Learning Inter-Agent Synergies in Asymmetric Multiagent Systems

Gaurav Dixit
Oregon State University
Corvallis, USA
dixitg@oregonstate.edu

Kagan Tumer
Oregon State University
Corvallis, USA
kagan.tumer@oregonstate.edu

ABSTRACT

In multiagent systems that require coordination, agents must learn diverse policies that enable them to achieve their individual and team objectives. Multiagent Quality-Diversity methods partially address this problem by filtering the joint space of policies to smaller sub-spaces that make the diversification of agent policies tractable. However, in teams of asymmetric agents (agents with different objectives and capabilities), the search for diversity is primarily driven by the need to find policies that will allow agents to assume complementary roles required to work together in teams. This work introduces Asymmetric Island Model (AIM), a multiagent framework that enables populations of asymmetric agents to learn diverse complementary policies that foster teamwork via dynamic population size allocation on a wide variety of team tasks. The key insight of AIM is that the competitive pressure arising from the distribution of policies on different team-wide tasks drives the agents to explore regions of the policy space that yield specializations that generalize across tasks. Simulation results on multiple variations of a remote habitat problem highlight the strength of AIM in discovering robust synergies that allow agents to operate near-optimally in response to the changing team composition and policies of other agents.

KEYWORDS

Team Composition; Quality Diversity; Multiagent learning

ACM Reference Format:
Gaurav Dixit and Kagan Tumer. 2023. Learning Inter-Agent Synergies in Asymmetric Multiagent Systems. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 9 pages.

1 INTRODUCTION

Multiagent learning is an appealing paradigm to address many complex real world domains such as coordinating healthcare [15], air traffic control [9, 25], strategy games [2, 28] and robotic automation [10, 14]. Success in many of these applications requires that agents not only learn good actions, but that they learn good *joint actions* so they can coordinate and work together as teams [1, 12]. This problem becomes even more difficult when agents are *asymmetric*—meaning they have different capabilities and different objectives—and must learn task agnostic synergies that capture inter-agent relationships beneficial across a spectrum of tasks. (We use asymmetric to distinguish this class of problems from *heterogeneous* agents that may have different capabilities but share aligned objectives.) Figure 1 highlights the importance of forming asymmetric synergies in a multiagent system.

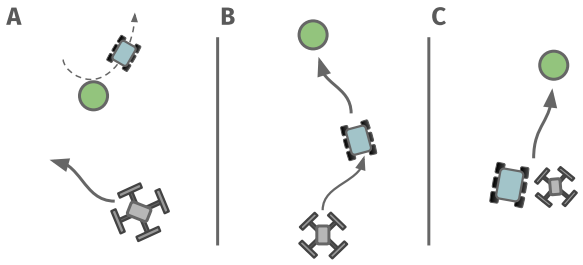

Figure 1: Motivating Example: Rovers and drones are deployed to find excavation sites in a remote environment. A rover can mark a site that a drone must then confirm and communicate to a ground station. [A] The drone starts its search after the rover has marked a site and left (no synergy). [B] The drone learns to loosely follow the rover to locate and confirm a marked site (partial synergy). [C] The rover and the drone learn to team up and navigate together leading to efficient search (full synergy). Our method equips agents to find such valuable inter-agent synergies for robust teaming.

In contrast to traditional learning methods, Quality-Diversity (QD) methods shift the focus from finding one optimal policy to finding several diverse policies [20, 23]. This shift of focus is crucial for learning in asymmetric settings as it offers a potential solution for learning complementary policies. In their most general form, QD methods follow a two-step iterative process: 1) Mutating a population of policies to create diversity; and 2) Cataloging and refining the population by projecting it in a behavior space [27]. However, in multiagent settings, exhaustively exploring the behavior space of policies is often intractable (since the joint space is a function of the action space of all agents).

Recent work has shown success in using a hierarchical approach to transform the behavior space into smaller sub-spaces that makes the diversity search tractable by filtering regions that yield good team performance [6]. However, filtering the behavior space conditioned on the team performance can lead to over-specialization of policies and inter-agent synergies that limits their ability to adapt to new tasks.

This work introduces Asymmetric Island Model (AIM), a multiagent learning framework that enables asymmetric agents to learn diverse synergies required to operate in teams on a variety of tasks. AIM leverages a combination of gradient-based QD and gradient-free evolutionary optimization with a migration strategy that allows both processes to converge to policies that produce good team

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom.

behaviors. The QD process allows agents to learn primitive agent-specific behaviors and explicitly focuses on improving the diversity in the population of agent policies, whereas the evolutionary optimization process evolves populations of teams to improve fitness across a spectrum of team tasks. Migration of policies from the best performing teams to the QD process biases its search towards regions of the policy space that yield useful team policies. Similarly, the replacement of low fitness teams with the policies from the QD process allows the injection of diversity into teams.

The allocation of policies from the QD process is governed by a softmax distribution that is updated via a gradient to maximize the performance of the QD population across the team tasks. *The competitive pressures arising from the changing population sizes of the asymmetric agent classes on the different team tasks, drives the QD process to explore regions of the policy space that yield complementary synergistic behaviors that can be generalized across tasks.*

We show the effectiveness of learning with AIM in several scenarios created in a remote habitat problem in which agents of three distinct classes must work together to find and excavate regolith. The ability of our method to learn generalizable synergies is highlighted in an experiment that evaluates teams on an unseen task.

2 BACKGROUND AND RELATED WORK

2.1 Cooperative Multiagent Learning

A fundamental problem that makes learning challenging in multiagent settings is the credit assignment problem: agents must deduce the contribution of their actions from a single metric of team performance [11, 26]. Methods based on reward decomposition such as Learning Individual Intrinsic Reward (LIIR) [7] utilize a linear combination of a parameterized intrinsic reward and the team reward to facilitate learning [8]. However, to ensure alignment between the two rewards, the intrinsic reward is updated based on a gradient from the team reward which makes this difficult to scale to problems with sparse team rewards. Multiagent DDPG [16] also partially addresses this problem by using a centralized critic that optimizes joint actions for all agents in the system. Like LIIR, it also relies on a dense team reward and can become intractable as the number of agents increases.

Multiagent Evolutionary Reinforcement Learning (MERL) combines gradient-based learning on an agents-specific reward with an evolutionary algorithm (EA) that maximizes the team reward. By applying selection pressure on policies optimized for agent-specific rewards, MERL ensures alignment [17]. However, its shared replay buffer architecture limits its application to homogeneous agents.

2.2 Learning Diversity

Quality-Diversity (QD) methods [19] offer a solution to learning diverse policies in multiagent systems that require agents to operate with asymmetric agents and tasks. QD methods can be described as iterative processes that mutate policies in a population and organize them in a behavior space for refinement. Because defining the behavior space requires significant insight into the problem, recent methods [4, 5] have used dimensionality reduction methods to infer the behavior space from the latent space of the policy population. In multiagent settings, the behaviors space is a function of the number of agents and their action spaces – making exhaustive exploration intractable. Multiagent Coevolution for Asymmetric Agents (MCAA) alleviates this problem by filtering the behavior space into smaller sub-spaces that can yield policies that work well in teams [6]. The filtering is determined by the team task and therefore requires repeated filtering and re-learning in response to changes in the task and team dynamics. In contrast to QD, methods such as Malthusian RL and Minimal Criterion Coevolution [3, 13] implicitly maximize the diversity in a population via resource limitation. While both methods promote the discovery of synergies, the synergies are largely task dependent (agent-task relationships). The goal of our method is to learn inter-agent synergies that can be generalized across a wide spectrum of tasks.

3 ASYMMETRIC ISLAND MODEL

Asymmetric Island Model (AIM) is a multiagent learning framework that leverages a combination of agent-specific and team-wide objectives to produce teams of asymmetric agents (agents with different capabilities and objectives) that learn diverse agent synergies required to cooperate on a variety of tasks. AIM produces a set of "islands" on which agents are trained on agent-specific tasks and a set of "mainlands" on which teams of agents are trained to optimize team tasks. Figure 3 provides a high level overview of AIM.

On each island, a Quality-Diversity (QD) process maintains a diverse population of agents of a specific class, trained to maximize an agent-specific reward. This allows agents of each class to learn primitive behaviors that can be potentially useful on the team tasks. Figure 2 shows the learning process on an island of rovers. Each mainland represents a distinct team task and maintains a population of teams (groups of policies) that is optimized using an evolutionary algorithm (EA). Each island learns a softmax distribution that dictates the allocation of its population to the mainlands (team tasks). Periodically, policies from the islands are sampled and sent to the mainlands (allocated to the mainlands using the softmax distribution of each island) to replace the lowest performing teams. This migration of policies allows AIM to inject diversity acquired using QD to the team tasks. Similarly, policies that are part of the best performing teams on the mainlands are sent to their respective islands in-order to bias the diversity search process towards regions of the policy space that yield good team behaviors.

The softmax distributions on each island is updated using a gradient rule to maximize allocation to mainlands on which the policies of that island get the highest team reward. The competitive pressure created by the allocation of island populations to the mainlands forces each agent class (island) to learn complementary policies that can be generalized across different tasks (mainlands).

3.0.1 Diversity Search on Islands. The QD process on the islands progresses as follows: Each agent class is assigned to a separate island. A population of policies (neural networks) is initialized for each island. The distribution of the population of each island i to the mainlands is defined as a softmax μ_i over a weight vector ω_i. Every island starts by performing N iterations of the QD process in parallel (algorithm 1, lines 2-3). On Island i, a random policy π is sampled from the current island population pop_i (line 4). The policy is mutated (by perturbing the weights of the neural network; line 5), then evaluated in the environment (line 6). The weights of the mutated policy $\pi\prime$ are then updated using PPO (line 7) [24],

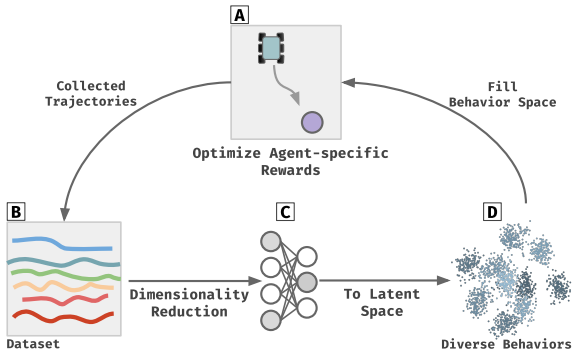

Figure 2: Quality-Diversity (QD) optimization on an Island. Agents (policies) are trained on an agent-specific task (A). Agent-specific data collected from the evaluation becomes part of a dataset (B) that is used to train a dimensionality reduction method (C). The resultant latent space is used as the behavior space which is then filled using QD by mutating the existing policies (D). The process then repeats by training the mutated policies (A).

the policy is added to the island population (line 8) and agent-specific data τ collected from the evaluation becomes part of the dataset used to train a dimensionality reduction (DR) method (line 9). The resultant of the DR is a latent space that is used as the policy space for the agents [5]. Any agent-specific data that is appropriate for the task can be used as the dataset. Previous works have successfully used end effector positions for a robotic arm [5], list of (x, y) coordinates and agent-specific metrics like average speed [6] as the agent-specific data to learn a policy space. The data used by our experiments is described in section 4.3.2.

After the N island iterations, the updated dataset is used to train the DR method which yields an updated latent space of policies for each island (algorithm 1, line 11). The policies on each island are then projected in this new latent space (using their corresponding data τ) to discard policies that are too close in the space (lines 12-13).

Algorithm 1: Islands (agent-specific tasks)

```
1  Function islands(pop_I:|I| Populations):
2      for iteration ← 0 to N do
3          Do in parallel for island i ∈ I
4              sample policy π ∈ pop_i
5              π′ = mutate(π)                    // perturb policy weights
6              τ = rollout(π′, r)                // with agent-specific reward r
7              Update π′ using PPO               // update policy π′
8              pop_i ← pop_i ∪ π′                // add to island population
9              dataset_i ← dataset_i ∪ (τ, π′)
10     foreach island i ∈ I do
11         train_DR(dataset_i)                   // update latent space
12         foreach policy (τ, π) ∈ pop_i do
13             DR(τ)                             // project to latent space
```

Algorithm 2: Mainlands (team tasks)

```
1  Function mainlands(T_M:|M| populations of |T| teams):
2      Do in parallel for mainland m ∈ M
3          T ← T_m                               // Teams for mainland m
4          for generation ← 0 to N do
5              foreach team t ∈ T do
6                  φ_t = evaluate (t)
7              Rank population T based on fitness φ_{0:T}
8              E = T[0 : e]                      // select first e teams as elites
9              Select the remaining (|T| − e) teams from T, to
                form set S using tournament selection
10             while |S| < (|T| − e) do
11                 crossover random policies {(π_x, π_y) |
                    π_x ∈ E, π_y ∈ S, (π_x, π_y) ∈ I}, append to S
12             T ← S ∪ e
```

3.0.2 Team Optimization on the Mainlands. Each mainland is assigned a distinct team task and is initialized with a population of teams. Policies are sampled from the softmax distribution of each island, which allocates a fixed number of policies to each mainland. Teams are then created on each mainland by randomly grouping t_n policies. Random grouping ensures that the proportion of each agent class in a team on a mainland is representative of the proportion of that class' population that was assigned to that mainland.

Each mainland evolves N generations of teams by using cooperative co-evolutionary algorithm (CCEA) [21, 22] (algorithm 2). Teams are evaluated on the mainland task and assigned a team fitness (lines 5-6). The team population is then ranked (line 7) and divided into e high fitness elite teams (line 8) and $(|T| - e)$ lower fitness teams. The policies from the $(|T| - e)$ teams are subjected to crossover with the policies from the e elite teams using tournament selection (line 9-11). During crossover of two policies (π_x, π_y), it is ensured that they belong to the same island I (i.e. they are the same agent class).

3.0.3 Policy Migrations. After both the island and mainland processes have completed N iterations, it is crucial to share information between the two processes. Policies from e elite teams from all the mainlands (elites across all tasks) are added to the population on the islands (algorithm 3, lines 9-10). This migration affects the latent representation on the islands since the elite mainland policies will now participate in the island rollouts (algorithm 1, lines 4-6) and provide data for the dataset (line 11). Ultimately, this biases the QD process to search policies in the regions of the policy space that yield high performance team policies. Thus every latent space update ensures that islands progressively explore the regions of the policy space that can deliver promising team behaviors.

The weights w_i of the allocation distribution μ_i that gives allocation of island population pop_i for each mainland m is a softmax $\boldsymbol{\mu(m, i)} = \frac{e^{w_{m,i}}}{\sum_{j-m} e^{w_{j,i}}}$. It is now updated via a gradient rule (equation 1) to change the allocation of pop_i in the direction that maximizes the cumulative performance of pop_i across the mainlands (algorithm 3, line 11).

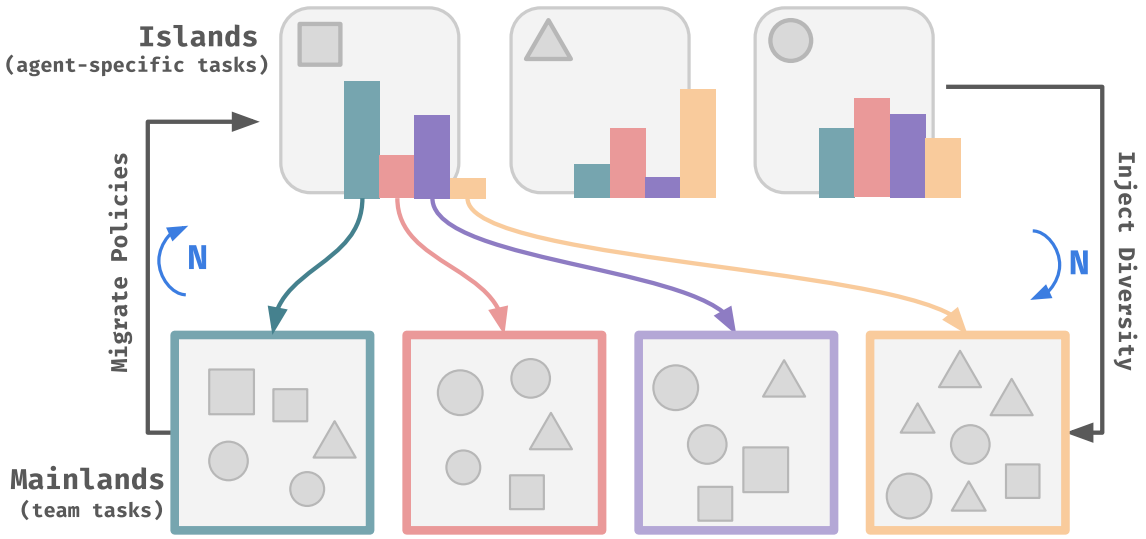

Figure 3: Populations of policies for each asymmetric class are divided into islands. Populations on the islands are trained to optimize agent-specific rewards and maintain diversity using Quality-Diversity (figure 2). Policies from the islands are allocated to the mainlands and grouped into teams to create a population of teams on each mainland (a mainland represents a distinct team task). Teams are optimized to maximize team rewards using an evolutionary algorithm. Periodically (every N steps), policies from the best performing teams from each mainland are sent back to the islands to bias the island population diversity towards good team behaviors, whereas policies from the islands are sampled to create new teams that replace the lowest performing teams and inject new diversity on the mainlands.

Algorithm 3: Asymmetric Island Model (AIM)

1 Initialize I islands, one island per agent class
2 Initialize a population pop_i of policies π for each $i \in I$
3 Initialize M Mainlands, one per team task
4 **Function** `AIM`(I:*Islands*, M:*Mainlands*)**:**
5 **for** $k \leftarrow 0$ **to** ∞ **do**
6 **do in parallel**
7 Pop_I = `islands` (Pop_I)
8 T_M = `mainlands` (T_M)
9 **foreach** *island* $i \in I$ **do**
10 $Pop_i \leftarrow Pop_i \cup T_{m,i}[0:e])\ \forall m \in M$
11 $w_{k+1,i} \leftarrow$ update$(w_{k,i})$ `// according to eqn (1)`
12 **foreach** *mainland* $m \in M$ **do**
 `/* Replace` $(|T|-e)$ `teams by sampling islands */`
13 $T_m \leftarrow T_m[0:e] \cup (|T|-e) \sim w_{k+1,i}, \forall i \in I$

$$\omega_{k+1,i} = \omega_{k,i} + \alpha \left[\sum_{m=1}^{|M|} \nabla_w \mu(m,i)(f_{m,i} - v log \mu(m,i)) \right] \quad (1)$$

In equation 1, $\omega_{k,i}$ is the weight vector ω for island i, iteration k (algorithm 3, line 5). α and v are the adaptation and regularization rates. $f_{m,i}$ is the cumulative performance of pop_i on mainland m and $log\mu(m,i)$ is entropy regularization to ensure that mainland m has a non-zero population of policies from island i.

Finally, policies from the islands are allocated to mainlands to replace policies from the $(|T|-e)$ lowest performing teams using the updated distributions $\mu_I(\omega_{k+1})$ (algorithm 3, line 13). This migration allows the injection of newly discovered diversity from the islands to the teams on the mainlands.

4 EXPERIMENTAL SETUP

We conduct the following experiments to evaluate the efficacy of AIM in finding generalizable agent synergies:

(1) **Asymmetric Coordination** for five distinct scenarios, each of which provides a unique team challenge
(2) **Adaptation to Unseen Tasks** to evaluate the performance of agents on an unseen task.
(3) **Correlations Between Behaviors** to study observed relationships between different agent classes in response to the dynamics of the environment.

4.1 Asymmetric Habitat Problem

We introduce the asymmetric habitat problem, which builds off of the rover exploration problem [6], in which agents are deployed to a remote environment to conduct pre-mission activities required for setting up a habitat. The habitat problem consists of three agent classes: 1) Rovers with sensing equipment; 2) Excavators with digging equipment; and 3) Drones with communication infrastructure. The three classes must operate together to find resourceful dig sites, excavate regolith and communicate the number of excavated dig sites back to a ground station.

The **rovers** are equipped with sensing devices with which they can locate, visit and mark dig sites. To fully mark a dig site, c rovers must visit them simultaneously. We call c the coupling requirement of a dig site.

Each rover has two distinct sensors: one that captures the density of excavators, drones and other rovers around it, and the other that captures the density of marked and unmarked dig sites around it.

$$S_{a,q} = \sum_{j \in J_q} \frac{1}{d(i,j)} \quad (2) \qquad S_{d,q} = \sum_{k \in K_q} \frac{v_k}{d(i,k)} \quad (3)$$

In equation 2, sensor S provides a density of agents of class a in quadrant q of the environment (for our experiments, the space is divided into four quadrants, centered around the agent); J_q is the set of agents of class a in q, within the agent's observation radius, and $d(i,j)$ is the Euclidean distance between the agent i itself and the other agent j.

Similarly, equation 3 computes the density of dig sites in quadrant q, within the agent's observation radius. v_k represents the value associated with dig site k that will be used to compute the team fitness (equation 4). $d(i,j)$ is the Euclidean distance between the agent i itself and dig site k.

The **excavators** posses the equipment to visit marked dig sites. if c excavators visit a dig site d simultaneously, the dig site is considered as successfully excavated. Similar to rovers, c is the coupling requirement for the excavators. The excavators are also equipped with the two density sensors: one for computing densities of the three agent classes (equation 2) and one for computing densities of marked dig sites (equation 3).

Drones are equipped with communication infrastructure and have the ability to communicate excavated dig sites within their observation radius. The cumulative team fitness will only consider the excavated sites that are within the observation radii of drones. Drones have a sensor (equation 2) for computing densities of the three agent classes and a sensor (equation 3) for computing densities of marked dig sites.

Finally, the **Cumulative Team Fitness** is computed using the following equation:

$$\phi(t) = \sum_{k \in K} \prod N_{(c,k)} v_k C_k \quad (4)$$

In equation 4, $\phi(t)$ is the fitness assigned to a team t, K is the set of all dig sites, $N_{(c,k)}$ is an indicator function that is true if c excavators (c is the coupling requirement) visited the dig site k simultaneously, C_k is an indicator function that is true if dig site k was within the observation radius of at least one drone and v_k is the value associated with the dig site k.

4.1.1 ***Inter-Agent Relationships.*** The fitness of a team $\phi(t)$ captures a successfully executed chain of dependencies between the agents: c rovers marking a dig site by simultaneously vising it (spatial intra-agent coupling) to make it available for the excavators (temporal inter-agent coupling), followed by the excavators simultaneously visiting the marked dig sites (spatial intra-agent coupling), then followed by the drones spreading around or teaming with other agent classes (inter-agent spatial and temporal coupling) to make sure the excavated dig sites are considered (given by the indicator function C_k) for computing fitness $\phi(t)$.

4.2 Compared Baselines

The primary metric for evaluating AIM is the quality of the teams it can create. To that end, as a quantitative metric we look at the average team fitness (equation 4) for various scenarios created in the habitat problem. To highlight AIM's ability to yield generalizable agent synergies, we also evaluate the performance of teams using AIM on new scenarios that were not encountered during training. Finally, we also inspect the effect of the environment dynamics on the discovered agent behaviors.

we compare the team fitness on a holdout task against three baselines, each of which address a particular dimension of the learning problem: 1) Multiagent Coevolution for Asymmetric Agents (MCAA), a learning framework that leverages an island model similar to AIM, but is designed to enable asymmetric agents to discover specializations required to complete a single team task optimally [6]; 2) Malthusian Reinforcement Learning (MRL), which drives populations of agents towards specialization by applying pressure through changing population sizes [13]; and 3) Multiagent Evolutionary Reinforcement Learning (MERL), a two-tier learning framework that combines gradient-based and gradient-free optimization for learning cooperative policies with sparse rewards [17].

Our method is designed to combine the strengths of the three baselines: AIM leverages a gradient-based optimizer for optimizing agent-specific rewards and an evolutionary algorithm to optimize teams on the sparse team-wide reward (like MERL), uses an island model to migrate policies that work well in teams to guide the diversity search process (like MCAA), and drives the search for cooperative inter-agent synergies by applying pressure via changing population sizes (like Malthusian RL). The resultant design of AIM enables agents to discover diverse inter-agent synergies that can be generalized to a wide set of team tasks.

4.3 Experimental Parameters

4.3.1 Habitat Environment. Unless specified otherwise, the environment is a 2D space of size 60x60 units. The episode length on the islands and the mainlands is 30 and 80 steps respectively.

Inputs The input to the excavators and drones is a concatenated vector of 16 density values: four values (one per quadrant) for three agent classes (equation 2) and four values for marked dig sites (equation 3) within their observation radius. The input to the rovers is similar, with four additional density values for unmarked dig sites (using equation 3), making it a concatenated vector of 20 values. The observation radius for all three agent classes is randomly assigned to be between $[5, 8]$ units during initialization.

Action Spaces All three agent classes have two navigational actions $(dx, dy) \in [-2.0, 2.0]^2$. A dig site is considered as marked when $c = 3$ rovers visit it together and requires no explicit action. Similarly, a dig site is considered excavated when $c = 3$ excavators visit it together.

Rewards The dense agent-specific reward for all three agent classes is the inverse of euclidean distance from the closest dig site, given by $\boldsymbol{r_{i,t} = \frac{1}{d(i,k)}}$, where $d(i,k)$ is the Euclidean distance between the agent i itself and the closest dig site k at time step t. This dense reward captures a fundamental skill (visiting a dig site) that can be potentially useful to all agent classes for the team task. This reward is used by the Quality-Diversity process on the islands

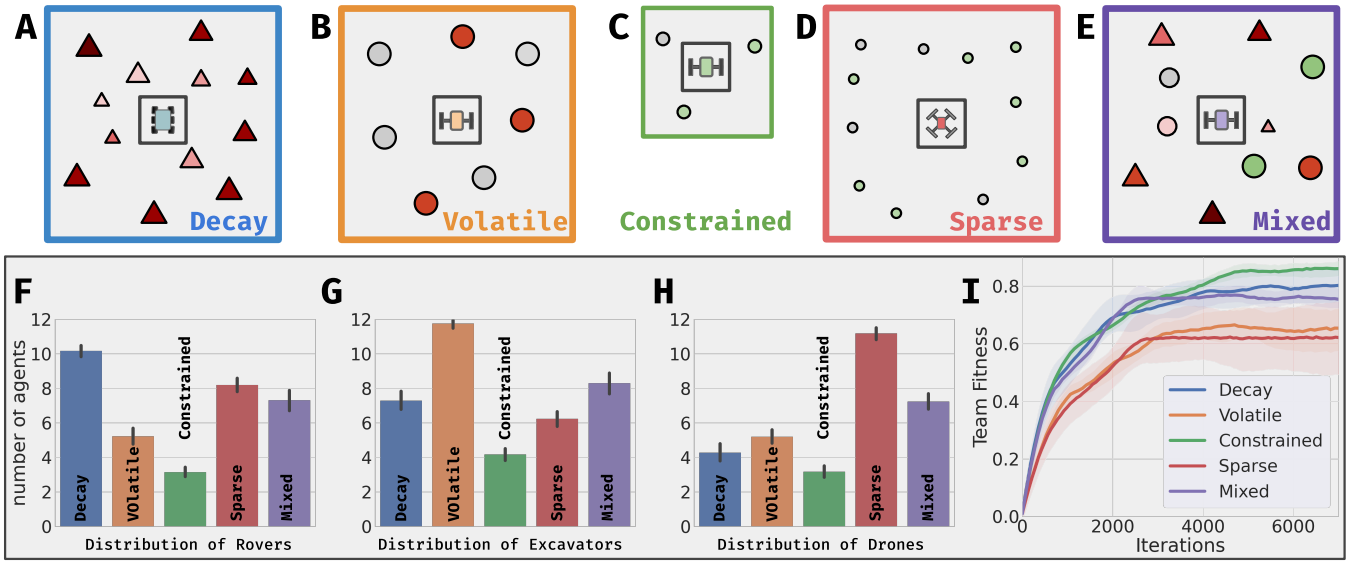

Figure 4: AIM is trained to produce teams that can perform well across five scenarios (Top row (A-E)). The bottom row shows the learnt allocation distribution for the three agents classes (F-H) and the average team fitness across the scenarios (I). In Decay (A), the values of dig sites reduce exponentially. The decay is faster the further they are from the center. Rovers are best suited to perform in this scenario and makeup over 50% of high performing teams (F). In Volatile (B), the sites stay marked for only 5 steps and rovers must re-mark them again for excavators. This scenarios sees a majority of excavators in the team (over 40%) (G) that learn to either follow rovers or spread in the environment to visit sites as soon as they are marked. This experiment highlights AIM's ability to produce strong inter-agent synergies and team compositions that adapt to the task. Section 5.1 describes the performance and distribution of agents across all scenarios in further detail.

in AIM and MCAA, the gradient based training in MERL and as the reward for agents using Malthusian RL.

The sparse team fitness (equation 4) is used by the evolutionary algorithm on the mainland(s) by AIM and MCAA, and by the evolutionary optimizer in MERL. The fitness is assigned to a team at the end of each episode. The number of sites marked by the rovers, the number of sites excavated by the excavators and the number of sites communicated by the drones on mainland m are used as the cumulative performance $f_{m,i}$ for the three classes (in equation 1).

4.3.2 AIM and Baselines. The **agent-specific data** collected during the Quality-Diversity process on the islands are vectors of $(d_{a,t}, d_{k,t})$ tuples for all t time steps of the episode. For the habitat problem, we believe that the distance $d_{a,t}$ to the closest agent a and the distance $d_{k,t}$ to the closest dig site k captures the agent's inclination towards going to a dig site and teaming with another agent. For all three classes of agents, we use this data as input to the dimensionality reduction method on the islands, PCA [5], which gives us the latent space that is used as the behavior space.

The baselines in section 5.2 use parameters from the original works [6, 13, 17] unless specified otherwise. AIM, MCAA and MERL use $N = 1000$ policy updates in their gradient-based process. To ensure fair comparison with Malthusian RL (MRL), we allow it $2N$ v-trace updates for every N gradient updates in AIM (to make up for the evolutionary updates). Every iteration i in section 5.2 corresponds to $2N$ updates for MRL and N updates for other baselines. Both AIM and MRL use rates $\alpha = 1e-05$ and $\nu = 0.01$ (equation 1) and the tournament selection (algorithm 2, line 9) is done according to [18]. The policies on the islands are fully connected feedforward networks with 16 inputs (20 for rovers), 3 hidden layers (size 32) with ReLU activations, and two output heads (similar to [17]). Mutation on the islands (algorithm 1, line 5) is applied to a fraction of the weights ($m_f = 0.15$) by probabilistically ($m_p = 0.9$) perturbing them with Gaussian noise ($m_\sigma = 0.1$).

5 RESULTS

5.1 Asymmetric Coordination

We start by creating five scenarios in the habitat problem, each of which demands a different team composition and unique behaviors from the three agent classes. The goal is two-fold: AIM should be able discover good team compositions and agents trained with AIM should be able to learn diverse policies that are specialized within their own class while still being generalizable across the five scenarios. The scenarios are shown on the top row of figure 4 (A-E). The allocation distribution μ for each agent class is shown on the bottom row (F-H), while the average team fitness (normalized) on each of the scenarios is shown in (I). The μ_i for each class i is the learnt distribution used for allocation of the agent population to the the five scenarios.

In the **first scenario, "Decay"** (figure 4.A), the task is to prioritize high value dig sites (v_k in equation 4). This is enforced by introducing a decay function to the value of dig sites ($v_{k,t+1} = v_{k,t} * 0.5$) and keeping the episode length relatively small (40 steps). With the

value of unmarked dig sites decreasing every time step, rovers must be able to navigate around lower valued sites to prioritize high valued sites. This also implies that rovers are the most important class in this scenario, which is evident from the μ for rovers (F), which shows the highest distribution of rovers (average 11 rovers per team) on Decay (A). With more rovers to mark high values sites before they decay, teams on this scenario are able to excavate over 80% of the sites (I).

In the **second scenario, "Volatile"** (figure 4.B), we limit the duration up to which a marked site stays marked (7 steps), after which it will need to re-marked by the rovers. While a higher number of rovers could solve this problem by repeated re-marking, interestingly, we saw a higher distribution of excavators on this scenario (average 13 excavators vs 5 rovers per team - (G)). We hypothesize (partially confirmed in section 5.3) that the higher requirement of rovers on Decay (A), drives the excavators to find diverse strategies like following rovers, spreading in the environment and camping in regions of previously marked sites that make them the premier class for this scenario. Teams are able to excavate over 70% sites (I) in this scenario.

In the **third scenario, "Constrained"** (figure 4.C), we halve the size of the environment to 30x30. The average team size on this scenario is significantly smaller and the team composition is fairly uniform (small comparable mean for each class as evident from F-H) as expected. The average performance of teams on this task is the highest (I) with teams readily excavating all sites in the environment.

The **fourth scenario, "Sparse"** (figure 4.D), doubles the size of the environment while keeping the number of sites constant, thus creating a very sparse environment. The drones learn to specialize in this environment by teaming with the other two classes to ensure that the team fitness captures every excavated site (evident in section 5.3). The μ for drones (H) also indicates that on an average every team has 12 drones in this scenario (compared to average 7 rovers and 5 excavators).

Finally, we combine Decay and Volatile by adding decaying sites that remain marked for 7 time steps to create the **fifth scenario, "Mixed"** (figure 4.E). The μ for each class indicates uniform team composition on this scenario (average 7 agents per class on each team - (F-H)). Teams on this scenario are able to excavate over 80% of the sites (I).

This experiment highlights the effectiveness of AIM in producing diverse policies for asymmetric agents that foster team work and generalize across a wide variety of scenarios **(over 70% performance across all five scenarios)**.

5.2 Adaptation to Unseen Tasks

While AIM's primary objective is to enable agents to learn diverse generalizable policies, both MCAA and Malthusian RL are designed to specialize policies for one task. This difference has potential consequences on how they compare on tasks that they have been trained on as well on unseen held-out tasks.

5.2.1 Training on the Decay scenario. We first train all the baselines on Decay (section 5.1). Figure 5.A shows the performance of AIM and the baselines on Decay. We evaluate two variants of AIM, both of which use three mainlands: 1) AIM (S=1) in which all three mainlands are assigned Decay; and 2) AIM (S=3) in which the first mainland is assigned Decay while the other two are assigned the Constrained and Sparse scenarios (section 5.1). The is to allow AIM (S=1) to specialize to one task like the baselines while AIM (S=3) has to learn to generalize policies across three tasks. For AIM (S=1) and Malthusian RL, we report the average fitness on the best performing mainland, whereas for AIM (S=3) the fitness of the appropriate mainland (Decay or held-out) is reported.

Teams trained with both AIM (S=1) and MCAA are able to excavate over 80% sites with MCAA slightly outperforming AIM (S=1) (figure 5.A). Because AIM (S=1) has to distribute its best performing policies across the three mainlands, we believe this gives MCAA a slight advantage since all its best performing policies are concentrated on its single mainland. Teams trained with AIM (S=3) learn policies to operate on three different scenarios yet are able to excavate over 70% of the sites and only performs slightly worse than AIM (S=1) and MCAA.

Because MERL is specifically designed for a single class of agents, we use the team composition learnt by MCAA (8 rovers, 4 drones, 5 excavators) to train a population of teams with MERL. Although MERL uses the same agent-specific and team rewards (section 4.3.1) as AIM and MCAA, it is unable to perform comparably, likely due to the lack of diversity in the policies. Teams trained with Malthusian RL (both average and best island performances reported) fail to learn on Decay. We attribute this to at least two potential reasons: Malthusian RL disperses its best performing policies on different islands (similar to AIM (S=3)) and overall lacks diversity in the populations (like MERL).

5.2.2 Evaluation on a held-out scenario. Now we evaluate the performance on a held-out scenario that none of the methods encountered during training. We choose Volatile (from section 5.1) as the held-out scenario. AIM (S=1), (S=3), MCAA and MERL use the same team composition they learnt for Decay.

Teams evaluated with AIM (S=3) outperform the other methods significantly and still manage to excavate over 60% sites (figure

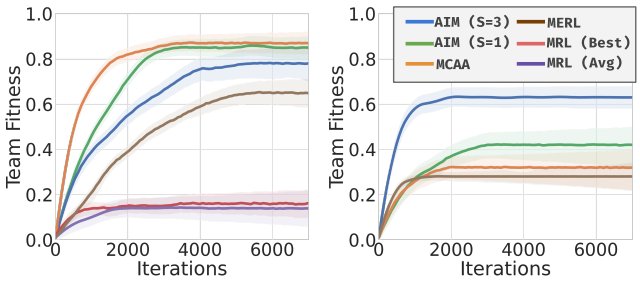

Figure 5: Performance of teams trained with AIM and the baselines on Decay (left), and evaluated on a held-out task (right). In the Decay scenario, variants of our method for specialization (AIM(S=1)) and generalization (AIM(S=3) have a comparable performance to MCAA and outperform Malthusian RL (MRL) and MERL (left). In the held-out task, AIM (S=3) significantly outperforms the baselines (right). This highlights AIM's effectiveness in discovering inter-agent synergies that can be generalized to unseen tasks.

5.B). The regions of the behavior space (on the islands) that were explored by AIM (S=3) were a product of searching for diverse policies that worked across the three scenarios it was trained on. This enabled teams to leverage generalized synergies to operate successfully on the unseen task. This is in contrast to MCAA and AIM (S=1), where the performance significantly reduced because the policy diversity was coupled to the trained task and failed to extrapolate when the dynamics of the task changed.

This highlights the ability of AIM to learn diverse policies that are generalizable across potentially unseen tasks in an environment.

5.3 Correlations Between Behaviors and Tasks

Finally, we take a closer look at some of the behaviors that were discovered by AIM that allowed different agent classes to learn specialized behaviors that worked well across a variety of tasks (section 5.1). In particular, we are interested in inspecting how the simultaneous exploration of behavior spaces, the inter-agent coupling, and the competitive pressure arising from the distribution of populations to the tasks affected the behaviors they learnt.

Figure 6.A shows the correlation between the size of the environment and the average distance between drones and other agent classes. In the habitat problem, rovers have spatial intra-agent coupling wherein they are required to team with other rovers to mark a dig site. The drones on the other hand have an inter-agent coupling: they must try and keep as many marked dig sites in their observation radius as possible so that they are counted in computing the team fitness once the excavators visit them. Drones thus have a nuanced relationship with both the task (dig sites) and other agents (rovers and excavators) that they must learn to balance. Figure 6.A indicates that for smaller environments (size less than 40 units), drones seem more closely linked to the task (dig sites) and prefer to spread in the environment (observed in Decay) or camp around dig sites (observed in Volatile). As the size of the environment decreases, so does the likelihood of finding dig sites (as evident in Sparse). The camping and spreading strategies become less effective and drones learn to follow rovers and excavators instead, as is evident from the decrease in the average distance.

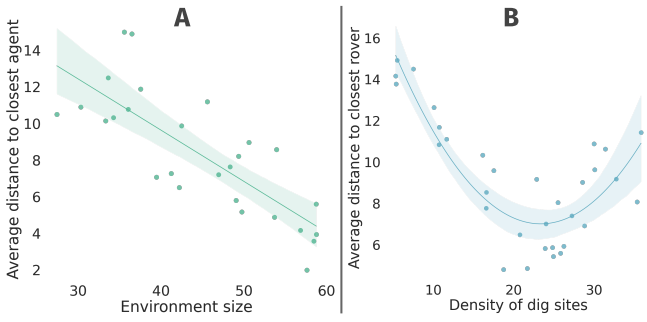

Figure 6: [A] As the size of the environment increases, drones learn to team up with rovers and excavators (reduced average distance) in favor of strategies like camping and spreading observed in section 5.1. [B] With increasing density of dig sites, excavators learn to team up with rovers. This trend reverses for very high density environments in which rovers often revisit already marked dig sites.

A similar inter-agent and inter-task coupling is observed between the excavators, rovers and the dig sites. Figure 6.B shows the correlation between the density of dig sites in the environment and the average distance between excavators and rovers. For sparser environments, the excavators seem to spread in the environment (contrasting the trend seen with drones), whereas with the rise in density, the overall inclination to team with rovers seems to increase (figure 6.B: density $\in [15, 24]$). Interestingly, this trend reverses again as the excavators seem to adopt a coverage strategy. We believe that a higher density might require rovers to repeatedly navigate back to already visited dig sites for remarking (in case of Volatile) which makes the excavator-site (agent to task) relationship more viable than the excavator-rover (inter-agent) relationship.

AIM's ability to train asymmetric classes in concert offers a rich insight into the intra, inter-agent and agent-task relationships required to coordinate and form synergies.

6 DISCUSSION

This work introduces AIM, a multiagent learning framework that enables asymmetric agents to learn diverse synergies required to operate in teams on a variety of tasks. AIM combines Quality-Diversity (QD) and evolutionary optimization with a migration scheme that allows it to find policies that are simultaneously diverse and produce good team behaviors.

QD allows policies to learn primitive agent-specific behaviors and explicitly maintains high diversity in the populations, whereas the evolutionary process evolves populations of teams to improve fitness across a spectrum of tasks. Migration of policies from the best performing teams to the QD process biases its search towards regions of the policy space that produce useful team policies. Similarly, replacing the lowest performing teams with the policies from the QD process injects diversity into teams.

The allocation of policies from the QD process is governed by a softmax distribution that is updated via a gradient to maximize the performance of the QD population across the team tasks. *The competitive pressures arising from changing population sizes of the asymmetric agent classes on the different team tasks drives the QD process to explore regions of the policy space that allow agents to learn complementary synergistic behaviors with other agents that can be generalized across tasks.*

The primary objective of AIM was to empower agent classes to learn generalizable synergies. This is often useful and specially desirable for unexpected changes in the environment or rapid adaptation to new tasks (AIM succeeds at this: figure 5.B). At the same time, there is certainly room for learning nuanced behaviors required to specialize in just one task (figure 5.A). In future work, we will explore extensions of AIM that can adaptively balance between generalization and specialization.

ACKNOWLEDGMENTS

This work was partially supported by the Air Force Office of Scientific Research (grant FA9550-19-1-0195) and the National Science Foundation (grants IIS-1815886 and IS-2112633).

REFERENCES

[1] Stefano V Albrecht and Peter Stone. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258 (2018), 66–95.

[2] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680* (2019).

[3] Jonathan C Brant and Kenneth O Stanley. 2020. Diversity preservation in minimal criterion coevolution through resource limitation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 58–66.

[4] Cédric Colas, Vashisht Madhavan, Joost Huizinga, and Jeff Clune. 2020. Scaling MAP-Elites to deep neuroevolution. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Jun 2020). https://doi.org/10.1145/3377930.3390217

[5] Antoine Cully. 2019. Autonomous skill discovery with quality-diversity and unsupervised descriptors. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 81–89.

[6] Gaurav Dixit, Everardo Gonzalez, and Kagan Tumer. 2022. Diversifying behaviors for learning in asymmetric multiagent systems. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 350–358.

[7] Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. 2019. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. *Advances in Neural Information Processing Systems* 32 (2019).

[8] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[9] Jared Hill, James Archibald, Wynn Stirling, and Richard Frost. 2005. A multi-agent system architecture for distributed air traffic control. In *AIAA guidance, navigation, and control conference and exhibit*. 6049.

[10] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.

[11] Guillaume J Laurent, Laëtitia Matignon, Le Fort-Piat, et al. 2011. The world of independent learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems* 15, 1 (2011), 55–64.

[12] Joel Z Leibo, Edward Hughes, Marc Lanctot, and Thore Graepel. 2019. Autocurricula and the emergence of innovation from social interaction: A manifesto for multi-agent intelligence research. *arXiv preprint arXiv:1903.00742* (2019).

[13] Joel Z. Leibo, Julien Perolat, Edward Hughes, Steven Wheelwright, Adam H. Marblestone, Edgar Duéñez Guzmán, Peter Sunehag, Iain Dunning, and Thore Graepel. 2019. Malthusian 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, 1099–1107.

[14] 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).

[15] Ying Liu, Brent Logan, Ning Liu, Zhiyuan Xu, Jian Tang, and Yangzhi Wang. 2017. Deep reinforcement learning for dynamic treatment regimes on medical registry data. In *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 380–385.

[16] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*. 6379–6390.

[17] Somdeb Majumdar, Shauharda Khadka, Santiago Miret, Stephen McAleer, and Kagan Tumer. 2020. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *International Conference on Machine Learning*. PMLR, 6651–6660.

[18] Brad L Miller, David E Goldberg, et al. 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex systems* 9, 3 (1995), 193–212.

[19] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[20] Jørgen Nordmoen, Kai Olav Ellefsen, and Kyrre Glette. 2018. Combining MAP-elites and incremental evolution to generate gaits for a mammalian quadruped robot. In *International Conference on the Applications of Evolutionary Computation*. Springer, 719–733.

[21] WR Paul, CL William, and KA De Jong. 2002. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms. *Journal Spector* (2002), 15.

[22] Mitchell A Potter and Kenneth A De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 249–257.

[23] Justin K Pugh, Lisa B Soros, and Kenneth O Stanley. 2016. Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI* 3 (2016), 40.

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

[25] Claire Tomlin, George J Pappas, and Shankar Sastry. 1998. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on automatic control* 43, 4 (1998), 509–521.

[26] Karl Tuyls and Gerhard Weiss. 2012. Multiagent learning: Basics, challenges, and prospects. *Ai Magazine* 33, 3 (2012), 41–41.

[27] Vassilis Vassiliades, Konstantinos Chatzilygeroudis, and Jean-Baptiste Mouret. 2017. A comparison of illumination algorithms in unbounded spaces. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 1578–1581.

[28] Oriol Vinyals, Igor Babuschkin, Junyoung Chung, Michael Mathieu, Max Jaderberg, Wojtek Czarnecki, Andrew Dudzik, Aja Huang, Petko Georgiev, Richard Powell, Timo Ewalds, Dan Horgan, Manuel Kroiss, Ivo Danihelka, John Agapiou, Junhyuk Oh, Valentin Dalibard, David Choi, Laurent Sifre, Yury Sulsky, Sasha Vezhnevets, James Molloy, Trevor Cai, David Budden, Tom Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Toby Pohlen, Dani Yogatama, Julia Cohen, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Chris Apps, Koray Kavukcuoglu, Demis Hassabis, and David Silver. 2019. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/.