AN ADVICE MECHANISM FOR HETEROGENEOUS ROBOT TEAMS

Steven Daniluk* and M. Reza Emami*,**

Abstract

This paper presents an advice mechanism compatible with heterogeneous advisers that incorporates advice into the advisee's policy via a method guaranteeing convergence to an optimal policy. Further, the mechanism has the capability to use multiple advisers at each time step and decide when advice should be requested and accepted, such that the use of advice decreases over time. Experiments are formed with a simulated team of heterogeneous robots performing a foraging task. We show that the proposed mechanism can provide a performance improvement for homogeneous and heterogeneous robot teams, and the use of advice decreases over time.

Key Words

Robot teams, robot learning, multi-robot coordination, advice mechanism

1. Introduction

Autonomous robots require sophisticated behaviour that is often achieved with performance enhancement algorithms that learn the appropriate behaviour through experience. By interacting with its environment, a robot can learn through trial-and-error to develop a policy that maps states to the appropriate actions. Applications have progressed from simplistic objectives, such as balancing an inverted pendulum [1], to more complex actions, such as groups of robots working together towards a specific goal [2], [3], [4], path planning for mobile robots [5], or enabling robotic arms to manipulate unknown objects [6], [7].

Despite the capabilities of such performance enhancement techniques, mostly based on the reinforcement learning algorithms, one shortcoming is the inevitable need to explore a large number of possible states to learn the most viable actions. Thus, as the task complexity grows, *i.e.*, the number of possible states and actions increases, so does the amount of experience required to learn an optimal policy for each task. Such exploration is not only a computational burden but also costly and even hazardous when learning is performed with physical hardware that often requires human involvement and is subject to wear. Thus, the ability to learn from a small number of trials is crucial.

Much work has been focussed on expediting the rate at which a robot learns their task, such as reducing the dimensionality of the state space [8], utilizing function approximation methods [9], and applying reward shaping techniques [10]. (See [11] and [12] for a survey on the use of reinforcement learning in robotics.) An additional approach to accelerating a robot's learning is to utilize knowledge from an *expert*. Doing so can guide the learning process and reduce the amount of potentially costly exploration required [13]. The expert knowledge is referred to as *advice*, where the expert can be a human being or another robot. Advice for an autonomous robot is analogous to advice for human beings, where the knowledge of others is leveraged to accelerate learning. Much like humans, a robot can have many advisers of varying quality or capabilities, and must effectively identify and utilize the potentially more valuable knowledge of an adviser [14]. The role of advice is even more crucial in robot teams, where many robots may be learning to perform similar tasks in parallel. Enabling teammates to exchange information may greatly improve the performance of the entire team [15]. However, care must be taken to ensure that the influence of advice will not prevent a robot from learning an optimal policy.

To date, the existing advice mechanisms have mainly relied on *ad hoc* rules for deciding who the adviser(s) should be, and if their advice can be utilized. A disadvantage of such heuristic approaches is that the mechanism's effectiveness is limited by the designer's assumptions about the application and their ability to generate the appropriate rules. Such rules can include assigning *a priori* which agents can be advisers, when their advice can be used, and limiting how long the advice will be available to ensure a stationary policy is reached. Therefore, this approach not only requires additional testing for the mechanism to work effectively but also limits the mechanism's adaptability to different applications.

In a realistic robot team, the robots may have different capabilities, making the team heterogeneous, and each robot may be in a different stage of learning. In this case, the capability and relevance of advisers may be unknown and change over time as each agent's learning progresses.

* University of Toronto Institute for Aerospace Studies, 4925 Dufferin Street, Toronto, Ontario, Canada M3H 5T6; e-mail: steven.daniluk@mail.utoronto.ca, emami@utias.utoronto.ca

** Division of Space Engineering, Luleå University of Technology, Luleå, Sweden S-981 28; e-mail: reza.emami@ltu.se
Corresponding author: M. Reza Emami

Recommended by Dr. Mohammad Biglarbegian
(DOI: 10.2316/J.2020.206-0166)

Thus, attempting to handle such a range of behaviour with static rules inevitably compromises the mechanism's effectiveness. Further, designing and testing rules for each variation of agents or applications are often impractical.

To advance the benefits of advice for robot teams, a mechanism is needed, which is capable of extracting the appropriate behaviour on its own. The purpose of this paper is to introduce a reinforcement-learning-based advice mechanism that

1. does not require advisers with full knowledge of the task;
2. is compatible with advisers of varying skill levels and relevance;
3. guarantees convergence to an optimal policy;
4. diminishes the influence and usage of advice over time.

Section 2 of the paper provides an overview of reinforcement learning and previous approaches to advice. Section 3 presents a formal definition of the proposed advice mechanism and proves the convergence to an optimal policy for an agent utilizing the proposed advice mechanism. In Section 4, the case study and experiments used to demonstrate the mechanism are presented, and the results from the experiments are discussed in Section 5. Finally, some concluding remarks are made in Section 6.

2. Background and Related Work

2.1 Reinforcement Learning

For an autonomous agent that performs actions, a, in states, s, and receives a reward, R, it must learn an appropriate mapping between states and actions. The purpose of reinforcement learning is to develop such mapping, called the policy π, that will maximize the expected reward of the agent over its lifetime. The reinforcement learning problem can be formalized as a Markov Decision Process (MDP), which consists of a set of states S, actions A, transition probabilities between states $T(s', a, a)$, and rewards $R(s, a, s')$.

The value of a particular state can be estimated by the cumulative expected reward for the current and future states (given the Markov property, the expected value of a state does not depend on any previous states). The Bellman equation (1) provides the expected value of a state, as a recursive formulation depending on the value of the succeeding state [16]. The factor $\gamma \in (0, 1)$ discounts the value of future states, giving a lower value to the states further into the future:

$$\begin{aligned} v_\pi(s) &= \mathbf{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s \right] \\ &= \sum_a \pi(a|s) \sum_{s'} T(s', a, s) \left[R(s, a, s') + \gamma v_\pi(s') \right] \end{aligned} \quad (1)$$

The Bellman optimality equation (2) represents the value of state s, given that every subsequent action taken is the optimal action:

$$v_*(s) = \max_{a \in A(s)} \sum_{s'} T(s', a, s) \left[R(s, a, s') + \gamma v_*(s') \right] \quad (2)$$

An alternative formulation of the Bellman optimality equation is to estimate the value of a state–action pair, $q_*(s, a)$, as given in (3). The benefit of such a formulation is that it is not required to know the transition function $T(s, a, s)$ and is hence referred to as model-free reinforcement learning:

$$q_*(s, a) = \sum_{s'} T(s', a, s) \left[R(s, a, s') + \gamma \max_{a'} q_*(s', a') \right] \quad (3)$$

A popular approach to estimate the state–action value function $q_*(s, a)$ is the temporal difference method, which iteratively updates the value of the previous state as the new state and the corresponding reward are experienced. One such temporal difference method is Q-learning, which uses the update rule given by (4) [17]. In (4), $\alpha \in (0, 1)$ is the learning rate, which approaches 0 as t approaches ∞, such that (4) will arrive at a locally optimal policy [17].

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t \Big(& R_t(s_t, a_t, s_{t+1}) \\ & + \gamma \max_{a \in A} Q_t(s_{t+1}, a) - Q_t(s_t, a_t) \Big) \end{aligned} \quad (4)$$

An alternative to Q-learning is SARSA(0) [18], which updates the values of $Q(s, a)$ based on the selected action in the next state, as given by (5):

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t (& R_t(s_t, a_t, s_{t+1}) \\ & + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t)) \end{aligned} \quad (5)$$

The policy for selecting actions, π, can either be deterministic, *e.g.*, greedily selecting the most optimal action, or probabilistic, which will form a distribution over all possible actions. The motivation for a probabilistic policy is that it provides a balance between exploration and exploitation of the current value function. An optimal policy depends on the value function being determined, which by (2) requires every state to be visited. Utilizing a probabilistic policy enables new states to be explored.

2.2 Related Work

The concept of advice for a computer program has existed for nearly 60 years [19]. Much of the recent research into advice has been focussed on uses for reinforcement learning algorithms, due to their popularity. In general, advice within a reinforcement learning context can take three forms:

1. incorporating prior information into the agent's policy;
2. sharing a set of experiences; and
3. recommending an action.

Incorporating prior information into the agent's policy is often referred to as *transfer learning* and is achieved by either imposing human generated rules onto the policy, or creating a mapping from a previously learned policy for a related task [20]. Transfer learning can create a proper jumpstart in the learning process [21] and avoid much of the

early exploration costs. However, this requires extensive prior knowledge about the task and agents to generate the appropriate mappings. Since there is a cost of human involvement to generate the mappings for each application, transfer learning is most applicable for tasks and agents that are not frequently subject to change, such as RoboCup competitions where it is commonly applied [22].

The second form of advice is sharing experiences between agents. After performing what is deemed as a successful series of actions, the adviser agent will share with the advisee the set of $\langle S_t, S_{t+1}, A_t, R_t \rangle$ tuples that led to success. The advisee then replays these experiences with the intention that they will learn to replicate this behaviour.

In [23], a theoretical analysis of exchanging experiences is performed, assuming that agents simply observe their adviser and replay each of the observed experiences as if they were their own. Further, it is shown that the use of experiences reduces the search time for an optimal policy to be independent of the state space size and only dependent on the length of the optimal solution, *i.e.*, the number of consecutive actions.

In [24], a series of successful experiences from a knowledgeable advisor, *i.e.*, an agent with a learned policy or a human controller, are replayed by a learning agent. Such demonstrations enabled the agent to not only learn more quickly but also perform tasks that they were initially unable to learn on their own. As mentioned in [24], a limitation of advice in the form of sharing experiences is the potential for harmful over-training of the agents, where an agent too strongly prefers certain actions because it has been repeatedly taught to do so, resulting in poor generalization for the agent.

An important point to note about exchanging experiences as advice is that if there is any heterogeneity between agents, such as robots with different capabilities, the rewards and state transitions shared in the experience may not be attainable for the advisee. Without some method of determining if an adviser is relevant, this can potentially cause the advisee to learn an incorrect policy through harmful over-training.

The third form of advice is a recommendation about which action to perform. Advice in the form of action recommendations can be generated either by another agent or by human beings and can be treated as a binary decision, *i.e.*, accept or reject, or as a suggestion that acts to bias the agent's own decision.

The effect of biasing an agent's decision via a reward signal is analysed in [23], where in each state an agent can receive an additional reward from the adviser. As identified in [23], biasing the reward signal provides feedback only after the action is performed, making the expert's advice not immediately available when it is first needed. However, it is shown that if advice can be provided immediately when it is needed, then the search time for an optimal policy will be independent of the state space size and only dependent on the length of the optimal solution (identical to the use of experiences). One disadvantage of supplementing the reward signal as a form of action recommendation is the possibility for an agent to learn the incorrect policy. Similar to the use of experiences as advice, this is a problem that can arise with the use of heterogeneous agents.

In the Reinforcement and Advice-Taking Learning Environment (RATLE) mechanism [25], human observers are used to generate action suggestions and decide when they should be used. Advice is formed as IF–THEN or WHEN–REPEAT–UNTIL statements, which are then incorporated as inputs into a connectionist Q-learning agent. By incorporating advice directly into the agent's neural network, it can be further processed and potentially overruled by the agent's own policy. When compared to an agent that blindly obeys the same advice, the use of the RATLE mechanism results in faster learning. An important result from this work is that it demonstrates that the capability to further process advice can be beneficial.

The Skill Advice Guided Exploration (SAGE) mechanism [26] uses action suggestions from multiple advisers to bias the agent's policy. An agent selects an action based on its own action selection probability plus a weighted combination of all advisers' action selection probabilities. The weighting is determined by a shaping function that depends on the difference between the advisee's current state's utility and that of the adviser, as well as the frequency that each action is advised in the current state. Such an approach is attractive, because all advisers are always utilized, and their influence can vary depending on the weighting attributed to them. The SAGE mechanism has the important ability to reject bad advice, *i.e.*, an action suggestion with repeatedly low utility values, by attributing a very low weighting to it. However, the ability to reject advice happens on a per state basis. If the same adviser continues to provide poor advice they are not rejected entirely, but only in particular states in which bad advice is repeatedly provided. This means that areas of the state space where an adviser has little experience can be identified, but outright rejection of an advisor will be very slow. Further, the benefit of using all available advisers comes with high computational and communication costs.

Building upon [25], Preference Knowledge-Based Kernel Regression (Pref-KBKR) provides another approach to using humans for biasing an agent's decision [27]. The advice is a preference for one action over another in a particular state, implying that the value of actions should have a specific hierarchy. Similar to RATLE and SAGE, Pref-KBKR provides the possibility for an agent to reject the advice, if they strongly disagree with it, by expressing advice as a preference. The advice is pre-generated by a human and takes the form of a series of IF–THEN rules, similar to the RATLE mechanism. Experiments were performed using agents foraging for food while avoiding enemies. Different pieces of advice were tested, and it was found that for all variations of advice the agents with advice outperformed those without.

An alternative to biasing an agent's decision is to provide an agent access to the policy of an adviser. Advice Exchange [28] is a mechanism that uses other agents as advisers, where each agent temporarily uses the policy of their adviser to select the current action. At the beginning of each new learning epoch, the adviser of the system is selected to be the best performing agent (determined by the

average utility value of selected actions). At each instant, agents decide to accept or reject the advice by comparing their average utility and the utility of each possible action to the advisers. Advice Exchange along with SAGE is the only mechanisms that actively select their advisers. A limitation of Advice Exchange is that the choice of adviser only changes once per learning epoch. Additionally, since adviser selection is only dependent on the performance, it is possible for an irrelevant, yet well performing, advisor to be selected. The possibility of irrelevant adviser selection, combined with infrequent adviser selection, is likely to be problematic for agents with heterogeneous capabilities. Improvements to Advice Exchange have been made by incorporating a form of memory [29], but at the cost of additional rules and parameters for the mechanism.

3. A New Advice Mechanism

In this section, the formulation of a new advice mechanism with the four capabilities listed in Section 1 is presented. The first capability, to not require advisers with full knowledge of the task, emphasizes the need for utilizing partial knowledge from other agents. During the initial stages of learning, each agent's knowledge of the task will be sparse, but the team collectively may have near complete knowledge. The second capability of compatibility with advisers of varying skill levels and relevance is essential for heterogeneous agents. As the robots' capabilities become more diverse, it becomes more important for an advice mechanism to identify the usefulness of each adviser on its own. The third capability of convergence to an optimal policy not only provides a theoretically sound mechanism but also is of great importance to heterogeneous robot teams. When a discrepancy exists between robots, there is a risk that an adviser may guide an agent towards a different policy. Finally, the fourth capability for the influence and usage of advice to diminish over time serves to make the mechanism practical for real world applications, where each use of advice incurs a communication and computational cost.

The new advice mechanism, called Preference Advice, will be presented as follows: first, a method for characterizing the information possessed by an agent about a state, as well as incorporating information from an adviser, *i.e.* advice, into the advisee, will be shown. This will be followed by demonstrating that a reinforcement learning agent incorporating the advice of an adviser in the method shown will converge to a stationary and optimal policy. Finally, a formulation for an MDP will be presented that determines when an agent needs advice, and which advisers it should use.

3.1 Incorporating Advice

As the purpose of a reinforcement learning agent is to develop an optimal policy by learning the value of each state and action, how knowledgeable that agent is about a state should be dependent on these values. However, instead of directly using utility values which can vary greatly in magnitude between applications, it is advantageous to use the probability values for selecting each action, which are bounded on the interval [0, 1]. To assess how knowledgeable an agent is about a state via action selection probability values, it is assumed that a high certainty about which action(s) to select can be directly related to a high level of knowledge in the given state. Therefore, the greater the deviation from equal probability for all actions, the more knowledgeable an agent is likely to be.

To convert state–action values, $Q(s,a)$, to probability values for each action, $p(a_i)$, a Boltzmann distribution can be used as shown below:

$$p(a_i) = \frac{e^{\tau_t(s)Q(s,a_i)}}{\sum_{j=1}^{n} e^{\tau_t(s)Q(s,a_j)}}, \qquad i = 1, \ldots, n \tag{6}$$

where n is the number of actions and $\tau_t(s)$ is a temperature parameter controlling the smoothness of the distribution, which can vary between states. In the context of action selection probabilities, an agent can be said to have zero knowledge about its current state when no single action is considered to be more valuable than any other. If there are n possible actions, then this corresponds to each action having an equal probability value given by

$$p(a_i) = \epsilon = \frac{1}{n}; \qquad i = 1, \ldots, n \tag{7}$$

Conversely, an agent can be said to have the maximum possible amount of knowledge about its current state when all probabilities are attributed to a single action, represented by

$$p(a_i) = 1, \quad \text{and} \quad p(a_j) = 0 \quad \forall i \neq j \tag{8}$$

Let $d(a_i) = p(a_i) - \epsilon$ define the distance between selection probability of an action and the base value. The preference level, $k(a_i)$, of action a_i is defined to be proportional to the square of the distance $d(a_i)$, as given by

$$k(a_i) = \text{sign}(d(a_i))d(a_i)^2, \qquad i = 1, \ldots, n \tag{9}$$

Therefore, the set $\boldsymbol{K} = \{k_1, \ldots, k_n\}$ represents the preference levels for the actions in a given state, where the magnitude of each element represents the magnitude of preference for each action, and the sign represents a "preferred to" ($k > 0$) or "preferred not to" ($k < 0$) action.

Let $\boldsymbol{K}_o$ denote an agent's initial set of preference levels for the actions in a state, and let $\boldsymbol{K}_m$ denote the preference levels of adviser m in the same state. For each action a, the advised preference level is a linear combination of the advisee and adviser preference levels, as given by

$$\hat{k}(a_i) = k_o(a_i) + \lambda_t(s)k_m(a_i); \qquad i = 1, \ldots, n \tag{10}$$

where $\lambda_t(s,m)$ controls the influence of advice from adviser m and will be derived in Section 3.2. The advised action selection probabilities, $\hat{p}(a_i)$, are then obtained by

$$\hat{p}(a_i) = \frac{\epsilon + \text{sign}(\hat{k}(a_i))\sqrt{|\hat{k}(a_i)|}}{\sum_{j=1}^{n} \hat{p}(a_j)}; \qquad i = 1, \ldots, n \tag{11}$$

Equation (11) ensures that all selection probabilities will form a valid distribution. The advised policy then selects actions based on the advised selection probabilities $\hat{p}(a_i)$.

3.2 Advised Policy Convergence

As the use of advice will alter the selection probabilities for each action, and hence the agent's policy, this can potentially prevent an agent from learning the true value of each $Q(s,a)$ and developing an optimal policy. Given the variety of methods available for updating the values of $Q(s,a)$, often having different convergence requirements, we will demonstrate convergence with an advised policy for two commonly used methods: Q-learning and SARSA(0). Both methods update one $Q(s,a)$ at a time, however, the Q-learning update (4) does not depend on the agent's current action selection policy, while the SARSA(0) update (5) does. Consequently, the SARSA(0) method has more strict convergence requirements.

As detailed in [30], the convergence of SARSA(0) can be proven by treating the update as a stochastic process described by Theorem 1 of [31]. The SARSA(0) method is shown to converge to the true values of $Q(s,a)$ when the policy for selecting actions guarantees infinite exploration (*i.e.*, each action in every state is experienced infinitely many times) and that at time infinity the policy becomes greedy. Naturally, the policy at time infinity will be optimal given that the true values of $Q(s,a)$ have been reached and actions are greedily selected. The convergence of Q-learning only requires infinite exploration to reach the true values of $Q(s,a)$; however, the policy must also become greedy at time infinity if an optimal policy is to be reached.

Thus, for Q-learning and SARSA(0) to converge using an advised policy, two requirements must be met. First, the advised policy must guarantee infinite exploration and become greedy at time infinity. Second, the influence of advice must diminish to zero at time infinity, enabling the agent to converge to a stationary policy. The fulfillment of these two requirements will guarantee convergence to an optimal policy in the presence of advice.

Lemma 3.1. *Let D_i be an increasing sequence of σ-fields, and let A_i be D_i measurable. Then the following holds with probability 1:*

$$\left\{\omega : \sum_{i=1}^{\infty} P(A_i|D_{i-1}) = \infty\right\} = \{\omega : \omega \in A_i \quad i.o.\} \tag{12}$$

Lemma 3.2. *Consider a communicating MDP and the reinforced decision process:*

$$(x_1, a_1, r_1, \ldots, x_t, a_t, r_t, \ldots) \tag{13}$$

Let $v_t(s)$ denote the number of visits to state s up to time t, $v_t(s,a)$ denote the number of times action a has been chosen in state s during the first t time steps, and $t_s(i)$ denote the time when state s has visited the ith time. Assume that the action at time t, a_t, is selected purely on the basis of the statistics D_t:

$$P(a_t = a|D_t, a_{t-1}, D_{t-1}, \ldots, a_1, D_1) = P(a_t = a|D_t) \tag{14}$$

where D_t is computed from the full history $(x_0, a_0, r_0, \ldots, x_t)$. Further, assume that the action selection policy is such that

$$\left\{\omega : \lim_{t\to\infty} v_t(s)(\omega) = \infty\right\} \subseteq \left\{\omega : \sum_{i=1}^{\infty} P(a_{t,i} = a|D_{t,i})(\omega) = \infty\right\} \tag{15}$$

Then, for all (s,a) pairs, $v_t(s) \to \infty$, and $v_t(s,a) \to \infty$, with probability 1.

Lemma 3.3. *Consider an agent with a policy π_o given as a set of probabilities $P(a|s,t,Q)$ determined from a Boltzmann distribution (6). If the temperature parameter $\tau_t(s)$ is defined as*

$$\tau_t(s) = \frac{\ln(v_t(s))}{\max_{a\in A} |Q_t(s, a_{max}) - Q_t(s,a)|} \tag{16}$$

where $a_{max} = \text{argmax}_{a\in A} Q_t(s,a)$, then policy π_o guarantees that each action a in every state s is experienced infinitely often and becomes greedy at time infinity.

The proofs for Lemmas 3.1–3.3 can be found in [30]. Lemmas 3.1 and 3.2 show that if the sum of selection probabilities for each action is infinite, then each action a in every state s will be experienced an infinite number of times (*i.e.*, infinite exploration is achieved). Lemma 3.3 provides a policy that achieves infinite exploration and becomes greedy at time infinity. Therefore, to achieve infinite exploration under an advised policy, we wish to show that $\sum_{i=1}^{\infty} P(a|s, t_s(i)) = \infty$, where $t_s(i)$ is the time of the ith visit to state s.

Theorem 3.4. *Let an agent determine its initial action selection probabilities, $p_o(a_i)$, via a method, which ensures infinite exploration and becomes greedy at time infinity, such as Boltzmann exploration (Lemma 3.3). Let the advised policy $\hat{\pi}$ be obtained by incorporating advice via (10) and having $\lambda_t(s)$ defined as*

$$\lambda_t(s) = \left(\frac{\epsilon}{v_t(s)} - 1\right)^2 + \frac{k_o(a_{min})}{\epsilon^2} \tag{17}$$

where $a_{min} = \text{argmin}_{a\in A} k_o(a)$. Then, the advised policy $\hat{\pi}$ guarantees that each action a in every state s is experienced infinitely often and becomes greedy at time infinity. Additionally, $\lambda_t(s) \to 0$ as $t \to \infty$. Therefore, an advised agent using the Q-learning or SARSA(0) update method will converge to a stationary and optimal policy at time infinity.

Proof. The influence of advice will alter the initial action selection probabilities, but it must not prevent infinite exploration. Utilizing Lemmas 3.1 and 3.2 and the knowledge that $\sum_{i=1}^{\infty} c/i = \infty$, where c is a constant, the requirement of infinite exploration can be fulfilled if the following

condition holds for the advised policy $\hat{\pi}$ with the action selection probabilities defined in (11):

$$\hat{p}(a_i) \geq \frac{c}{v_t(s)}, \qquad i = 1, \ldots, n \tag{18}$$

The advised action selection probability $\hat{p}(a_i)$ will depend on the influence of the adviser in (11), governed by $\lambda_t(s)$. Additionally, as we are concerned with maintaining a minimum action selection probability across all actions $a \in A$, $\lambda_t(s)$ will be limited by the action that reaches the minimum selection probability given by the condition in (18) with the least influence from adviser. This will occur when the actions with the lowest preference level for the advisee and adviser are the same. Let this limiting action be denoted by a_{min}. An expression for $\lambda_t(s)$ which satisfies (18) can be found by relating $\hat{p}(a_{min})$ to the advised preference level $\hat{k}(a_{min})$:

$$\begin{aligned}
\hat{p}(a_{min}) &\geq \frac{c}{v_t(s)} \\
\left[-\hat{k}(a_{min})\right]^{1/2} + \epsilon &\geq \frac{c}{v_t(s)} \\
\left[-k_o(a_{min}) - \lambda_t(s) k_m(a_{min})\right]^{1/2} + \epsilon &\geq \frac{c}{v_t(s)} \\
\frac{-k_o(a_{min}) - \left(\epsilon^2/v_t(s) - \epsilon\right)^2}{k_m(a_{min})} &\geq \lambda_t(s)
\end{aligned}$$

where the sign of $\hat{k}(a_{min})$ in the square root is set to negative, because $\hat{p}(a_{min})$ always results in a negative preference (prefer not to), and the constant c is selected to be equal to ϵ^2. We set the adviser preference $k_m(a_{min})$ to its lowest limit (strongly prefer not to) of $-\epsilon^2$, determined by setting $p(a_i) = 0$ in (9), to come up with a conservative upper bound for $\lambda_t(s)$. This indeed corresponds to the advisor's agreement with the advisee on the least preferred action, resulting in a level of influence for which anything below will guarantee condition (18). Hence, we can express $\lambda_t(s)$ as

$$\lambda_t(s) \leq \left(\frac{\epsilon}{v_t(s)} - 1\right)^2 + \frac{k_o(a_{min})}{\epsilon^2}$$

Therefore, by choosing $\lambda_t(s)$ as defined in (17), each action a in every state s will be experienced infinitely often. Further, the influence of advice diminishes to zero, as for every state s, $\lim_{t\to\infty} v_t(s) = \infty$, resulting in $\lim_{t\to\infty} \lambda_t(s) = 0$. As at time infinity the initial agent's action selection policy becomes greedy, and the influence of advice diminishes to zero, the advised policy will also become greedy at time infinity. Thus, the convergence requirements for Q-learning and SARSA(0) have been met, and the advised agent will converge to a stationary and optimal policy. □

It can easily be verified that the advised preference levels produced by incorporating an adviser's advice via (10) with $\lambda_t(s)$ defined in (17) will always be within the bounds determined by (9).

Lemma 3.5. *With $\lambda_t(s)$ defined by (17), it will always be true that*

$$-\epsilon^2 \leq \hat{k}(a_i) \leq (1-\epsilon)^2 \qquad i = 1, \ldots, n$$

Proof. The lower and upper bounds of $\hat{k}(a_i)$ are $-\epsilon^2$ and $(1-\epsilon)^2$, as determined by (9) for $p(a_i) = 0$ and $p(a_i) = 1$, respectively. The proof that $-\epsilon^2 \leq \hat{k}(a_i)$ follows directly from theorem 3.4, as $\lambda_t(s)$ was derived such that $\hat{p}(a_i) \geq \epsilon^2/v_t(s) \quad i = 1, \ldots, n$, for any values of $k_m(a_i)$. For the largest advised preference level, we consider the case that maximizes the increase in preference level due to advice. This occurs when the preference levels for each action of the advisee and adviser have the same sign, and when $n-1$ actions have equal preference levels of $k(\hat{a}_{min}) < 0$, and a single action has the preference level of $k(\hat{a}_{max})$, where $\hat{a}_{min} = \text{argmin}_{a\in A}\hat{k}(a)$ and $\hat{a}_{max} = \text{argmax}_{a\in A}\hat{k}(a)$. In this case, the advisee's preference level for action $\hat{a}_{max}$ can be expressed as $k_o(\hat{a}_{max}) = -(1/\epsilon - 1)^2 k_o(\hat{a}_{min})$, where $k_o(\hat{a}_{min})$ can be set to $-(\epsilon^2/v_t(s) - \epsilon)^2$, which comes from setting $p_o(\hat{a}_{min}) = \epsilon^2/v_t(s)$. Letting the adviser's preference be set to the maximum value of $(1-\epsilon)^2$, we can compare $\hat{k}(\hat{a}_{max})$ to the upper limit

$$\begin{aligned}
(1-\epsilon)^2 &\geq \hat{k}(\hat{a}_{max}) \\
(1-\epsilon)^2 &\geq k_o(\hat{a}_{max}) + \lambda_t(s)(1-\epsilon)^2 \\
(1-\epsilon)^2 &\geq -(1/\epsilon - 1)^2 k_o(\hat{a}_{min}) \\
&\quad + \left(\left(\frac{\epsilon}{v_t(s)} - 1\right)^2 + \frac{k_o(a_{min})}{\epsilon^2}\right)(1-\epsilon)^2 \\
(1-\epsilon)^2 &\geq (1/\epsilon - 1)^2(\epsilon^2/v_t(s) - \epsilon)^2 \\
&\quad + \left(\left(\frac{\epsilon}{v_t(s)} - 1\right)^2 - \frac{(\epsilon^2/v_t(s) - \epsilon)^2}{\epsilon^2}\right)(1-\epsilon)^2 \\
(1-\epsilon)^2 &\geq (1-\epsilon)^2(\epsilon/v_t(s) - 1)^2
\end{aligned}$$

which will be true for any $v_t(s) \geq 1$ (and $n > 1$). □

An interesting property of the proposed method of incorporating advice is that the convergence of the advised policy does not depend on the adviser. Infinite exploration and greedy action selection can be achieved for any advice. A poor adviser may guide an agent towards imperfect actions and slow the rate at which the agent learns the task, but it will not prevent convergence to an optimal policy. This property is particularly useful for scenarios with multiple advisers in heterogeneous teams.

3.3 Determining When to Use Advice

With regards to utilizing the advice of an adviser, it is of course possible to use several advisers at each time step. However, each time advice is requested from an advisor, a communication cost is incurred. Further, the benefit of advice will diminish over time as the advisee learns its own task, which should be reflected in how frequently advice is requested over time. Thus, polling every advisor for its advice, or polling a fixed number of advisers at each step, is not desirable, as it can result in receiving advice of little benefit and unnecessary communication costs.

Our approach is to enable the advice mechanism to learn when it should accept or reject advice, and when it should continue seeking more advisers.

The advice utilization process, namely learning the relationship between an agent's own stage in improving its performance and the benefit of advice, can be represented by an MDP, and hence a reinforcement learning algorithm can be applied. At each time step, an agent can decide if it needs advice, if it should accept the adviser's advice, ask another adviser, or cease asking additional advisers. Thus, the mechanism must learn when it is most appropriate to request advice, and when the advice should be accepted. The advice utilization process is defined by the $\langle S_A, A_A, T_A, R_A \rangle$ tuple:

State $s_A \in S_A$ is defined by $s_A = \{\psi_o, \bar{\psi}_m, \gamma\}$, which contains information in the agent's current state about (i) the advisee's confidence level, (ii) whether or not the adviser's confidence level is greater than the advisee's, and (iii) the advisee's experience. The elements ψ and $\bar{\psi}_m$ are defined as

$$\psi = \sqrt{\frac{1}{1-\epsilon}} \left[\sum_{i=1}^{n} \left(d(a_i) \right)^2 \right]^{\frac{1}{2}} \tag{19}$$

$$\bar{\psi}_m = \begin{cases} 1, & \text{when } \psi_m > \psi_o \\ 0, & \text{otherwise} \end{cases}$$

The values of ψ_m and ψ_o are obtained using the action selection probabilities of the adviser and advisee, respectively. The value of ψ provides a single metric about the agent's confidence level in the current state, which is bound on the interval $[0, 1]$ for any number of actions. In the zero confidence case, where the selection probabilities of all actions are equal, $\psi = 0$; whereas for the maximum confidence case, when all probabilities are attributed to a single action, $\psi = 1$. The agent's experience in the current state is represented by $\gamma = 1/v_t(s)$.

Action $a_A \in A_A$= {*accept*, *skip*, *cease*}, where *accept* incorporates the advice with (10) and asks the next adviser for advice, *skip* ignores the advice and asks the next adviser for advice, and *cease* rejects the advice and ceases asking for any more advice in the current state.

Transition $T_A(s'_A, a_A, s_A)$ represents the probability of transitioning from state s_A to state s'_A after performing action a_A.

Reward $R_A(s_A, a_A, s'_A)$ defines the reward received by the mechanism after each action and is given by

$$R_A = \begin{cases} \left(1 + \delta\left(\hat{\psi} - \psi_o\right)\right)^2, & a_A = accept \\ \left(1 + \psi_o\right)^2, & \text{otherwise} \end{cases} \tag{20}$$

where $\hat{\psi}$ is found with (19) using the advised action selection probabilities from (11). The coefficient δ is a constant positive value to balance accepting and rejecting advice and can be adjusted depending on the application.

At the beginning of each time step, before an agent takes an action, the advice mechanism will first decide if advice should be requested at all. As the mechanism's state s_A depends on the adviser's advice, and no advisers have yet been polled, this decision making is achieved by selecting an action from the mechanism's policy as if there were an adviser available with stronger confidence than the advisee. Hence, the first state consists of $s_A = \{\psi_o, 1, \gamma\}$, which corresponds to the mechanism being optimistic such that a suitable adviser is available. If the selected action is *cease*, then the agent does not request any advice at the current time step. Otherwise, an adviser is polled for its advice, a new state s_A is formed, and the mechanism chooses between *accept*, *skip*, and *cease*. When the advice is accepted, the advisee's preference levels are updated with (10), and advice is requested from a new adviser (presuming one is available). When the *skip* action is selected, the advice is ignored, and advice from a new adviser is requested. Finally, when the *cease* action is selected, the adviser's advice is not used, and the mechanism ceases requesting any additional advice for this time step. Hence, the mechanism continues requesting advice, either accepting or ignoring the advice, until either the *cease* action is selected or there are no more available advisers.

The order in which advisers are polled for their advice is not part of the mechanism's decision-making process. Instead, a ranking scheme is used to determine the order in which advisers are polled. The optimal adviser for an agent is one with identical capabilities and who has learned the values of $Q(s, a), \forall s, a$. If any heterogeneity exists between agents, their transition probabilities between states will differ, and hence the true values of $Q(s, a)$ for agents will not necessarily be equal for each action a in every state s. As long as the method for learning the values of $Q(s, a)$ implemented by the agent is guaranteed to converge to the true values of $Q(s, a), \forall s, a$, at time infinity $Q_o(s, a) = Q_{m'}(s, a)$, and hence $k_o(a) = k_{m'}(a)$, where the o and m' subscripts denote the advisee and an optimal adviser, respectively. Therefore, selecting the best available adviser is equivalent to finding the advisor with the greatest similarity, using a similarity measure which is maximized when $Q_o(s, a) = Q_{m'}(s, a)$. The dot product between the unit vectors of the advisee's and adviser's preference levels, given by (21), is one such measure. Using the preference levels, as opposed to directly using the values of $Q(s, a)$ or $p(a)$, is advantageous as the values of $k(a)$ are signed, so the dot product can also indicate when two agents oppose each other. Equation (21) reaches a maximum of 1 only when the preference levels are equal, is zero when either agent has zero preference across all actions, and can be negative when the preference levels oppose each other. The similarity measure is updated as an exponential moving average each time the adviser is polled for its advice, given by (22), where ρ is a constant decay rate. During each advice round, advisers are then selected in the order of most to least similar:

$$\beta_m = \left(\frac{\boldsymbol{K}_m}{||\boldsymbol{K}_m||} \right) \cdot \left(\frac{\boldsymbol{K}_o}{||\boldsymbol{K}_o||} \right) \tag{21}$$

$$\omega_{m,t+1} = \rho \omega_{m,t} + (1 - \rho) \beta_m \tag{22}$$

Lastly, the policy for the advice mechanism, $\pi_A(s_A)$, is developed through a Q-learning algorithm with the update rule defined in (4). The steps to be performed by the mechanism to update an agent's preference levels at each time step are summarized in Algorithm 1.

Algorithm 1. Steps for updating the advice mechanism.

$\hat{\mathbf{K}} \leftarrow \mathbf{K}_o$
Set state s_A
$a_A \leftarrow \pi_A(s_A)$
if a_A ! = *cease* **then**
  Set adviser order
  Poll next adviser m
  Set state s_A
  while *next_adviser_available* **do**
    Update similarity measure
    $a_A \leftarrow \pi_A(s_A)$
    if $a_A ==$ *accept* **then**
      Update $\hat{\boldsymbol{K}}$ with (10)
    end if
    if $a_A ==$ *cease* **and** *next_adviser_available* **then**
      Poll next adviser m
    end if
    Set state s_A
    Perform Q-learning update with (4)
    if !*next_adviser_available* **or** $a_A ==$ *cease* **then**
      Break
    end if
    $\boldsymbol{K}_o \leftarrow \hat{\boldsymbol{K}}$
  end while
end if
return $\hat{\boldsymbol{K}}$

There are two additional points to note about the proposed mechanism. First, the mechanism has only two parameters: δ in (20) and ρ in (21). Second, the state space for the advice mechanism is small. With three possible actions, two state elements in the range [0,1], and one binary state element, the state space size is $6 \times u \times v$, where u and v are the number of discretization intervals for the state elements ψ and γ, respectively. A small state space is intentional, as advice is most beneficial in the early stages of learning, and the state space of the advice mechanism must be significantly smaller than the agent's reinforcement learning problem if a benefit is to be received in the early stages of learning.

4. Case Study

To demonstrate the effectiveness of the Preference Advice mechanism, a case study with a heterogeneous robot team is used. The team is composed of robots with varying capabilities, where each robot is individually learning its task with a Q-learning algorithm while receiving advice from their peers or virtual advisers.

A foraging scenario is used where the goal of the team is to collect all the items in the area and bring them to a target zone while avoiding obstacles. An illustration of the scenario is shown in Fig. 1. Each robot is initially assigned an item to collect randomly. The total foraging area is 10 m by 10 m, with four obstacles each having a diameter of 1 m, and a single target zone with a diameter of 2 m. The items to be collected are 0.5 m in diameter, and their quantity is equal to the number of robots used in each experiment. Additionally, there is a circular area 4 m in diameter in the centre of the foraging area that represents rough terrain, which only certain types of robots can pass through. The rough terrain is intended to mimic real applications of heterogeneous robot team, such as search and rescue, where environmental conditions may be more unfavourable for certain robots. Four different types of robots are used, differing in their speed of movement and their ability to traverse the rough terrain: S–NR, S–R, F–NR, and F–R, where S, F, NR, and R represent slow, fast, non-rugged, and rugged, respectively. Only rugged robots can traverse the rough terrain. Each run begins with the items, obstacles, target zone, and robots, randomly positioned within the foraging area, and ends when either all items have been returned to the target zone, or 4,000 iteration steps have been performed.

The learning process for each robot is an MDP with a $\langle S, A, T, R\rangle$ tuple defined as

State $\boldsymbol{s} \in \boldsymbol{S}$ is defined by $s = \{t_d, t_\theta, g_d, g_\theta, o_d, o_t\}$, which contains the distance d and relative angle θ from the robot to centre of the item t, goal location g, as well information about the closest obstacles. The obstacle states, $o_d = \{o_{1,d}, \ldots, o_{k,d}\}$ and $o_t = \{o_{1,t}, \ldots, o_{k,t}\}$, contain the distance d and type t of the closest obstacle along k equally spaced detection rays. Three detection rays are used, orientated with a separation of $pi/10$ rads. The rigid obstacles, walls, and items (other than the one the robot is aiming at) are treated as the same type, while the rough terrain is treated as a separate type. All distances are limited to a maximum range of 2 m and divided into five discrete intervals, while all angles are divided into five discrete quadrants within the interval $[0, 2\pi)$ rads.

Action $\boldsymbol{a} \in \boldsymbol{A}$ is defined by [*move_forward*, *rotate_left*, *rotate_right*, *interact*]. The *move_forward* action will move the robot 0.2 m for a slow robot and 0.4 m for a fast robot, while the *rotate_left* and *rotate_right* actions will move the robot $\pm 1/5\pi$ radians for all robots. *interact* attaches an item to the robot if the robot is within 0.5 m of the item and the item is the robot's assigned item.

Transition $\boldsymbol{T(s', a, s)}$ represents the probability of transitioning from state s to state s' after performing action a.

Reward $\boldsymbol{R(s, a, s')}$ is the reward given to each robot for its action, and is defined in Table 1. A reward is given when the robot moves at least a threshold distance Δd (set to 30% of the robot's step size) towards or away from an item or goal area, or their assigned item is returned. When the robot does not receive any reward for its movement with respect to an item or for returning an item, a reward

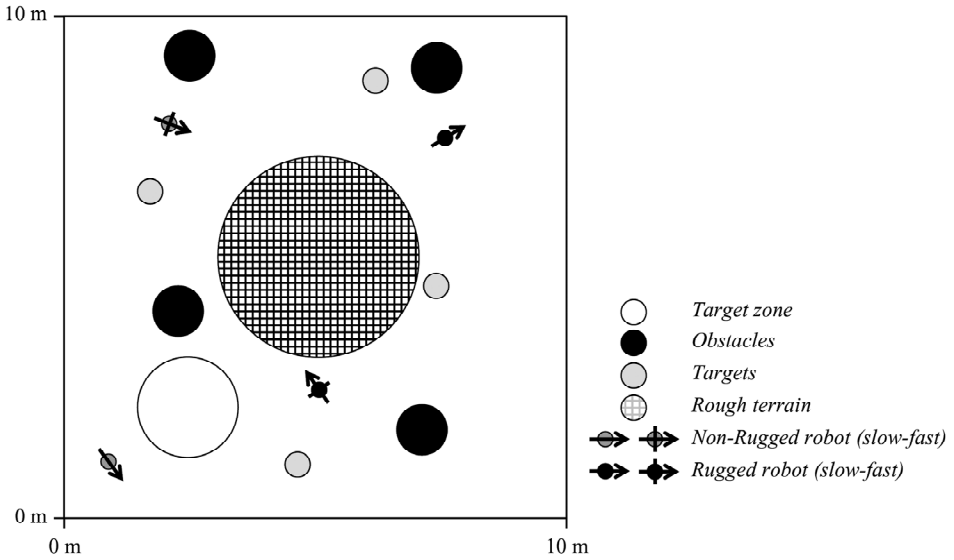


Figure 1. Sample foraging scenario displaying robots, items, obstacles, target zone, and the area of rough terrain.

Table 1
Reward Function for Robot Actions

Behaviour	Reward
Robot moved at least Δd towards assigned item	5
Robot moved at least Δd away from assigned item	0.1
Item moved at least Δd towards target zone	5
Item moved at least Δd away from target zone	0.1
Item is returned to target zone	50
None of the above occurs	1

of 1.0 is given. The magnitude of the reward for each action depends on the priority of tasks as well as team configuration. In a foraging scenario, a relatively high reward should be given when a task is completed, *i.e.*, an item is returned to the target zone. Further, a reward value should be selected for moving towards a designated item or moving an item towards the target zone. Actions that are not aligned with the defined tasks should be given an award less than one (*i.e.*, penalty) to discourage the robots from repeating them, such as moving away from the designated item or carrying the item away from the target zone.

Each robot develops an individual policy π through a Q-learning algorithm with the update rule in (4). The discount factor γ is held constant at 0.3, while the learning rate α decays to 0 as $t \to \infty$, governed by $\alpha = 1/(1 + v_t(s))^{\sigma}$, where σ is a constant that is set to 0.9. Each action has a probability of being selected defined by a Boltzmann distribution with a modified version of $\tau_t(s)$ in (16) that becomes more greedy over time while maintaining a minimum probability of 0.02 over all actions. This modification is necessary in a simulation environment where the state resolution is kept low enough to be tractable. Finally, for the Preference Advice mechanism, actions are selected via a ϵ-greedy policy, where a random action is selected with probability 0.05, otherwise the highest valued action is selected. The coefficient δ applied to the advice mechanism reward in (20) is set to 2.5.

A series of experiments using the Preference Advice mechanism have been formed to demonstrate its capabilities. In the following sections, a *novice* robot refers to a robot that has had zero prior experience at the task, with zero initialized quality values. An *expert* robot is one that has previously performed and learned from the given task for a defined number of runs. Further, experts do not perform policy updates during simulations to perform a controlled analysis of the advice mechanism.

4.1 Experiment 1: Homogeneous Peers as Advisers

When agents are concurrently learning a task, a difficult challenge is to use the partial knowledge of the other agents in a beneficial way. The ability of the proposed mechanism to do so is demonstrated by performing the foraging scenario with 4 novice agents of the same type (chosen to be S–NR). Each agent will inevitably learn different portions of the state space before others, so the mechanism will need to use the partial knowledge of others for the agent to develop the appropriate policy more quickly. When the advice mechanism is used, each agent should learn its task more quickly when compared to the case without the advice mechanism. This can be reflected by the simulation time to complete the task (number of iterations) and total team effort (number of actions) at each run, as well as the average reward received by the team during each run. We also compare the performance of the proposed mechanism to the team performance using the Advice Exchange algorithm [28].

4.2 Experiment 2: Heterogeneous Peers as Advisers

We extend experiment 1 to use heterogeneous advisers to demonstrate the mechanism's performance with advisers possessing different capabilities. Four robots, one of each S–NR, S–R, F–NR, and F–R type, perform the foraging scenario. In the case of non-rugged robots, the rugged advisers will attempt to guide them through the rough terrain, which they are incapable of moving through. Such a scenario will illustrate that the biasing effect of the advice in the proposed mechanism can influence an agent without aggressively forcing it to perform detrimental actions.

4.3 Experiment 3: Expert Advisers of Varying Skill Level

Due to the random nature of action selection (as well as scenario initialization), certain agents may learn the task faster or more slowly than others, resulting in the usefulness of advice varying between agents. When advisers of the same type, but different expertises, are made available to the advisee, the advice mechanism must be capable of recognizing varying levels of knowledge about the task. To demonstrate this, a simulation is performed with one novice S–NR robot having access to expert advisers of the same type trained for 10, 50, and 100 runs. As all robots are homogeneous, the advice mechanism must evaluate advisers based on their skill at the task. The relevance of each adviser is indicated by the similarity measure in (22). The appropriate behaviour of the mechanism is expected to attribute a greater similarity to the experts with more experience as time proceeds.

4.4 Experiment 4: Expert Advisers of Varying Capabilities

As previously stated, for heterogeneous robot teams, the suitability of advice depends not only on the expertise of the adviser but also on how similar the adviser is to the advisee. When advisers of different types but similar expertise are made available to the advisee, the advice mechanism must still be capable of determining the relevance of each adviser. To demonstrate this, a simulation is performed with a novice S–NR robot having access of expert advisers of each possible type (*i.e.*, S–NR, F–NR, S–R, and F–R), all previously trained for 100 runs. In this scenario, the rugged advisers will attempt to guide the robot across the rough terrain, which it is incapable of doing, while the fast advisers will have learned a different sequence of motions than the advisee due to the larger movement during each action. Again, the relevance of advisers is indicated by the similarity measure with (22).

4.5 Experiment 5: Supplement a Team of Novices with a Partially Trained Adviser

An interesting use of advice for robot teams is when a group of novice robots can have access to at least one expert robot. Even if the expert robot is only partially trained, its availability should still accelerate the learning process for the entire team. To investigate this, 4 S–NR novice robots perform the foraging scenario. Each novice has access to the advice of its peers as well as a previously trained S–NR robot. The supplementary expert adviser does not participate in the scenario. The experiment is repeated using the supplementary expert adviser, which is trained for 10, 50, and 100 runs, to illustrate the effects of varying levels of expertise made available in the early stages.

5. Results

In this section, results from the case study described in Section 4. are presented and discussed. During each experiment, the simulations are limited to 200 runs when four robots are used, and 100 runs when one robot is used, with each run limited to 4,000 iterations. The reduction in runs when a single robot is used is due to the robot's performance converging more quickly when other robots are not present to impede its motion. Each experiment is repeated 15 times, and the data is averaged over all 15 trials. Additionally, a 10 point moving average (*i.e.*, averaged over 10 runs) is applied to the averaged results from the 15 trials.

Experiment 1, the use of homogeneous peers as advisers with 4 S–NR robots, is considered first. Figure 2a and 2b shows the simulation time (number of iterations) and total effort (number of actions) required by the team at each run to complete the task without advice, with the Preference Advice mechanism, and with the Advice Exchange mechanism for comparison. It is apparent from the figures that the use of advice provides a consistent reduction in the mean values of both simulation time and total effort. This is especially evident during the transient stage of learning, considered as the first 50 runs where rapid convergence occurs. For both metrics, the use of the Preference Advice mechanism provides a greater improvement than the Advice Exchange mechanism. The benefit of Advice Exchange is most noticeable within the first 100 epochs for simulation time and total effort but becomes indistinguishable from the no advice case beyond that. With the Preference Advice mechanism, the mean values of simulation time and total effort appear to be consistently less than both the Advice Exchange and no advice cases for the first 100 runs, beyond which only the reduction in simulation time is discernible.

The standard deviation of simulation time and total effort at each run is presented in Fig. 3a and 3b. With the Advice Exchange mechanism, there is no discernible reduction in the standard deviation of either simulation time or total effort. However, with the Preference Advice mechanism, there is a clear reduction in the standard deviation of both simulation time and total effort. This reduction is most prominent during the transient stage but is present for all 200 runs. Reducing the standard deviation of simulation time and total effort with the Preference Advice mechanism indicates that it increases the consistency in the team's performance at the task. Again, this improvement is the result of utilizing multiple advisers at each time step.

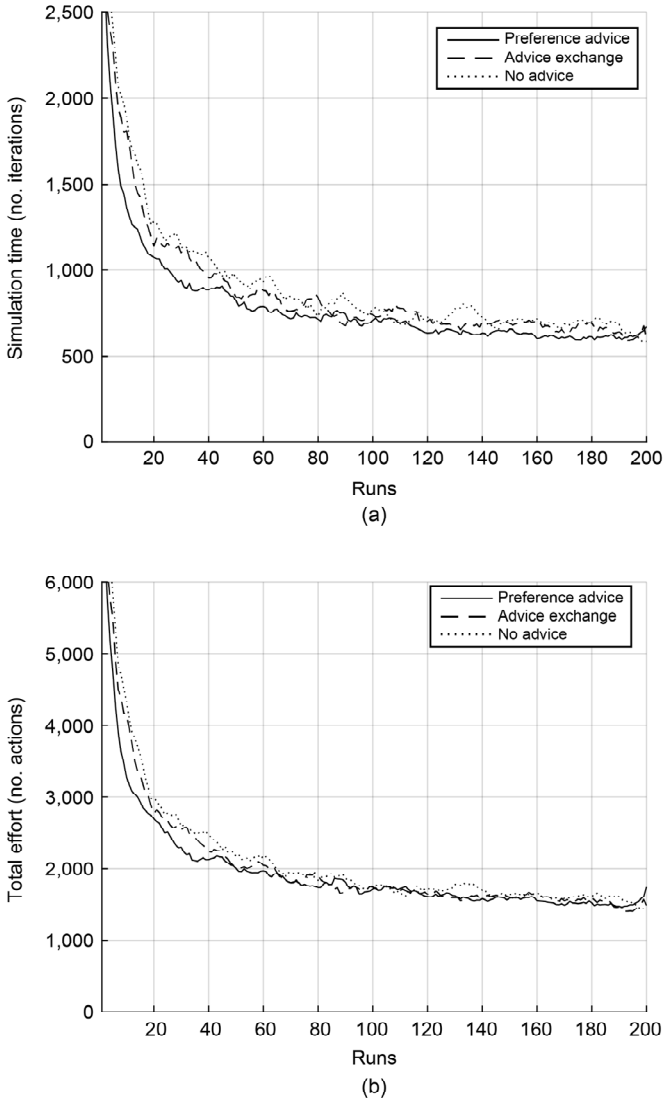


Figure 2. Performance with the Preference Advice mechanism, with the Advice Exchange mechanism, and without advice, for four S–NR robots in experiment 1: (a) simulation time and (b) total effort.

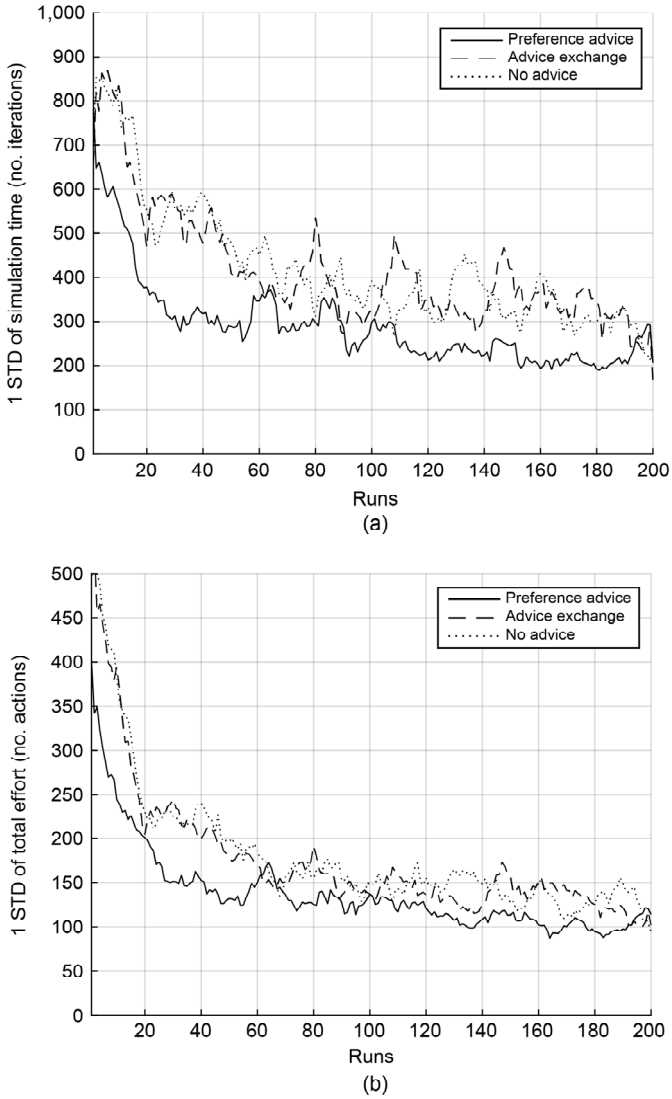


Figure 3. Standard deviation of performance at each run for experiment 1: (a) simulation time and (b) total effort.

The ability of the Preference Advice mechanism to incorporate advice from multiple advisers at each time step is a key factor in the improvement in both the mean values and the standard deviation of simulation time and total effort. In the early stages of learning, a robot's experience in the state space will be sparse, hence several advisers may need to be polled before one with sufficient experience in the desired state is found. If only a single adviser can be utilized at each time step, which is the case for Advice Exchange, it severely limits the likelihood of obtaining useful advice. Additionally, it is likely that each robot will have some experience in the state, but not a large amount. This will frequently prevent advice from being used if it is required that the adviser's performance or experience exceeds the advisee's (as evaluated by certain conditions). Conversely, if an adviser's input can be utilized, regardless of the magnitude of their contribution, the benefits of advice can be redeemed more frequently and provide a more consistent performance improvement.

Lastly, the average reward of all robots in the team is displayed in Fig. 4. Again, the use of advice appears to provide a consistent improvement in the mean value of reward obtained compared to without advice. Acquiring reward in larger quantities indicates that the use of advice encourages the selection of more favourable actions earlier. Interestingly, the distinction between the Preference Advice and Advice Exchange mechanisms is only apparent within the first 50 runs.

The second experiment demonstrates the mechanisms compatibility with heterogeneous advisers, where one of each type of robot (*i.e.*, S–NR, F–NR, S–R, and F–R) performs the foraging scenario together. Hence, no two robots performing the foraging task have identical capabilities. The simulation time and total effort for the team to complete the task at each run, with and without advice, are shown in Fig. 5a and 5b. The comparison to Advice Exchange is not made here, as it is not compatible with heterogeneous advisers. Again, the mean values of simulation time and total effort are consistently lower with the

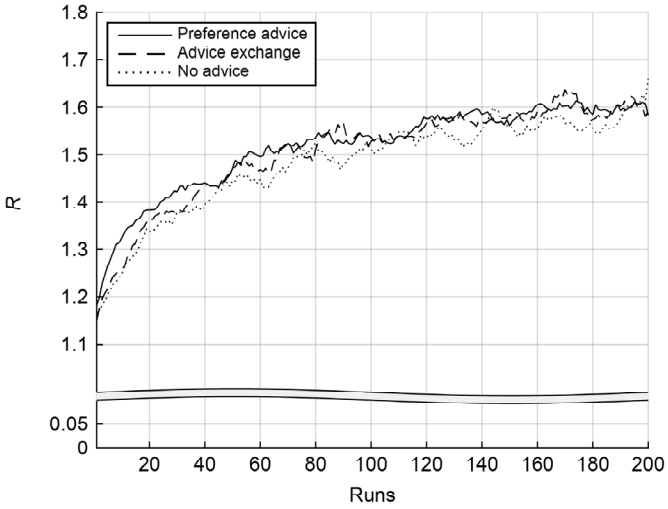


Figure 4. Average reward obtained between four S–NR robots in experiment 1.

Preference Advice mechanism than without advice, particularly during the transient stage of learning. In this scenario, the advice from rugged robots would be to cross the rough terrain, which non-rugged robots are incapable of doing, while the advice from non-rugged robots would be to go around the terrain, which is an inferior policy for rugged robots. If unsuitable advice were being used, we would expect to see an increase in iterations accompanied by an increase in the standard deviation of mission iterations, as a result of robots getting "stuck" performing actions they are incapable of doing. However, we observe a decrease in simulation time and total effort, as well as a decrease in their standard deviations (Fig. 6a and 6b) that is comparable to the reduction in standard deviation from experiment 1 with homogeneous advisers. Therefore, this is a strong indication that the Preference Advice mechanism is compatible with heterogeneous advisers.

Such compatibility is due to the adviser influence $\lambda_t(s)$ in (17) being derived to ensure a conservative use of advice that will not bias the advisee's policy too strongly. The average reward of all team members is shown in Fig. 7, where a similar improvement as in experiment 1 is observed.

Experiments 3 and 4 use virtual expert advisers that do not participate in the task, where the single robot performing the foraging task simply has access to their policies. Such a scenario enables the use of advice over time with the Preference Advice mechanism to be studied under static adviser conditions, as well as demonstrating how the Preference Advice mechanism could be used with alternative advice sources, such as humans. In experiment 3, virtual expert advisers with different amounts of experience are made available to a S–NR robot. Figure 8 shows the percentage which the Preference Advice mechanism requests and accepts advice at each time step. As the robot learns the task, the frequency in which advice is requested rapidly diminishes. Reducing the use of advice over time is a desirable property for real world robot teams, where unnecessary communication and computational costs should

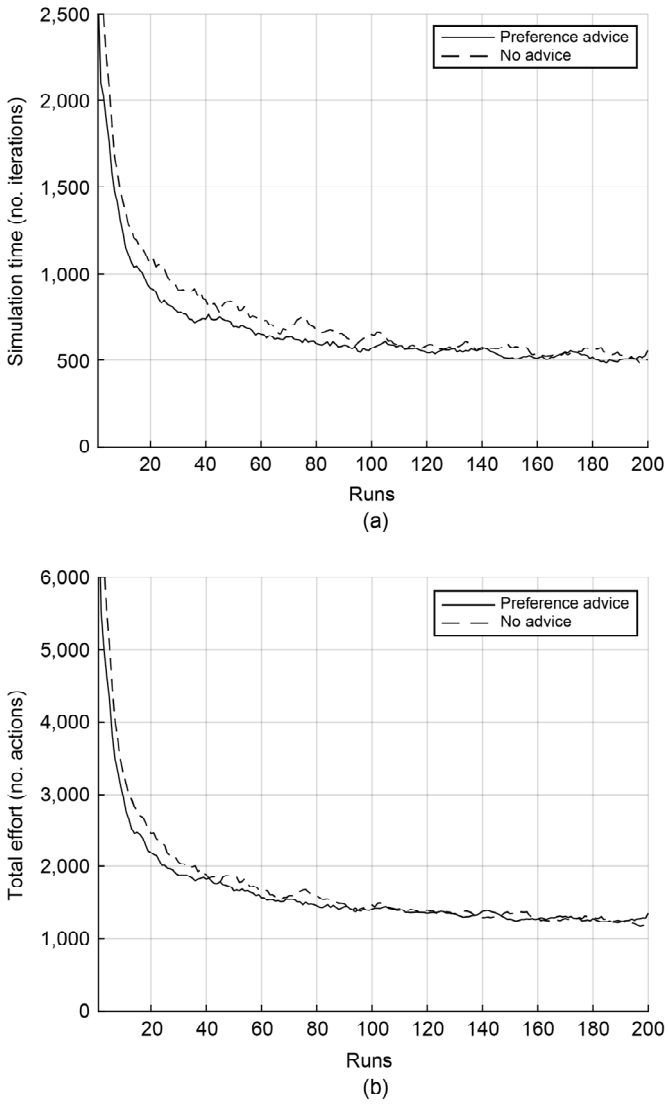
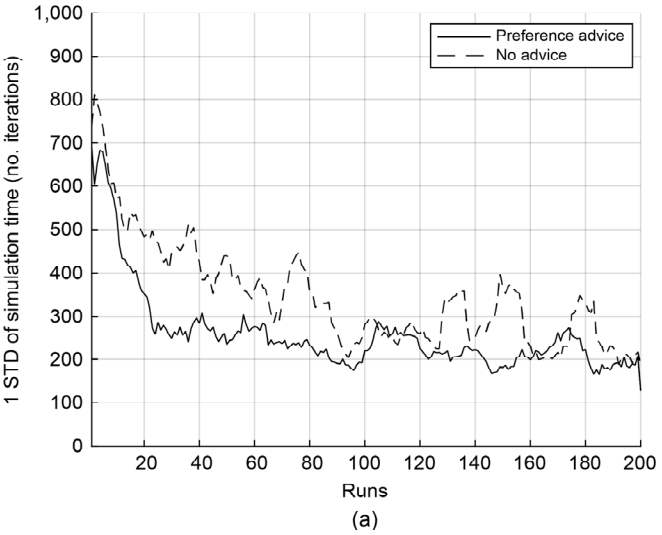


Figure 5. Performance with the Preference Advice mechanism and without advice for four heterogeneous robots (S–NR, F–NR, S–R, and F–R) in experiment 2: (a) simulation time and (b) total effort.

be avoided. The acceptance of advice at each time step increases to a peak near run 30 and steadily diminishes thereafter. The increase in advice acceptance between run 1 and run 30 indicates that the Preference Advice mechanism quickly learns the value of advice, while the decline afterwards paired with the small request occurrence indicates that it also becomes more selective with advice over time. The relevance of each adviser, as measured by (22), is displayed in Fig. 9. The relevance of an adviser increases with its experience, which is the appropriate behavior, resulting in the adviser with the most experience being polled first for its advice. Therefore, despite the mechanism being not directly learned the relevance of each adviser, the ranking scheme implemented has worked successfully.

Experiment 4 uses expert advisers differing in terms of capabilities. A S–NR robot having access to a virtual expert adviser of each type (S–NR, F–NR, S–R, and F–R) is used for the simulation. The relevance of each type of



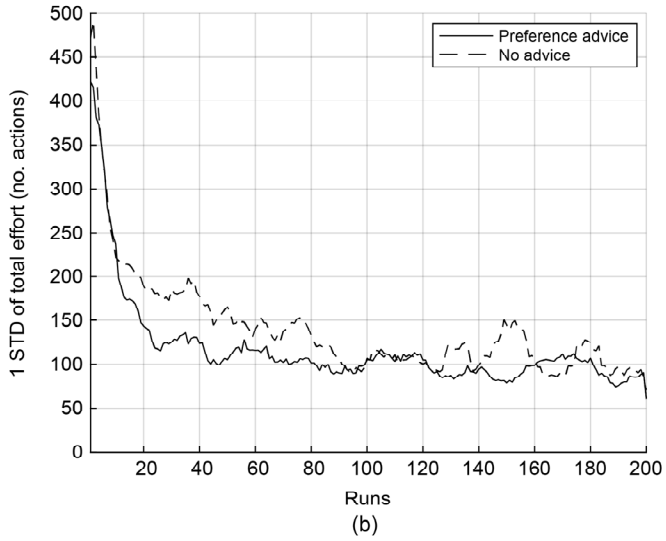


Figure 6. Standard deviation of performance at each run for experiment 2: (a) simulation time and (b) total effort.

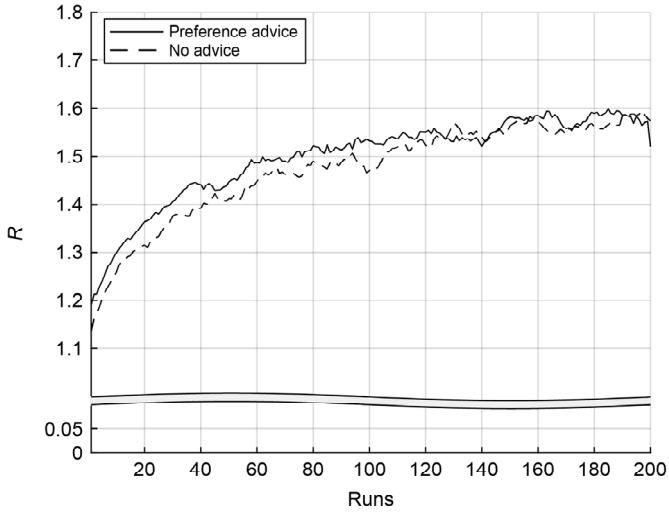


Figure 7. Average reward obtained between four heterogeneous robots (S–NR, F–NR, S–R, and F–R) in experiment 2.

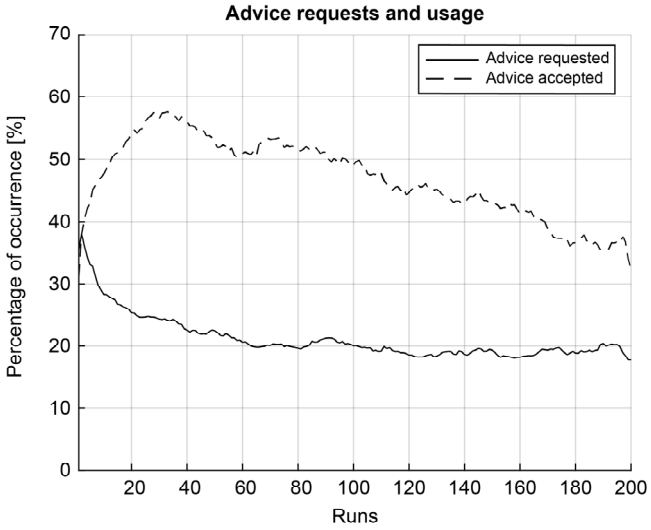


Figure 8. Percentage which advice is requested and used during each run in experiment 3.

adviser, as determined by (22), is shown in Fig. 10. The two advisers of the slow type, namely S–NR and S–R, appear to be equally the most relevant, while the advisers of the fast type, F–NR and F–R, have similar lower, yet similar, levels of relevance.

The Preference Advice mechanism differentiates between the policies of different adviser types, although the results indicate that the difference in policies between rugged and non-rugged robots is smaller than the difference between fast and slow robots. The results from this experiment highlight the importance of having an advice mechanism capable of determining the relevance of advisers on its own, as prior to operation the similarity in policies may not be obvious enough to generate static rules regarding the use of advisers.

Figure 11 shows the number of iterations for the single S–NR robot to complete the foraging task without advice, with advice from advisers of varying skill levels and with advice from advisers of varying capabilities. Both variations in advisers provide similar improvements in simulation time, especially during the transient stage of learning. Again, as the supplementary expert adviser is a virtual adviser, it could also be a human providing advice to the team. Such a reduction in iterations with a virtual expert adviser can provide significant benefits to real world robot teams, where the use of a single human can greatly reduce the time to learn the task at hand, as well as reduce the operational time for the robots, and hence costs. Based on the adviser relevance values, it appears that for experiment 3 the 100 run expert (of the same type as the advisee) was consistently polled first. Additionally, for experiment 4, the relevance of the S–NR and S–R robots were the largest and nearly equal, indicating that those two advisers were consistently polled first. These two observations imply that having two expert advisers with high relevance is not distinctly more beneficial than having a single expert adviser available.

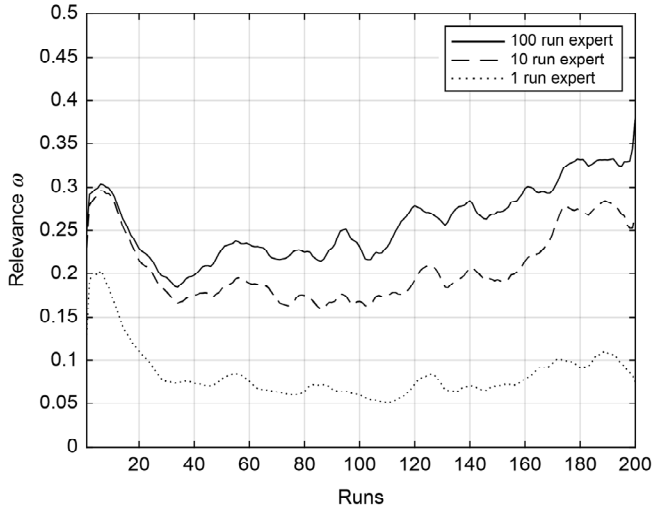


Figure 9. Relevance for advisers of varying skill in experiment 3.

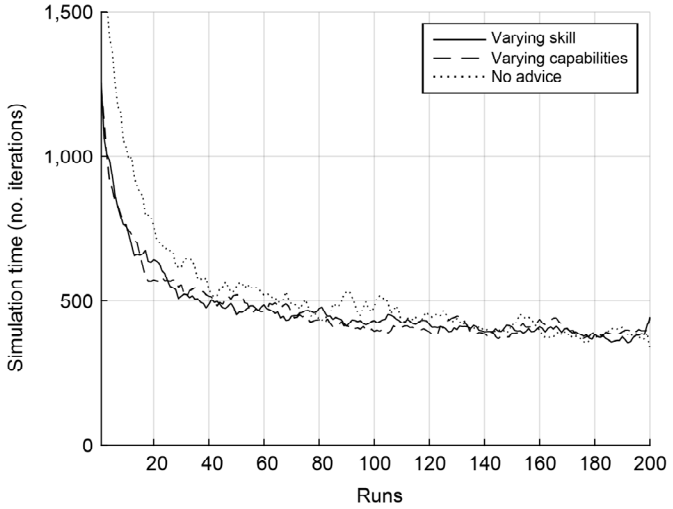


Figure 11. Simulation time for one S–NR robot with advisers of varying skill (experiment 2) and advisers of varying capabilities (experiment 4), compared to without advice.

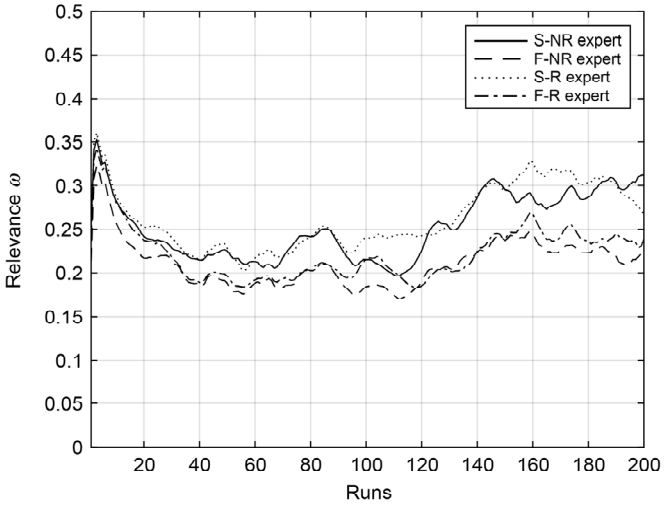


Figure 10. Relevance of advisers of varying capabilities in experiment 4.

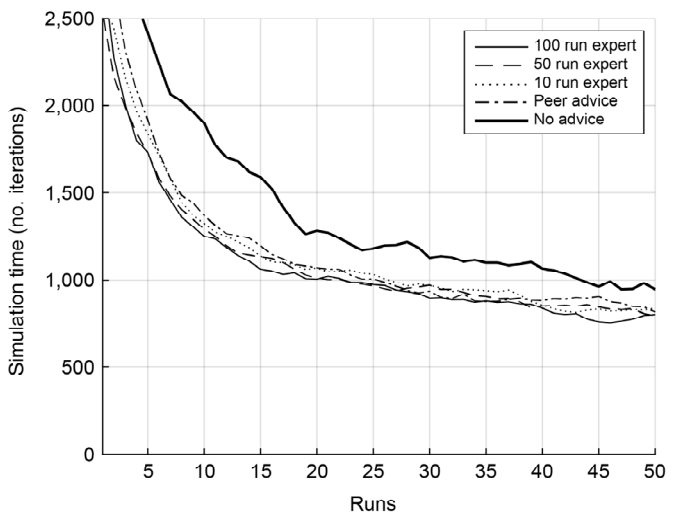


Figure 12. Simulation time for four S–NR robots with a supplementary expert adviser in experiment 5, compared to peer only advice and no advice.

Lastly, we consider experiment 5, where a team of four S–NR robots use peers as advisers, plus one supplementary expert adviser. The simulation time for each case of adviser expertise (10, 50, and 100 runs) is shown in Fig. 12, as well as peer only advice (identical to experiment 1) and no advice for comparison. Given the large number of curves on the plot, for clarity, only the simulation time during the transient stage is shown. Relative to peer only advice, the supplementary adviser provides an additional reduction in simulation time, with the advisers having 50 and 100 runs of experience providing the most improvement. Despite the improvement obtained from having a supplementary expert adviser in addition to peers being relatively small, it does indicate that all four robots are successfully able to leverage the additional source of advice early in the learning process. Similar to the balance between exploration and exploitation with reinforcement learning, there is a similar balance between exploiting an experienced adviser and enabling a robot to experience a sufficient amount of exploration. The policy of the expert adviser with 100 runs of experience could be directly adopted, but it may completely prevent the novice robots from experiencing large areas of the state space.

6. Conclusion

This paper has presented the Preference Advice mechanism for heterogeneous robot teams. The mechanism has the capability to decide when advice should be requested and

accepted, uses multiple advisers at each time step, is compatible with heterogeneous advisers, and reduces the use of advice over time. Further, advice is incorporated into the advisee's policy via a method, which guarantees convergence to an optimal policy.

Experiments were performed with a simulated team of robots performing a foraging task. When all robots have zero prior experience at the task, the mechanism accelerates the learning process, in comparison to without advice, by reducing the number of iterations for the team to complete the task during the initial stages of learning. This improvement in performance also exceeds the improvement experienced with an alternative advice mechanism that uses a form of direct policy adoption. Further, the mechanism is compatible with advisers having different capabilities than the advisee and also appropriately ranks advisers based on their similarity to the advisee and experience. Lastly, the Preference Advice mechanism was shown to provide a reduction in iterations for a team of robots to complete a task when a virtual expert adviser is available. This highlights the importance of advice for robot teams, where the risk of an entire team performing potentially hazardous exploration can be reduced by providing human advice, or pre-training a single robot.

Future work would include applying the Preference Advice mechanism to scenarios with increasing amounts of robot heterogeneity, as well as more complex environments that more closely mimic real world applications of robot teams.

References

[1] C.J.C.H. Watkins, *Learning from delayed rewards*, Ph.D. dissertation, King's College, Cambridge, UK, May 1989. Available: http://www.cs.rhul.ac.uk/∼chrisw/new_thesis.pdf

[2] M.J. Matarić, Reinforcement learning in the multi-robot domain, *Autonomous Robots, 4*(1), 1997, 73–83.

[3] Y. Wang, P.G. Siriwardana, and C.W. de Silva, Multi-robot cooperative transportation of objects using machine learning, *International Journal of Robotics and Automation, 26*(4), 2011, 369–375.

[4] J. Girard and M.R. Emami, Concurrent Markov decision processes for robot team learning, *Engineering Applications of Artificial Intelligence, 39*, 2015, 223–234.

[5] Y. Zhang and C.W. de Silva, Rsmdp-based robust q-learning for optimal path planning in a dynamic environment, *International Journal of Robotics and Automation, 31*(4), 2016, 290–300.

[6] C. Finn, X.Y. Tan, Y. Duan, T. Darrell, S. Levine, and P. Abbee, *Learning visual feature spaces for robotic manipulation with deep spatial autoencoders*, http://arxiv.org/abs/1509.06113

[7] H. Yang and J. Liu, Minimum parameter learning method for an n-link manipulator with nonlinear disturbance observer, *International Journal of Robotics and Automation, 31*(3), 2016.

[8] C. Boutilier, Planning, learning and coordination in multiagent decision processes, *Proc. of the 6th Conf. on Theoretical Aspects of Rationality and Knowledge*, Amsterdam (San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. 1996) 195–210.

[9] C.E. Rasmussen and C.K.I. Williams, *Gaussian processes for machine learning (Adaptive computation and machine learning)* (Cambridge, MA: The MIT Press, 2005).

[10] A.Y. Ng, D. Harada, and S.J. Russell, Policy invariance under reward transformations: Theory and application to reward shaping, *Proc. of the Sixteenth Int. Conf. on Machine Learning*, Bled, Slovenia (San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999) 278–287.

[11] J. Kober and J. Peters, *Reinforcement learning in robotics: A survey* (Berlin, Heidelberg: Springer, 2012) 579–610.

[12] A.S. Polydoros and L. Nalpantidis, Survey of model-based reinforcement learning: Applications on robotics, *Journal of Intelligent and Robotic Systems, 86*(2), 2017, 153–173.

[13] E. Wiewiora, G. Cottrell, and C. Elkan, Principled methods for advising reinforcement learning agents, *Proc. of the Twentieth Int. Conf. on Machine Learning*, Washington, DC (Palo Alto, CA: AAAI Press, 2003) 792–799.

[14] Y. Zhan, H. Bou-Ammar, and M.E. Taylor, Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer, *Proc. of the Twenty-Fifth Int. Joint Conf. on Artificial Intelligence, IJCAI 2016*, New York, NY, USA, 2016, 2315–2321.

[15] L. Ng and M.R. Emami, Concurrent individual and social learning in robot teams, *Computational Intelligence, 32*(3), 2016, 420–438.

[16] R. Bellman, *Dynamic Programming*, 1st edn (Princeton, NJ, USA: Princeton University Press, 1957).

[17] C.J. Watkins and P. Dayan, Technical note: Q-learning, *Machine Learning, 8*(3), 1992, 279–292.

[18] G.A. Rummery, *Problem solving with reinforcement learning*, Ph.D. dissertation, University of Cambridge, 1995.

[19] J. McCarthy, Programs with common sense, *Semantic Information Processing* (Cambridge, MA: MIT Press, 1968) 403–418.

[20] M.E. Taylor and P. Stone, Transfer learning for reinforcement learning domains: A survey, *Journal of Machine Learning Research, 10*(1), 2009, 1633–1685.

[21] G. Boutsioukis, I. Partalas, and I. Vlahavas, *Transfer learning in multi-agent reinforcement learning domains* (Berlin, Heidelberg: Springer, 2012) 249–260.

[22] L. Torrey, J. Shavlik, T. Walker, and R. Maclin, *Skill acquisition Via transfer learning and advice taking* (Berlin, Heidelberg: Springer, 2006) 425–436.

[23] S.D. Whitehead, A complexity analysis of cooperative mechanisms in reinforcement learning, *Proc. of the Ninth National Conf. on Artificial Intelligence (AAAI-91)*, Anaheim, 1991, 607–613.

[24] L.-J. Lin, Programming robots using reinforcement learning and teaching, *Proc. of the Ninth National Conf. on Artificial Intelligence - Volume 2*, Anaheim, CA (Palo Alto, CA: AAAI Press, 1991) 781–786.

[25] R. Maclin, J.W. Shavlik, and P. Kaelbling, Creating advice-taking reinforcement learners, *Machine Learning*, 1996, 251–281.

[26] R.J. Malak and P.K. Khosla, A framework for the adaptive transfer of robot skill knowledge using reinforcement learning agents, *IEEE Int. Conf. on Proc. 2001 ICRA*, Seoul, South Korea, vol. 2, 2001, 1994–2001.

[27] R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild, Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression, *Proc. of the 20th National Conf. on Artificial Intelligence - Volume 2*, Pittsburgh, Pennsylvania (Palo Alto, CA: AAAI Press, 2005) 819–824.

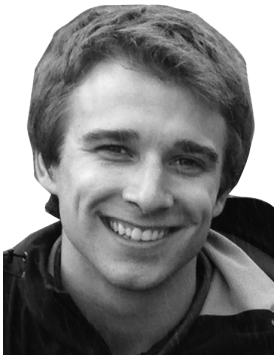
[28] L. Nunes and E. Oliveira, *Cooperative learning using advice exchange* (Berlin, Heidelberg: Springer, 2003) 33–48.

[29] L. Nunes and E. Oliveira, *Exchanging advice and learning to trust, Lecture notes in computer science* vol. 2782 (Berlin: Springer-Verlag, 2003) 250–265.

[30] S. Singh, T. Jaakkola, M.L. Littman, and C. Szepesvári, Convergence results for single-step on-policy reinforcement-learning algorithms, *Machine Learning, 38*(3), 2000, 287–308. Available: http://dx.doi.org/10.1023/A:1007678930559

[31] T. Jaakkola, M.I. Jordan, and S.P. Singh, On the convergence of stochastic iterative dynamic programming algorithms, *Neural Computing, 6*(6), 1994, 1185–1201.

Biographies



Steven Daniluk obtained his Bachelor's degree in Aerospace Engineering from Carleton University in 2015, and joined the Aerospace Mechatronics group at the University of Toronto Institute for Aerospace Studies for his Master's studies. His research thesis was focussed on robot team learning. He graduated in 2017, and has been a Robotics Engineer at Marble since then.

M. Reza Emami is the Founding Chair of Onboard Space Systems at the Luleå University of Technology (Sweden). He has also been the Director of Aerospace Mechatronics group and Coordinator of Aerospace and Design Laboratories at the University of Toronto Institute for Aerospace Studies (Canada) since 2001. His research focuses on concurrent engineering of multidisciplinary systems, such as miniaturized spacecraft and robotic systems, with applications including concurrent base-arm control of space manipulators, satellite formation flying, asteroid exploration and mining, intelligent heterogeneous robot teams, and reconfigurable manipulators. He is the Associate Editor of the *International Journal of Advanced Robotic Systems*.