. Amato. Shepherding behaviors. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 4, pages 4159–4164 Vol.4, 2004. doi: 10.1109/ROBOT.2004.1308924.

[78] Jyh-Ming Lien, S. Rodriguez, J. Malric, and N.M. Amato. Shepherding behaviors with multiple shepherds. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 3402–3407, 2005. doi: 10.1109/ROBOT.2005.1570636.

[79] Nathan K Long, Daniel Sgarioto, Matthew Garratt, Karl Sammut, and Hussein Abbass. Multi-vessel sea state estimation utilising swarm shepherding. In *RINA, Royal Institution of Naval Architects-Pacific 2019 International Maritime Conference, IMC 2019*, pages 173–184. Royal Institution of Naval Architects, 2019.

[80] Nathan K. Long, Karl Sammut, Daniel Sgarioto, Matthew Garratt, and Hussein A. Abbass. A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 4(4):523–537, 2020. doi: 10.1109/TETCI.2020.2992778.

[81] Nathan K Long, Matthew Garratt, Karl Sammut, Daniel Sgarioto, and Hussein A Abbass. Shepherding autonomous goal-focused swarms in unknown environments using hilbert space-filling paths. *Shepherding UxVs for Human-Swarm Teaming: An Artificial Intelligence Approach to Unmanned X Vehicles*, pages 31–50, 2021.

[82] Sebastian Mai, Maximilian Deubel, and Sanaz Mostaghim. Multi-objective roadmap optimization for multiagent navigation. In *2022 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2022. doi: 10.1109/CEC55065.2022.9870300.

[83] Melinda Malley, Bahar Haghighat, Lucie Houel, and Radhika Nagpal. Eciton robotica: Design and algorithms for an adaptive self-assembling soft robot collective. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4565–4571, 2020. doi: 10.1109/ICRA40945.2020.9196565.

[84] Patrick Mannion, Jim Duggan, and Enda Howley. Generating multi-agent potential functions using counterfactual estimates. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, pages 3643–3652, 2016.

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

[86] Patrick Mannion, Sam Devlin, Jim Duggan, and Enda Howley. Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning. *The Knowledge Engineering Review*, 33:e23, 2018.

[87] Wilfred S Manuela Jr et al. Factors affecting airline profits: Evidence from the philippines. *Journal of Applied Business Research (JABR)*, 27(6):17–22, 2011.

[88] B. Marthi. Automatic shaping and decomposition of reward functions. ICML '07, page 601–608. Association for Computing Machinery, 2007. ISBN 9781595937933. doi: 10.1145/1273496.1273572.

[89] Maciej A. Mazurowski and Jacek M. Zurada. Solving decentralized multi-agent control problems with genetic algorithms. In *2007 IEEE Congress on Evolutionary Computation*, pages 1029–1034, 2007. doi: 10.1109/CEC.2007.4424583.

[90] William R McShane, Arnold Jay Bloch, and William Ihlo. *The energy advantages of public transportation*, volume 1. US Urban Mass Transportation Administration Office of Policy Research . . . , 1980.

[91] Runlong Miao, Shuo Pang, and Dapeng Jiang. Development of an inexpensive decentralized autonomous aquatic craft swarm system for ocean exploration. *Journal of Marine Science and Application*, 18:343–352, 2019.

[92] Tsutomu Miki and Tetsuya Nakamura. An effective simple shepherding algorithm suitable for implementation to a multi-mmobile robot system. In *First International Conference on Innovative Computing, Information and Control-Volume I (ICICIC'06)*, volume 3, pages 161–165. IEEE, 2006.

[93] Manilo Monaco, Mario GCA Cimino, Gigliola Vaglini, Francesco Fusai, and Giovanni Nico. Managing the oceans cleanup via sea current analysis and bio-inspired coordination of usv swarms. In *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, pages 8344–8347. IEEE, 2021.

[94] Gregory F Nemet. Robust incentives and the design of a climate change governance regime. *Energy Policy*, 38(11):7216–7225, 2010.

[95] Hung Nguyen, Matthew Garratt, and Hussein Abbass. Apprenticeship bootstrapping. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2018. doi: 10.1109/IJCNN.2018.8489064.

[96] Hung T Nguyen, Tung D Nguyen, Matthew Garratt, Kathryn Kasmarik, Sreenatha Anavatti, Michael Barlow, and Hussein A Abbass. A deep hierarchical reinforcement learner for aerial shepherding of ground swarms. In *Neural Information Processing: 26th International Conference, ICONIP 2019, Sydney, NSW, Australia, December 12–15, 2019, Proceedings, Part I*, pages 658–669. Springer, 2019.

[97] Hung The Nguyen, Matthew Garratt, Lam Thu Bui, and Hussein Abbass. Apprenticeship bootstrapping: Inverse reinforcement learning in a multi-skill uav-ugv coordination task. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2204–2206, 2018.

[98] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9):3826–3839, 2020. doi: 10.1109/TCYB.2020.2977374.

[99] Tung Nguyen, Jing Liu, Hung Nguyen, Kathryn Kasmarik, Sreenatha Anavatti, Matthew Garratt, and Hussein Abbass. Perceptron-learning for scalable and transparent dynamic formation in swarm-on-swarm shepherding. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. doi: 10.1109/IJCNN48605.2020.9207539.

[100] Anil Özdemir, Melvin Gauci, and Roderich Groß. Shepherding with robots that do not compute. In *Artificial Life Conference Proceedings*, pages 332–339. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info . . . , 2017.

[101] J. Duggan P. Mannion, S. Devlin and E. Howley. Reward shaping for knowledge-based multi-objective multi-agent reinforcement learning. *The Knowledge Engineering Review*, 33:e23, 2018. doi: 10.1017/S0269888918000292.

[102] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11:387–434, 2005.

[103] Alyssa Pierson and Mac Schwager. Bio-inspired non-cooperative multi-robot herding. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1843–1849. IEEE, 2015.

[104] Scott Proper and Kagan Tumer. Modeling difference rewards for multiagent learning. In *AAMAS*, pages 1397–1398, 2012.

[105] Scott Proper and Kagan Tumer. Multiagent learning with a noisy global reward signal. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 826–832, 2013.

[106] Zahra Rahaie and Hamid Beigy. Addition of learning to critic agent as a solution to the multi-agent credit assignment problem. In *2009 Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control*, pages 1–4, 2009. doi: 10.1109/ICSCCW.2009.5379439.

[107] Aida Rahmattalabi, Jen Jen Chung, Mitchell Colby, and Kagan Tumer. D++: Structural credit assignment in tightly coupled multiagent domains. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4424–4429, 2016. doi: 10.1109/IROS.2016.7759651.

[108] J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. pages 463–471, 01 1998.

[109] Sazalinsyah Razali, Nurul Fathiyah Shamsudin, Mashanum Osman, Qinggang Meng, and Shuang-Hua Yang. Flock identification using connected components labeling for multi-robot shepherding. In *2013 International Conference on Soft Computing and Pattern Recognition (SoCPaR)*, pages 298–303. IEEE, 2013.

[110] Carrie Rebhuhn, Ryan Skeele, Jen Jen Chung, Geoffrey A. Hollinger, and Kagan Tumer. Learning to trick cost-based planners into cooperative behavior. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4627–4633, 2015. doi: 10.1109/IROS.2015.7354036.

[111] Craig W Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 25–34, 1987.

[112] D. Kudenko S. Devlin and M. Grześ. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 14:251–278, 2011.

[113] D. Kudenko S. Devlin, L. Yliniemi and K. Tumer. Potential-based difference rewards for multiagent reinforcement learning. *Conference on Autonomous Agents and Multi-Agent Systems*, 2014.

[114] Fabrice Saffre, Hanno Hildmann, Hannu Karvonen, and Timo Lind. Monitoring and cordoning wildfires with an autonomous swarm of unmanned aerial vehicles. *Drones*, 6(10):301, 2022.

[115] Martin Saska, Vojtěch Vonásek, Jan Chudoba, Justin Thomas, Giuseppe Loianno, and Vijay Kumar. Swarm distribution and deployment for cooperative surveillance by micro-aerial vehicles. *Journal of Intelligent & Robotic Systems*, 84:469–492, 2016.

[116] Paul Scharre. How swarming will change warfare. *Bulletin of the atomic scientists*, 74(6): 385–389, 2018.

[117] Alan Schultz, John Grefenstette, and William Adams. Robo-shepherd: Learning complex robotic behaviors. 03 1999.

[118] Elhadi Shakshuki and Malcolm Reid. Multi-agent system applications in healthcare: current technology and future roadmap. *Procedia Computer Science*, 52:252–261, 2015.

[119] Kun Shao, Yuanheng Zhu, Zhentao Tang, and Dongbin Zhao. Cooperative multi-agent deep reinforcement learning with counterfactual reward. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020. doi: 10.1109/IJCNN48605.2020.9207169.

[120] Rajneesh Sharma and Matthijs T. J. Spaan. Fuzzy reinforcement learning control for decentralized partially observable markov decision processes. In *2011 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2011)*, pages 1422–1429, 2011. doi: 10.1109/FUZZY.2011.6007675.

[121] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial intelligence*, 171(7):365–377, 2007.

[122] Auyon Siddiq and Terry A Taylor. Ride-hailing platforms: Competition and autonomous vehicles. *Manufacturing & Service Operations Management*, 24(3):1511–1528, 2022.

[123] Peter Warren Singer. *Wired for war: The robotics revolution and conflict in the 21st century*. Penguin, 2009.

[124] Arambam James Singh and Akshat Kumar. Approximate difference rewards for scalable multigent reinforcement learning. 2021.

[125] International Sheep Dog Society. "rules for trials". 2018. URL `https://www.isds.org.uk/trials/sheepdog-trials/rules-for-trials/`.

[126] Rory Sullivan. *Corporate responses to climate change: Achieving emissions reductions through regulation, self-regulation and economic incentives*. Routledge, 2017.

[127] K.C. Tan, T.H. Lee, Y.J. Yang, and D.S. Liu. A cooperative coevolutionary algorithm for multiobjective optimization. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 2, pages 1926–1931 vol.2, 2004. doi: 10.1109/ICSMC.2004.1399965.

[128] Yusuke Tsunoda, Yuichiro Sueoka, Yuto Sato, and Koichi Osuka. Analysis of local-camera-based shepherding navigation. *Advanced robotics*, 32(23):1217–1228, 2018.

[129] Ovunc Tuzel, Gilberto Marcon dos Santos, Chloë Fleming, and Julie A Adams. Learning based leadership in swarm navigation. In *Swarm Intelligence: 11th International Conference, ANTS 2018, Rome, Italy, October 29–31, 2018, Proceedings 11*, pages 385–394. Springer, 2018.

[130] Georgios Tzoumas, Lenka Pitonakova, Lucio Salinas, Charles Scales, Thomas Richardson, and Sabine Hauert. Wildfire detection in large-scale environments using force-based control for swarms of uavs. *Swarm Intelligence*, pages 1–27, 2022.

[131] Richard Vaughan, J. Henderson, and N. Sumpter. Introducing the robot sheepdog project. In *Proc. Int. Workshop Robot. Automated Machinery Bio-Productions*, pages 1–7, 01 1997.

[132] Richard Vaughan, Neil Sumpter, Jane Henderson, Andy Frost, and Stephen Cameron. Experiments in automatic flock control. *Robotics and autonomous systems*, 31(1-2):109–117, 2000.

[133] Pamela Wang, Malika Meghjani, and Gong Chen. Marine trash collection: A multi-agent, multi-target search. In *OCEANS 2022, Hampton Roads*, pages 1–7. IEEE, 2022.

[134] Yubing Wang, Peng Bai, Xiaolong Liang, Weijia Wang, Jiaqiang Zhang, and Qixi Fu. Reconnaissance mission conducted by uav swarms based on distributed pso path planning algorithms. *IEEE access*, 7:105086–105099, 2019.

[135] Zhongya Wang, Min Li, Lianhang Dou, Yang Li, Qingying Zhao, and Jie Li. A novel multi-objective artificial bee colony algorithm for multi-robot path planning. In *2015 IEEE International Conference on Information and Automation*, pages 481–486, 2015. doi: 10.1109/ICInfA.2015.7279336.

[136] Richard E Wener and Gary W Evans. Comparing stress of car and train commuters. *Transportation research part F: traffic psychology and behaviour*, 14(2):111–116, 2011.

[137] Justin Werfel, Kirstin Petersen, and Radhika Nagpal. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343(6172):754–758, 2014.

[138] Simon A. Williamson, Enrico H. Gerding, and Nicholas R. Jennings. Reward shaping for valuing communications during multi-agent coordination. In *Autonomous Agents And MultiAgent Systems (30/04/09)*, pages 641–648, May 2009. URL `https://eprints.soton.ac.uk/267076/`. Event Dates: May, 2009.

[139] David H Wolpert, Kagan Tumer, and Keith Swanson. Optimal wonderful life utility functions in multi-agent systems. 2000.

[140] Rong Wu and Jin Xu. Learning markov decision processes based on genetic programming. In *2022 2nd Asia Conference on Information Engineering (ACIE)*, pages 72–76, 2022. doi: 10.1109/ACIE55485.2022.00023.

[141] Baicen Xiao, Bhaskar Ramasubramanian, and Radha Poovendran. Shaping Advice in Deep Multi-Agent Reinforcement Learning. *arXiv e-prints*, art. arXiv:2103.15941, March 2021. doi: 10.48550/arXiv.2103.15941.

[142] Connor Yates, Reid Christopher, and Kagan Tumer. Multi-fitness learning for behavior-driven cooperation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 453–461, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371285. doi: 10.1145/3377930.3390220. URL `https://doi.org/10.1145/3377930.3390220`.

[143] Connor Yates, Ayhan Alp Aydeniz, and Kagan Tumer. Reactive multi-fitness learning for robust multiagent teaming. In *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, pages 92–100, 2021. doi: 10.1109/MRS50823.2021.9620689.

[144] Logan Yliniemi and Kagan Tumer. Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In *Simulated Evolution and Learning: 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings 10*, pages 407–418. Springer, 2014.

[145] Zhong Yu, Gu Guochang, and Zhang Rubo. A new approach for structural credit assignment in distributed reinforcement learning systems. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 1, pages 1215–1220 vol.1, 2003. doi: 10.1109/ROBOT.2003.1241758.

[146] Won Joon Yun, Soohyun Park, Joongheon Kim, MyungJae Shin, Soyi Jung, David A. Mohaisen, and Jae-Hyun Kim. Cooperative multiagent deep reinforcement learning for reliable surveillance via autonomous multi-uav control. *IEEE Transactions on Industrial Informatics*, 18(10):7086–7096, 2022. doi: 10.1109/TII.2022.3143175.

[147] Emaad Mohamed H Zahugi, Mohamed M Shanta, and TV Prasad. Oil spill cleaning up using swarm of robots. In *Advances in Computing and Information Technology: Proceedings of the Second International Conference on Advances in Computing and Information Technology (ACITY) July 13-15, 2012, Chennai, India-Volume 3*, pages 215–224. Springer, 2013.

[148] Nick Zerbel and Kagan Tumer. The power of suggestion. In *AAMAS Conference proceedings*, 2020.

[149] Jixuan Zhi and Jyh-Ming Lien. Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(2):4163–4168, 2021. doi: 10.1109/LRA.2021.3068955.

[150] Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 33:11853–11864, 2020.