Autonomous Swarm Shepherding Using Curriculum-Based Reinforcement Learning

Aya Hussein
University of New South Wales
Canberra, Australia
a.hussein@adfa.edu.au

Eleni Petraki
Faculty of Education, University of Canberra
Canberra, Australia
eleni.petraki@canberra.edu.au

Sondoss Elsawah
University of New South Wales
Canberra, Australia
s.elsawah@adfa.edu.au

Hussein A. Abbass
University of New South Wales
Canberra, Australia
h.abbass@unsw.edu.au

ABSTRACT

Autonomous shepherding is a bio-inspired swarm guidance approach, whereby an artificial sheepdog guides a swarm of artificial or biological agents, such as sheep, towards a goal. While the success in this guidance depends on the set of behaviours exhibited by the sheepdog, the main source of complexity for learning effective behaviours lies within the highly non-linear dynamics featured among the swarm members as well as between the swarm and the sheepdog. Attempts to apply reinforcement learning (RL) to shepherding have so far relied greatly on rule-based algorithms for calculating waypoints to guide the RL algorithm. In this paper, we propose a curriculum-based approach for RL that does not rely on any external algorithm to pre-determine waypoints for the sheepdog. Instead, the approach uses task decomposition by formulating shepherding in terms of two sub-tasks: (1) pushing an agent from a start to a target location and (2) selecting between collecting scattered agents or driving the biggest cluster of agents to the goal. Simple-to-complex curriculum learning is used to accelerate the learning of each sub-task. For the first sub-task, the complexity is gradually increased over training time, whereas for the second sub-task a simplified environment is designed for initial learning before proceeding with the main environment. The proposed approach results in high-performance shepherding with a success rate of about 96%. While curriculum learning was found to expedite the learning of the first sub-task, it was not as efficient for the second sub-task. Our analyses highlight the need for the careful design of the curriculum to ensure that skills acquired in intermediate tasks are useful for the main tasks.

KEYWORDS

Curriculum Learning; Hierarchical Reinforcement Learning; Swarm Guidance; Machine Teaching

ACM Reference Format:
Aya Hussein, Eleni Petraki, Sondoss Elsawah, and Hussein A. Abbass. 2022. Autonomous Swarm Shepherding Using Curriculum-Based Reinforcement Learning. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Swarm navigation behaviours have been extensively studied by researchers. In their review of robot swarms' applications, Brambilla et al. [6] identified navigation as one of the four key behaviours that are necessary for swarms to tackle real-life complex problems. Swarm navigation algorithms have been largely inspired by navigation strategies found in Nature (e.g. flocking [8]). However, the application of such strategies in a fully distributed manner is not always practical for robotic systems involving simple swarm members. For instance, to enable scattered swarm members to cluster into a single group, these members need to possess the ability to calculate their positions in the environment or to have sensors with sensing ranges spanning the whole environment to sense every other member in the environment. Another example is flocking towards a distant target, which would require swarm members to perform complex path planning. These requirements may not be always possible for simple swarm members operating in large environments.

One bio-inspired strategy that supports the navigation of swarm members with low capabilities (relative to the navigation task) is shepherding, which is naturally exhibited in the guidance of sheep swarms. In shepherding, a sheepdog is used to influence sheep movements to guide them to a given destination. The use of a sheepdog in this setting facilitates the control of the swarm due to the cognitive and physical superiority of the dog compared to swarm members [2]. A well-trained sheepdog uses such cognitive and physical advantages and exploits sheep's behavioural tendencies to herd the flock into a designated target area [43]. As such, shepherding strategies enable the efficient guidance and control of simplistic swarm members by employing a more capable agent.

Mathematical modelling of sheepdog-sheep interactions enabled the reproduction of the shepherding capability in robot systems. Robot-based shepherding has potential applications in several civil and military domains [26]. Research on using a robotic agent for herding ducks and sheep in a paddock has already shown some promising results [11, 46]. The use of a robotic sheepdog is anticipated to be a cost-effective solution given the high cost of training and keeping biological sheepdogs. Past studies also suggested that robot-based shepherding could improve sheep safety whilst avoiding extra stress on the sheep [47].

Most existing shepherding strategies use rule-based algorithms for controlling the sheepdog behaviour [9, 13, 39, 44, 45]. However,

in real environments, the sheepdog behaviour may need to be adjusted in response to variations in environmental conditions or sheep behaviour. For instance, Evered et al. [11] showed that the response of biological sheep to a robotic sheepdog changes over time as the sheep get accustomed to the robot. Learning-based algorithms are thus preferred due to their ability to adapt to changes in the task. Only a handful of studies used learning algorithms for shepherding. However, the existing learning-based shepherding algorithms either achieved very low performance [7, 15] or relied on external rule-based shepherding algorithms to calculate the waypoints for the shepherd and used these waypoints as input to the learning algorithm [16, 33]. Using a rule-based algorithm for waypoints calculation is an oversimplification of the learning problem by turning it into a mere scaling problem [33].

This work aims to propose a reinforcement learning (RL)–based approach for learning an effective shepherding policy. To overcome the complexity of the learning process, the shepherding problem is first divided into a hierarchy of two sub-tasks: (1) the lower sub-task is learning to push one or more agents from position A to position B and (2) the higher sub-task is deciding on whether the shepherd should collect scattered agents (i.e. push the furthest sheep towards the center of mass of all the sheep) or drive the largest cluster to the goal (i.e. push the center of the largest sheep cluster towards the goal). A simple curriculum is then used to speed up the learning of each sub-task by commencing with simple scenarios, which then grow in complexity over time.

2 RELATED WORK

Lien et al. [23] defined shepherding behaviours as those in which external agents control the movement of a swarm of agents. Lien et al. identified four types of shepherding behaviours: driving, covering, patrolling, and collecting. Driving is the guidance of a swarm from one location to a goal location. Covering is guiding the swarm to visit all locations in the environment. Collecting comprises gathering scattered swarm members. Finally, patrolling involves protecting a designated region by preventing swarm members from entering it.

One of the common shepherding applications found in nature is the use of a sheepdog to guide the motion of a sheep swarm. A significant body of research has focused on the sheepdog behaviour to both understand and replicate it. Strömbom et al. [39] have found that sheepdog operation is made up of two key behaviours: collecting scattered sheep and driving the swarm once collected towards the goal area. Strömbom et al. [39] proposed an algorithm that resembles the sheepdog behaviour. In this algorithm, the collect behaviour is achieved by moving the sheepdog behind the furthest sheep and then along the vector from the furthest sheep to the global center of mass (GCM) of the swarm. For the driving behaviour, the sheepdog moves behind the GCM of the clustered sheep, facing the goal, and then moves along the vector from GCM to the goal area. The algorithm was shown to mimic the behaviour of a natural sheepdog.

Variants of Strömbom et al. [39]'s algorithm were proposed to improve task performance in terms of success rate and completion time. For instance, Hepworth et al. [20] argued that sheep agents demonstrate heterogeneous behaviours such that some sheep have more influence on the swarm than others. Consequently, Hepworth et al. [20] proposed that the center of influence (CoI) should be used instead of GCM when driving heterogeneous sheep. Another variant was proposed in [9] by imposing circular paths on sheepdog motion when approaching driving and collecting points. It was suggested that such circular paths are necessary to avoid undesirable swarm fragmentation.

However, rule-based algorithms for shepherding are prone to dramatic performance drops in response to low-to-moderate changes in the environment. For instance, El-Fiqi et al. [9] showed that the presence of obstacles in the environment severely impacts the performance such that with an obstacle density of as low as 4%, the success rate of a single-sheepdog shepherding scenario drops to zero. El-Fiqi et al. [9] suggested the need for increasing the number of sheepdogs to enable success in such environments.

The low adaptability of existing rule-based algorithms to task characteristics imposes additional limits on their performance. For instance, Strömbom et al. [39] found that the performance of their single sheepdog algorithm drops notably (in terms of success rate and task time) as the number of sheep increases. Lien and Pratt [24] also argued that existing algorithms can handle small swarms using a single sheepdog, but require multiple sheepdogs for swarms of 40 or more sheep. This is well inferior to the capabilities of biological sheepdogs, as a single sheepdog can herd swarms of 80 or more sheep [39].

The above discussed limitations of rule-based shepherding algorithms call for shepherding algorithms that can adjust their behaviour based on some given task characteristics. Learning-based algorithms naturally address the requirement for adaptation due to their inherent learning capabilities. Go et al. [16] used RL for shepherding tasks involving one sheepdog and multiple sheep. As the complexity of the shepherding problem is very high due to its large state and action spaces, Go et al. [16] used waypoints which are the driving/collection positions as calculated from Strömbom's algorithm. Go et al. discretised these waypoints and included them in the observation vector of SARSA algorithm, which is used for learning discrete actions for the sheepdog. A positive reward is given to the RL agent when the sheepdog reaches its waypoint. A similar attempt has been presented in [33] where Strömbom's algorithm is also used to calculate a waypoint for the sheepdog. The relative directional vectors between the sheepdog and the waypoint is then used as the observation vector. A deep reinforcement learning algorithm is used to generate continuous sheepdog actions. Continuous motion actions are preferred over discrete ones to enable smooth motion when executed on an actual robot.

While using waypoints as input for a learning algorithm makes learning significantly easier, this directly compromises the advantage of learning. This is because the problem is reduced to learning to move from location A to location B in a straight line to get the maximum reward, such that both A and B are provided as input to the RL algorithm. This is also noticed by Nguyen et al. [33] who found out that their formulation turned the problem into a scaling problem.

Two studies presented learning-based shepherding algorithms without relying on waypoints calculated from external rule-based shepherding algorithms. Clayton and Abbass [7] used hierarchical genetic RL to evolve a policy for the sheepdog. They used task

decomposition to design a multi-part reward function to facilitate learning of the different skills needed for shepherding. However, their analysis showed that the success rate achieved by the learnt policy was about 50%. The other study by Gee and Abbass [15] decomposed the task into two sub-tasks: collect and drive. They used supervised learning to learn suitable behaviours for each sub-task given data collected from humans operating a simulated sheepdog. The reported results showed low performance as the success rate was just about 32%.

Following this discussion of existing shepherding algorithms, two key findings can be concluded. First, learning-based algorithms for shepherding are required as they can naturally adapt to different contexts. Second, only a handful of studies presented learning-based algorithms for shepherding such that they either achieved very low performance or oversimplified the learning problem in a way that voids the advantages of learning. This paper addresses this research gap by using an approach based on curriculum learning to enable learning effective shepherding policies without relying on external shepherding algorithms. The next section gives a brief introduction to curriculum learning and discusses some of its key directions in RL domains.

3 CURRICULUM LEARNING

In RL domains, researchers have been studying various techniques to enable efficient learning of complex tasks. Task decomposition is one of the earliest attempts for accelerating RL [18]. It works by analysing a complex task into multiple partitions (or sub-tasks) such that learning the sub-tasks is easier than learning the whole task. Several approaches based on task decomposition have been used in the context of RL. These approaches include reward shaping where a reward is given when the agent reaches a milestone [18], hierarchical learning where temporal abstraction is used for tasks with temporally extended actions [28, 40], modular learning where different sub-tasks use different partitions of the state space [5, 19, 38], and learning sub-tasks independently [14, 42]. Reward shaping and hierarchical learning aim to mitigate the impact of sparse rewards on the complexity of long-horizon tasks, whereas modular learning and learning sub-tasks independently address the effect of large state and action spaces on the complexity of learning.

While task decomposition uses a divide-and-conquer style for learning, curriculum learning is concerned with the question of how learning should be progressed over time. In this work, we use the term *curriculum* in a way that aligns with its use in the ML literature, but we are aware that this term refers to a more comprehensive process in the human education literature [1, 21]. The emergence of curriculum-based approaches in ML can be traced back to 1993 when Elman [10] highlighted the importance of starting small when it comes to teaching a complex task to an ML model. Elman suggested that machine learners, akin to human beings, show improved learning capacity when learning starts with simple concepts before advancing to more complex ones. More recently, the term *curriculum learning* was introduced by Bengio et al. [4] to refer to the training strategies that organise training examples according to their level of difficulty such that easier examples are used for learning before more difficult ones. Bengio et al. showed that curriculum learning outperforms traditional non-curriculum training strategies in terms of both the speed of convergence and the magnitude of generalisation error. Subsequent studies confirmed these findings for different supervised learning tasks [3, 17, 22].

In RL domains, Narvekar et al. [30] reviewed existing approaches to, and proposed a framework for, curriculum learning. Their framework formulates curriculum learning in terms of three steps: (1) task generation (2) sequencing, and (3) transfer learning. Task generation is the process of creating a useful set of intermediate tasks which are used to facilitate learning of the main task. In the simplest case, no intermediate tasks are generated as only samples from the main task are used for learning. In this case, samples in the replay buffer, where past state-action-reward experience tuples are stored, can be sequenced in order of importance [36] or complexity [35]. No transfer learning is employed for the case with no intermediate tasks. In the more generic case, one or more intermediate tasks can be used which allows the learning to start under simplified settings before proceeding with the more complex setting. Intermediate tasks might be generated by introducing changes to one or more elements of the Markov Decision Process (MDP) of the main task: state space, action space, initial states, goal states, transfer function, and/or reward function [31, 41]. Different sequencing methods can be used for ordering the intermediate tasks including: simple-to-complex ordering [12], relying on a human for specifying the sequence [34], or formulating the sequencing as a search problem [32]. As the intermediate tasks have different MDPs, transfer learning is used to transfer knowledge from these tasks to the main task. Depending on the RL algorithm used for learning and the differences in MDPs between the intermediate and main tasks, transfer learning can be achieved by transferring policy and value functions, task model, training instances, or partial policies.

Our work employs both task decomposition and curriculum learning for facilitating the learning of the shepherding task. Task decomposition is achieved by analysing the shepherding problem and decomposing it into two sub-tasks. Then, curriculum learning with intermediate tasks is used within each sub-task to speed up their learning. Intermediate tasks are linearly sequenced to generate a simple-to-complex curriculum such that knowledge is transferred between tasks in the curriculum by transferring the policy and value functions.

4 PROBLEM STATEMENT

The single-sheepdog multiple-sheep shepherding problem consists of a sheepdog β with a position $P^t_\beta = (x_\beta, y_\beta)$ at time t and a swarm of N sheep $\{\pi_1, \pi_2, ...\pi_N\}$ with positions $P^t_{\pi_i} = (x_{\pi_i}, y_{\pi_i})$ for agent π_i at time t. All the agents are deployed within a continuous-space environment of dimensions $L \times L$. The environment has a designated goal area described in terms of the goal center $P_G = (x_G, y_G)$ and radius R_G. The positions of the sheepdog, sheep, and goal centre are randomly initialised in each shepherding scenario. The maximum linear velocity of the sheepdog and the sheep is V_β and V_π, respectively. β is assumed to be spatially aware of its position P^t_β, the positions of all the swarm members $P^t_{\pi_i} \forall i \in [1, N]$, and the goal center P_G. The objective is to gather all the sheep in the goal area within a predefined amount of time T_{max}. The task comes to an end at t_{end} either when all the sheep are successfully homed to the

Parameter	$R_{\pi\pi}$	$R_{\pi\beta}$	R_{Λ}	$W_{\pi\pi}$	$W_{\pi\beta}$	W_{Λ}	V_{π}	V_{β}
Value	0.4	45	5	1.05	1.0	2.0	1.5	1.0

Table 1: Sheep and sheepdog parameters.

goal area or when T_{max} is reached, whichever happens first. The task is considered successful if and only if the following condition is met:

$$||P_{\pi_i}^{t_{end}} - P_G|| < R_G \ \forall i \in [1, N] \tag{1}$$

such that $||P_{\pi_i}^{t_{end}} - P_G||$ is the Euclidean distance between $P_{\pi_i}^{t_{end}}$ and P_G.

5 METHODOLOGY

The aim of a shepherding algorithm is to generate a set of sheepdog actions that leads to successful operation given the state vector of a swarm of sheep in the environment. Changes in the state vector are caused by the behaviour exhibited by the swarm of sheep. In this paper, it is assumed that the sheep behaviour is determined by a set of rules that characterise the key behavioural tendencies of biological sheep. The sheep model is described in subsection 5.1. Meanwhile, the behaviour of the sheepdog is determined through learning. Subsection 5.2 analyses the learning into two sub-tasks and uses curriculum learning to learn each sub-task.

5.1 Sheep Behaviour

The behaviour of the sheep agents is determined by a set of predefined equations to mimic the behaviours of actual sheep, akin to previous studies (e.g. [13, 16, 24, 27, 43]). Three forces are used to determine sheep movement in the current work: cohesion, separation, and sheep-to-sheepdog repulsion. The total force applied on sheep agent π_i at time t is denoted $F_{\pi_i}^t$ and is calculated according to the equation:

$$F_{\pi_i}^t = W_{\pi\Lambda} F_{\pi_i\Lambda_{\pi_i}}^t + W_{\pi\pi} F_{\pi_i\pi_{-i}}^t + W_{\pi\beta} F_{\pi_i\beta}^t \tag{2}$$

$F_{\pi_i\Lambda_{\pi_i}}^t$ is the cohesion force that pushes π_i towards the local center of mass of its neighbours; i.e other sheep within the attraction range R_Λ of π_i. $F_{\pi_i\pi_{-i}}^t$ is the separation force used to avoid collisions between π_i and other sheep within the separation radius $R_{\pi\pi}$ of π_i. $F_{\pi_i\beta}^t$ is the repulsion force that causes π_i to escape from the sheepdog in the direction of the vector $(\vec{P_\beta} - \vec{P_{\pi_i}})$. $F_{\pi_i\beta}^t$ is only applied if π_i is within the influence range of the sheepdog $R_{\pi\beta}$ such that the magnitude of this force is inversely proportional to the distance between β and π_i. The parameters $W_{\pi\Lambda}, W_{\pi\pi}$, and $W_{\pi\beta}$ are the weights determining the influence of the three forces on the total force. Table 1 lists the setting of the sheep parameters.

5.2 Sheepdog Behaviour

Given the problem statement specified in Section 4, the sheepdog is required to learn an appropriate behaviour to successfully and efficiently complete the shepherding task. Two performance metrics are used in this work to evaluate the sheepdog behaviour: success rate and task completion time. Past studies describe the sheepdog behaviour in terms of two key behaviours: collecting and driving [16, 33, 39]. Collecting is about guiding scattered sheep agents towards the center of mass of the swarm; whereas driving is about guiding a cluster of sheep towards the goal. The two behaviours are inherently similar in terms of the *pushing* effect that causes the movement of an entity (single sheep or a cluster of sheep) from its initial position P_a towards a target position P_b. Thus, using a learning lens, it does not seem very useful to consider the collect and drive behaviours as two distinct sub-tasks. Alternatively, the two behaviours can be considered a single sub-task with different input parameters. Given this formulation, the shepherding task can be divided into a hierarchy of two sub-tasks. The first sub-task is *pushing* an entity from P_a to P_b and is concerned with generating the low-level actions for the sheepdog movement. The second sub-task is *mode selection* which is concerned with selecting a high-level action (collect or drive) such that the parameters of the first sub-task, P_a and P_b, are set accordingly. The rest of this section presents the details of how RL is used for learning these two sub-tasks.

5.2.1 The pushing sub-task. The scenarios used for learning the *pushing* sub-task include a single sheep acting as the entity to be pushed, a sheepdog, and a goal location which are randomly initialised at positions P_a, P_β, and P_b, respectively. To push an entity from position P_a to position P_b, the sheepdog needs to receive as input P_a, P_b, and its own position P_β. These variables form the state space of the *pushing* sub-task S^{push} as follows:

$$S^{push} = \{\frac{x_\beta - x_B}{L}, \frac{y_\beta - y_B}{L}, \frac{x_A - x_B}{L}, \frac{y_A - y_B}{L}\} \tag{3}$$

All variables are normalised by dividing them by the length of the environment L. The final states S_F^{push} are defined as those where the following condition holds:

$$||P_a - P_b|| < R_G \tag{4}$$

As P_a, P_b, and P_β can be any position in the environment, the initial states for the *pushing* sub-task can be any state except for the final states $S_I^{push} = S^{push} \setminus S_F^{push}$. The action space is defined in terms of the velocity of the sheepdog $A^{push} = \{v_x, v_y\}$ where v_x and $v_y \in (-1, 1)$ refer to the velocity components along the x and y axes, respectively. The agent is given a penalty of -0.06 for every time step spent in this sub-task and is given a reward of 15 when the condition in Equation 4 is met.

Curriculum learning is used to accelerate the learning of this sub-task by starting with simplified *pushing* scenarios then gradually moving towards more complex ones. The complexity of a *pushing* scenario is controlled by manipulating the distribution of S_I^{push} with respect to S_F^{push}. This is operationalised by limiting $D_{a,b}^0$ and $D_{a,\beta}^0$ which are the initial distances between P_a and P_b and between P_a and P_β, respectively. In the beginning of the learning, $D_{a,b}^0$ and $D_{a,\beta}^0$ are limited to very small values. Then $D_{a,b}^0$ and $D_{a,\beta}^0$ increase over time based on the agent's performance in the last ω training iterations.

Algorithm 1 shows the steps used for training the *pushing* sub-task. $D_{a,b}^0$ and $D_{a,\beta}^0$ are first initialised to make P_a at most d_0 units away from both the shepherd's influence range and the goal region (lines 2-3). Agent performance is monitored using the *successHistory* variable which is used to calculate the success rate in the previous ω training episodes (line 4). Before each training episode, if the

success rate is found to be greater than a performance threshold θ, $D^0_{a,b}$ and $D^0_{a,\beta}$ are increased by δ (lines 6-9). A new scenario is then randomly initialised and used for training the RL agent (lines 10-11). The result of the training scenario is logged in the *successHistory* variable (line 12-16).

Algorithm 1 Training for the pushing sub-task.

1: **INPUT** ω , L, T_{max}, d_0, δ, θ, $max\_iterations$
2: $D^0_{a,\beta} = R_{\beta\pi} + d_0$
3: $D^0_{a,b} = R_G + d_0$
4: successHistory = zeros(ω)
5: **for** $itr = 1, 2, .., max\_iterations$ **do**
6: **if** mean(successHistory) > θ **then**
7: $D^0_{a,\beta} = D^0_{a,\beta} + \delta$
8: $D^0_{a,b} = D^0_{a,b} + \delta$
9: **end if**
10: scenario = initialise_scenario($D^0_{a,\beta}$, $D^0_{a,b}$)
11: arrived = trainRL(scenario)
12: **if** arrived **then**
13: successHistory[$mod(itr, \omega)$] =1
14: **else**
15: successHistory[$mod(itr, \omega)$] =0
16: **end if**
17: **end for**

Deep deterministic policy gradient (DDPG) [25] is used as the RL agent for the *pushing* sub-task. The actor and critic networks have four hidden layers, each with 64 neurons. The discounting factor is set to 0.98 and the learning rate is 0.002. The exploration model used is Gaussian noise with variance = 0.005. Matlab's Reinforcement Learning Toolbox has been used for implementing the DDPG agent.

5.2.2 Mode selection sub-task. The scenarios used for the *mode selection* sub-task involve a swarm of N sheep randomly initialised in a square of length $\frac{L}{2}$ and center (x_c, y_c) such that $x_c \in (\frac{L}{4}, \frac{3L}{4})$ and $y_c \in (\frac{L}{4}, \frac{3L}{4})$. Existing shepherding algorithms assume that a sheepdog needs to first collect all sheep together and then drive the clustered swarm towards the goal location. The proposed approach also incorporate these two modes of operation (i.e. *collect* and *drive*), however no strict rule is imposed on when each mode should be activated. Instead, we leave it to the learning algorithm to find when each of these modes should be put into operation. When the *collect* mode is activated, the shepherd locates the furthest sheep and applies the *push* action to move this sheep towards the largest sheep cluster. On the other hand, when the *drive* mode is activated, the shepherd finds the largest sheep cluster and applies the *push* action to move it towards the goal.

The *mode selection* sub-task is modelled as a semi-Markov Decision Process [40] with two temporally abstracted actions $A^{mode} = \{collect, drive\}$. When the action *collect* is selected, the *pushing* sub-task is executed with $P_a = (x_{\pi^*}, y_{\pi^*})$ and $P_b = (x_{LCCM}, y_{LCCM})$, such that π^* is the furthest sheep from the GCM of the swarm and *LCCM* is the centre of mass of the largest sheep cluster. As for the *drive* action, it is performed by executing the *pushing* sub-task with $P_a = (x_{LCCM}, y_{LCCM})$ and $P_b = (x_G, y_G)$. The duration for a single execution of the *collect* or *drive* action is denoted by τ and can range between 1 and τ_{max}. An action terminates when the condition in Equation 4 is met or after τ_{max} steps from its start. The agent gets a reward of 30 when all sheep are at the goal area, otherwise it gets a small negative reward of -0.005τ at the end of each action execution. The state space for the *mode selection* sub-task comprises state variables reflecting the mean and standard deviation of the number of neighbours per sheep, the distance between the sheepdog and the furthest sheep to the swarm, and the average distances between each of: the sheep and the goal, the sheep and the sheepdog, and the sheep and the furthest sheep to the swarm. The state space S^{mode} is formed as follows:

$$S^{mode} = \{\frac{\mu_{Neighbours}}{N-1}, \frac{\sigma_{Neighbours}}{N-1}, \frac{1}{L}D_{\pi^*,\beta}, \frac{1}{L}\sum_{i=1..N}\frac{D_{\pi_i,G}}{N}, \frac{1}{L}\sum_{i=1..N}\frac{D_{\pi_i,\beta}}{N}, \frac{1}{L}\sum_{i=1..N}\frac{D_{\pi_i,\pi^*}}{N}\} \quad (5)$$

such that $\mu_{Neighbours}$ and $\sigma_{Neighbours}$ are the mean and standard deviation of the number of neighbours per sheep, and $D_{x,y}$ is the Euclidean distance between x and y. The term $\frac{1}{N-1}$ is used to normalise the first two state variables, whereas the term $\frac{1}{L}$ is used to normalise the other four variables.

Curriculum learning is employed for this sub-task by designing a simplified environment where the learning starts. The simplified environment is created by using a state space that is smaller than the state space of the main task by reducing dimensions of the environment and the number of sheep. Once the performance in the simplified environment is satisfactory, learning continues in the main environment. The benefits of using the simplified environment is two fold. First, the state space is greatly reduced which lowers the learning complexity. Second, the average episode length is reduced which mitigates the negative effects of the sparse reward on learning efficiency [29].

Proximal Policy Optimization (PPO) [37] is used for learning this sub-task. The actor and critic networks have two hidden layers, each with 64 neurons. The discounting factor is set to 0.999 and the learning rate is 0.005. Matlab's Reinforcement Learning Toolbox has been used for implementing the PPO agent.

6 EXPERIMENTAL RESULTS

This section presents the details of the training experiments conducted for the two sub-tasks defined in Section 5.2. The resulting sheepdog behaviour is then evaluated to assess its efficacy.

6.1 Pushing Sub-task

6.1.1 Training: The parameters used for the *pushing* sub-task are listed in Table 2. For comparison purposes, another set of non-curriculum training experiments have been conducted to train the DDPG agent on the *pushing* sub-task. For the non-curriculum training, the positions of the sheep, sheepdog, and goal are randomly initialised in the environment with no limits on the initial positioning of the sheep with respect to the sheepdog and the goal. Fifteen training experiments were run for both the curriculum and the non-curriculum approaches where each training experiment lasted for 10,000 training episodes.

Parameter	T_{max}	L	N	R_G	d_0	θ	ω	δ
Value	250	60	1	2.2	1.0	80%	30	0.3

Table 2: Training parameters for pushing sub-task. The last four parameters are used only by the curriculum-based training experiments.

The episode rewards obtained in different episodes of the curriculum approach cannot be directly compared due to the differences in episode complexity. As the complexity changes based on $D^0_{a.\beta}$ and $D^0_{a,b}$, the reward obtained in each episode is normalised using the following equation:

$$\hat{R} = (R + |R_{min}|) * \frac{D_{a,\beta} + D_{a,b}}{2 * D_{max}} \quad (6)$$

such that R is the episode reward, R_{min} is the minimum value of episode reward, and D_{max} is the maximum possible distance between any two points in the environment, which is calculated as $\sqrt{2}L$. The top part of Figure 1 shows the average normalised reward over time. It can be seen that the average normalised reward obtained by the curriculum learning approach increased notably overtime. Meanwhile, the non-curriculum learning had very rare successful episodes that it could not learn at all. The bottom part of Figure 1 shows average scenario complexity (calculated as $D_{a,\beta} + D_{a,b}$) over time. For the curriculum learning approach, scenario complexity increased at a high rate in early episodes as the DDPG agent was relatively fast at reaching the performance threshold θ required to move to higher levels of complexity. As learning progresses, the DDPG agent needed more time to find a policy that meets the required threshold, hence the slower increase in complexity.

6.1.2 Evaluation: After completing the training of the *pushing* sub-task, the resulting policy from each training experiment has been evaluated under 100 scenarios leading to 1500 evaluation scenarios. In these evaluation scenarios, $D^0_{a,\beta}$ and $D^0_{a,b}$ were set to their maximum value of D_{max}. The results demonstrate the high performance achieved in terms of the success rate (mean = 91.3%, SD = 4.8) and the episode length (mean = 119.1, SD = 58.9 steps). Out of these policies, three achieved a success rate of 97% thus has been used as input when learning the *mode selection* sub-task.

6.2 Mode Selection Sub-task

6.2.1 Training: After learning the skill of pushing a sheep from its location to a target location, the sheepdog starts learning the next sub-task of selecting between collecting scattered sheep or driving the biggest cluster to the goal area. A separate PPO agent is trained on the *mode selection* sub-task. To use the curriculum approach within this task, a simplified task has been designed using the parameters listed in Table 3. The agent was first trained in the simplified environment for 500 episodes then continued training in the main environment for 10,000 episodes. For comparison, another set of non-curriculum training experiments have been conducted by starting the training directly in the main environment. Figure 2 shows the average episode reward obtained by both the curriculum and the non-curriculum approaches in the main environment. The

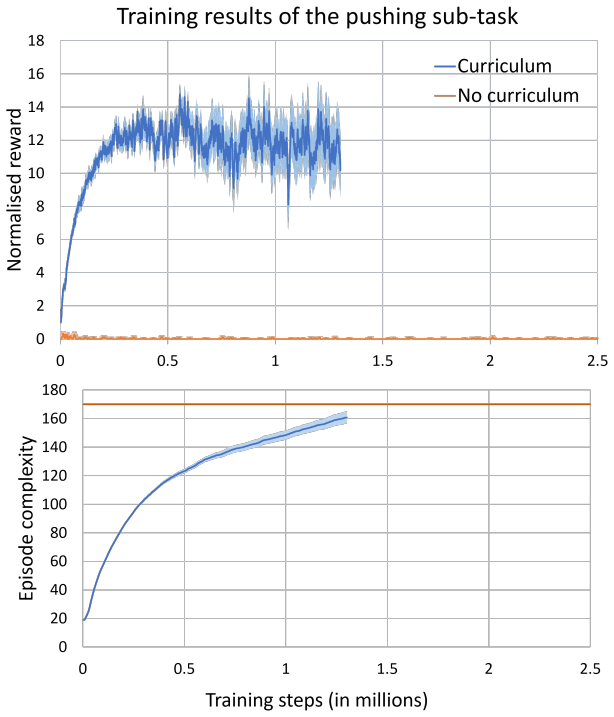

Figure 1: Training results of the pushing sub-task. Thick lines represent the mean and shades represent the standard error.

Task	L	N	T_{max}	R_G	τ_{max}
Simplified	50	15	150	8.7	10
Main	200	50	350	15.8	10

Table 3: Parameters for the simplified and main tasks used for learning the mode selection sub-task. The simplified task is only used in the curriculum approach.

non-curriculum based approach was slightly faster than the curriculum approach in this environment even though the curriculum approach was pre-trained in the simplified environment.

6.2.2 Evaluation: After training, the learnt policies are evaluated to assess their performance and to gain insights into the reason behind the lack of benefit of the curriculum used for the *mode selection* sub-task. To do this, the policy resulting from each training experiment has been evaluated under 100 randomly initialised scenarios. Figure 3 shows that the policies obtained from the curriculum and non-curriculum approaches achieve fairly similar performance. The success rate achieved by the curriculum and non-curriculum approaches is (mean = 94.7%, SD =4.04) and (mean = 95.9%, SD =2.12), whereas the task time recorded is (mean = 1710.4 , SD =581.4 time steps) and (mean = 1710.7 , SD =536.4 time steps); respectively.

To understand why the curriculum was not effective in speeding up the learning of the *mode selection* sub-task, the policies obtained in the simplified environment are compared to those from the main

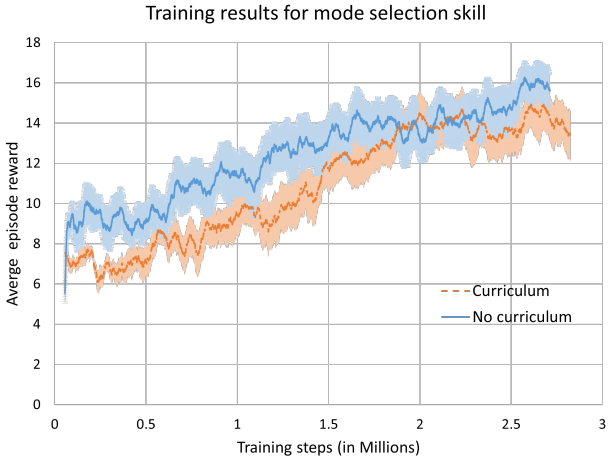

Figure 2: Average training episodes for the mode selection sub-task when trained with and without a curriculum.

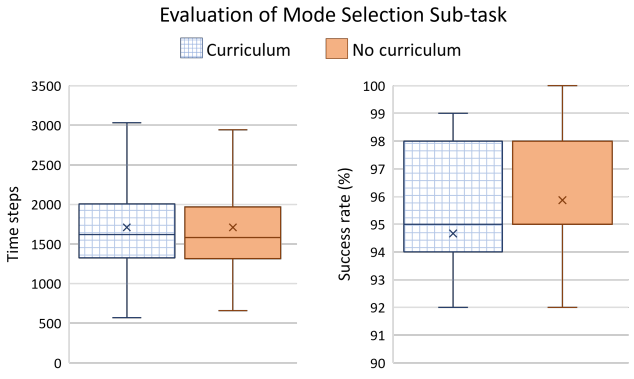

Figure 3: Average success rate and episode length achieved by the mode selection sub-task in the main environment.

environment. The numbers of collect and drive actions are logged for each evaluation scenario. Then, scenario time is divided evenly into five intervals to inspect trends in the policies' behaviour over time. Figure 4 shows the average percentage of collect and drive actions in the simplified task (upper part) and in the main task, both when trained with a curriculum (left bottom part) and without a curriculum (right bottom part). In the three sub-figures, the length of the bars decreases as tasks finish execution over time. Policies from the simplified tasks rarely uses the *collect* action which means that they drive sheep sub-clusters directly to the goal. Due to the small size of the simplified environment, this policy is very useful as the distance between the sheep and the goal is relatively small. This makes pushing a sheep sub-cluster to the fixed position of the goal more efficient than pushing the sub-cluster towards another moving sub-cluster.

However, when transferring this policy to the main environment, it did not work well due to the large distances between the

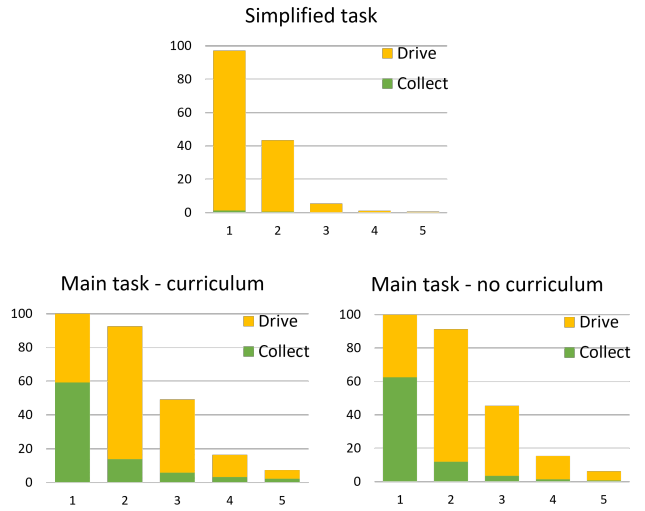

Figure 4: Average percentage of collect and drive actions executed by the sheepdog over five intervals of scenario time. The length of the bars decreases as more and more tasks finish execution over time.

sheep and the goal. In this case, collecting the sheep together before driving them toward the goal resulted in better performance. Policies obtained from the curriculum and non-curriculum training experiments exhibit very similar behaviours where about 60% of the actions executed in the first interval are *collect*. But then, the percentage of *collect* actions diminish dramatically in favor of *drive* actions. It is worth noting though that even in the first time interval, *drive* actions are executed for about 40% of the time. By visually inspecting the behaviour, it was shown that the sheepdog would sometimes drive sheep sub-clusters without collecting them when at least one sub-cluster is very close to the goal or when sub-clusters are very distant from each other. A visual representation of the sheepdog behaviour in different scenarios is provided in a video [1].

7 DISCUSSION

The results obtained in Section 6 indicate a number of interesting findings. First, the use of curriculum learning was necessary to enable the efficient learning of the *pushing* sub-task. This finding confirms previous results in the literature regarding the advantages of curriculum learning in goal-oriented tasks with continuous state and action spaces [12]. Without curriculum, the DDPG agent had very rare successes, which were insufficient to discover a useful policy for the *pushing* sub-task. In contrast, using a curriculum that starts with easy tasks where the sheepdog, sheep, and the goal are very close to each other meant that the initial states of the MDP are very close to its goal states. The likelihood of success in easy tasks was sufficiently high that the agent could utilise the received reward to find a useful policy. The progressive increase in task complexity over time ensured that the next task is never too hard given the current capability of the agent.

[1] https://youtu.be/vvyVTbaXzPk

Unlike the *pushing* sub-task, the curriculum used to learn the *mode selection* sub-task did not improve the learning experience. The post-learning analyses performed in Section 6.2 show that the policies learnt in the simplified environment differed significantly from those for the main environment. In other words, though both environments are different instances of the same task, policies that work best for the simplified environment are not scalable enough to maintain their superiority in the main environment. This finding highlights a key advantage of the RL algorithm used for this sub-task, but also raises a question about how to best design a curriculum for learning. The advantage is that the RL algorithm shows a high level of adaptability to the environment such that the sheepdog behaviour changed notably to maintain its high performance across different environments. This links back to the main motivation for designing a learning-based shepherding algorithm that can adapt to changes in the environment, as discussed in Section 2. However, the question that needs further investigation relates to the design of a curriculum to accelerate learning. Particularly, how to design a simplified task for the curriculum such that optimal policies in the simplified setting share some notable aspects with those in the main setting.

Another interesting finding is that the sheepdog behaviour following the learnt policies have some differences to the sheepdog behaviour specified by the existing rule-based algorithms. Rule-based algorithms specify that the sheep must be collected in one cluster before driving them towards the goal. If any fragments occur during driving, the shepherd has to recollect the sheep again before continuing the driving. In contrast, the policies learnt by the PPO agent exhibit a more diverse set of behaviours. They tend to collect the sheep first in most cases, unless the sheep are very close to the goal or sheep clusters are very far from each other. This is another important advantage of RL-based shepherding algorithms as several aspects of the state of the sheep can be used as input when deciding on the best action.

8 CONCLUSION AND FUTURE WORK

The shepherding problem is highly complex due to the non-linearity of swarm behaviour. Previous studies using RL for shepherding relied on external rule-based algorithms to calculate the waypoints for the shepherd to simplify learning. However, this compromises the advantages of learning by limiting the space of behaviours the sheepdog can learn. In this work, we address the complexity of the shepherding problem by using a curriculum learning approach. Task decomposition is used by decomposing the shepherding problem into a hierarchy of two sub-tasks: (1) push an agent from its location to a target location and (2) select between collecting scattered agents or driving the largest cluster of agents toward the goal. A simple-to-complex curriculum was designed to expedite the learning of each sub-task. The evaluation results demonstrate the effectiveness of the proposed approach as measured by the average success rate of 95.6% and task time of 1017 steps. The results show that curriculum learning was effective for the *pushing* sub-task where the non-curriculum approach was unsuccessful. On the contrary, the curriculum used for the second sub-task has not improved learning. The analyses showed that the policies learnt in the simplified environment are not sufficiently scalable to maintain their superiority in the main environment. Future work should investigate how to design simplified tasks such that learning in the simplified setting leads to speeding up learning in the main target task.

Our analyses also showed that the learnt policies exhibit diverse behaviours based on the status of the sheep. Unlike rule-based approaches that require all the sheep to be collected in one cluster before being driven to the goal, the learnt policies allow the shepherd sometimes to drive sub-clusters to the goal when this is more efficient. For rule-based algorithms, the performance of the sheepdog drops as the size of the sheep swarm increases [39] due to the strict requirement on driving all the sheep together. By allowing adaptive shepherding behaviours, similar to those exhibited by the learnt policies as discussed in Section 6.2, a single agent can handle large swarms by driving smaller sub-clusters separately. In future work, we will aim to investigate how a single sheepdog can learn efficient policies to shepherd swarms of different sizes and characteristics.

ACKNOWLEDGMENTS

This work was funded by the Australian Research Council Discovery Grant number DP200101211.

REFERENCES

[1] Hussein Abbass, Eleni Petraki, Aya Hussein, Finlay McCall, and Sondoss Elsawah. 2021. A model of symbiomemesis: machine education and communication as pillars for human-autonomy symbiosis. *Philosophical Transactions of the Royal Society A* 379, 2207 (2021), 20200364. https://doi.org/10.1098/rsta.2020.0364

[2] Hussein A Abbass and Robert A Hunjet. 2021. Smart shepherding: Towards transparent artificial intelligence enabled human-swarm teams. *Shepherding UxVs for Human-Swarm Teaming: An Artificial Intelligence Approach to Unmanned X Vehicles* (2021), 1–28.

[3] Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in Neural Information Processing Systems* (2015).

[4] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. 41–48.

[5] Andrea Bonarini, Alessandro Lazaric, and Marcello Restelli. 2007. Reinforcement learning in complex environments through multiple adaptive partitions. In *Congress of the Italian Association for Artificial Intelligence*. Springer, 531–542.

[6] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. 2013. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence* 7, 1 (January 2013), 1–41. https://doi.org/10.1007/s11721-012-0075-2

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

[8] Fei Dai, Ming Chen, Xianglin Wei, and Huibin Wang. 2019. Swarm Intelligence-Inspired Autonomous Flocking Control in UAV Networks. *IEEE Access* 7 (2019), 61786–61796. https://doi.org/10.1109/ACCESS.2019.2916004

[9] Heba El-Fiqi, Benjamin Campbell, Saber Elsayed, Anthony Perry, Hemant Kumar Singh, Robert Hunjet, and Hussein A Abbass. 2020. The Limits of Reactive Shepherding Approaches for Swarm Guidance. *IEEE Access* 8 (2020), 214658–214671.

[10] Jeffrey L Elman. 1993. Learning and development in neural networks: The importance of starting small. *Cognition* 48, 1 (1993), 71–99.

[11] Mark Evered, Peter Burling, Mark Trotter, et al. 2014. An investigation of predator response in robotic herding of sheep. *International Proceedings of Chemical, Biological and Environmental Engineering* 63 (2014), 49–54.

[12] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse curriculum generation for reinforcement learning. In *Conference on robot learning*. PMLR, 482–495.

[13] Daniel Y. Fu, Emily S. Wang, Peter M. Krafft, and Barbara J. Grosz. 2018. Influencing Flock Formation in Low-Density Settings. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems* (Stockholm, Sweden) *(AAMAS '18)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 1613–1621.

[14] Teruo Fujii, Yoshikazu Arai, Hajime Asama, and Isao Endo. 1998. Multilayered reinforcement learning for complicated collision avoidance problems. In *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, Vol. 3. IEEE, 2186–2191.

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

[16] Clark Kendrick Go, Bryan Lao, Junichiro Yoshimoto, and Kazushi Ikeda. 2016. A reinforcement learning approach to the shepherding task using SARSA. In *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 3833–3836.

[17] Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. 2017. Automated curriculum learning for neural networks. In *international conference on machine learning*. PMLR, 1311–1320.

[18] Vijaykumar Gullapalli and Andrew G Barto. 1992. Shaping as a method for accelerating reinforcement learning. In *Proceedings of the 1992 IEEE international symposium on intelligent control*. IEEE, 554–559.

[19] Tobias Helms, Steffen Mentel, and Adelinde Uhrmacher. 2016. Dynamic State Space Partitioning for Adaptive Simulation Algorithms. In *Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools*. 149–152.

[20] Adam J Hepworth, Kate J Yaxley, Daniel P Baxter, Keith F Joiner, and Hussein Abbass. 2020. Tracking Footprints in a Swarm: Information-Theoretic and Spatial Centre of Influence Measures. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2217–2224.

[21] Aya Hussein, Sondoss Elsawah, Eleni Petraki, and Hussein A Abbass. 2021. A machine education approach to swarm decision-making in best-of-n problems. *Swarm Intelligence* (2021). https://doi.org/10.1007/s11721-021-00206-5

[22] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G Hauptmann. 2015. Self-paced curriculum learning. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[23] Jyh-Ming Lien, O Burchan Bayazit, Ross T Sowell, Samuel Rodriguez, and Nancy M Amato. 2004. Shepherding behaviors. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, Vol. 4. IEEE, 4159–4164.

[24] Jyh-Ming Lien and Emlyn Pratt. 2009. Interactive Planning for Shepherd Motion.. In *AAAI Spring Symposium: Agents that Learn from Human Teachers*. 95–102.

[25] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971* (2015).

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

[27] Reem E. Mohamed, Saber Elsayed, Robert Hunjet, and Hussein Abbass. 2021. A Graph-based Approach for Shepherding Swarms with Limited Sensing Range. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. 2315–2322. https://doi.org/10.1109/CEC45853.2021.9504706

[28] Ofir Nachum, Shixiang Shane Gu, Honglak Lee, and Sergey Levine. 2018. Data-Efficient Hierarchical Reinforcement Learning. *Advances in Neural Information Processing Systems* 31 (2018), 3303–3313.

[29] Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 6292–6299. https://doi.org/10.1109/ICRA.2018.8463162

[30] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. 2020. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research* 21 (2020), 1–50.

[31] Sanmit Narvekar, Jivko Sinapov, Matteo Leonetti, and Peter Stone. 2016. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. 566–574.

[32] Sanmit Narvekar and Peter Stone. 2019. Learning Curriculum Policies for Reinforcement Learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems* (Montreal QC, Canada) *(AAMAS '19)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 25–33.

[33] Hung The Nguyen, Tung Nguyen, Vu Phi Tran, Matthew Garratt, Kathryn Kasmarik, Sreenatha Anavatti, Michael Barlow, and Hussein A Abbass. 2020. Continuous Deep Hierarchical Reinforcement Learning for Ground-Air Swarm Shepherding. *arXiv preprint arXiv:2004.11543* (2020).

[34] Bei Peng, James MacGlashan, Robert Loftin, Michael L Littman, David L Roberts, and Matthew E Taylor. 2018. Curriculum design for machine learners in sequential decision tasks. *IEEE Transactions on Emerging Topics in Computational Intelligence* 2, 4 (2018), 268–277.

[35] Zhipeng Ren, Daoyi Dong, Huaxiong Li, and Chunlin Chen. 2018. Self-paced prioritized curriculum learning with coverage penalty in deep reinforcement learning. *IEEE transactions on neural networks and learning systems* 29, 6 (2018), 2216–2226.

[36] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized Experience Replay. In *International Conference on Learning Representations (ICLR)*.

[37] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[38] Christopher Simpkins and Charles Isbell. 2019. Composable modular reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 33. 4975–4982.

[39] Daniel Strömbom, Richard P Mann, Alan M Wilson, Stephen Hailes, A Jennifer Morton, David JT Sumpter, and Andrew J King. 2014. Solving the shepherding problem: heuristics for herding autonomous, interacting agents. *Journal of the royal society interface* 11, 100 (2014), 20140719.

[40] Richard S Sutton, Doina Precup, and Satinder Singh. 1999. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112, 1-2 (1999), 181–211.

[41] Matthew E Taylor and Peter Stone. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10, 7 (2009).

[42] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor. 2017. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[43] Yusuke Tsunoda, Yuichiro Sueoka, and Koichi Osuka. 2017. On statistical analysis for shepherd guidance system. In *2017 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 1246–1251.

[44] Yusuke Tsunoda, Yuichiro Sueoka, Teruyo Wada, and Koichi Osuka. 2020. Sheepdog-type robot navigation: Experimental verification based on a linear model. In *2020 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 1144–1149.

[45] Richard Vaughan, Neil Sumpter, Andy Frost, and Stephen Cameron. 1998. Robot sheepdog project achieves automatic flock control. In *Proc. Fifth International Conference on the Simulation of Adaptive Behaviour*, Vol. 489. 493.

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

[47] Kate J Yaxley, Keith F Joiner, and Hussein Abbass. 2021. Drone approach parameters leading to lower stress sheep flocking and movement: sky shepherding. *Scientific reports* 11, 1 (2021), 1–9.