Learning to Select Mates in Evolving Non-playable Characters

Dylan R. Ashley[0], Valliappa Chockalingam[0], Braedy Kuzma[0], Vadim Bulitko
Department of Computing Science
University of Alberta
Edmonton, Alberta, T6G 2E8, Canada
{dashley, valliapp, braedy, bulitko}@ualberta.ca

***Abstract*—Procedural content generation (PCG) is an active area of research with the potential to significantly reduce game development costs as well as create game experiences meaningfully personalized to each player. Evolutionary methods are a promising method of generating content procedurally. In particular asynchronous evolution of AI agents in an artificial life (A-life) setting is notably similar to the online evolution of non-playable characters in a video game. In this paper, we are concerned with improving the efficiency of evolution via more effective mate selection. In the spirit of PCG, we genetically encode each agent's preference for mating partners and thereby allowing the mate-selection process to evolve. We evaluate this approach in a simple predator-prey A-life environment and demonstrate that the ability to evolve a per-agent mate-selection preference function indeed significantly increases the extinction time of the population. Additionally, an inspection of the evolved preference function parameters shows that agents evolve to favor mates who have survival traits.**

***Index Terms*—artificial intelligence, character generation, evolutionary computation, machine learning, multi-agent systems, reinforcement learning**

I. Introduction

Procedural content generation (PCG) has the potential to reduce game-development costs as well as tailor the game content to each individual player [1]–[6]. Evolutionary search has been used for procedurally generating appearances [7] and even behaviour policies of non-playable characters (NPCs) [8], [9]. Artificial life (A-life) often combines several machine-learning methods in an asynchronous manner [10] representative of an game-time evolution of NPCs [11], [12].

Developing artificial intelligence (AI) in video games can also be viewed as a stepping stone towards the development of AI at large [13]. Indeed, our modern world is increasingly becoming saturated with smart, interconnected, AI-driven devices. With the emergence of vast networks of such agents comes the potential of interesting emergent behavior among them. This potential raises important questions. What will this behavior look like? Will only simple behavior develop or could we see something complex emerging? How will this impact our society? A-life simulations are one way to study such phenomena. A-life allows the study of both single- and multi-agent learning at multiple timescales: within an agent's lifetime (e.g., via reinforcement learning), as well as across generations of agents via genetic search [10]. The potential of emergent collective behavior is a powerful provision of A-life since even simple agents acting according to simple rules can produce a population that exhibits complex behavior (e.g., Conway's Game of Life [14] and Wolfram's cellular automaton rule 110 [15]).

Making evolution more efficient is a key problem especially when the evolution of NPCs happens during playtime and has to be visibly responsive to a player's actions [9], [11]. One technique for increasing efficiency of evolution is better mate selection which is critical for ensuring the diversity of the gene pool as well as protecting innovation as it matures. For instance, the NEAT algorithm uses speciation as a part of its mating strategy [16]. One could port speciation and other hand-designed mate-selection strategies to A-life.[1] However, in the spirit of procedural content generation, we propose a different approach: we genetically encode a preference function in each agent thus allowing evolution to find good mate-selection strategies. Since the preference function is specific to each agent, our approach allows for evolving an ensemble of complementary mate-preference strategies in a population. We implement and evaluate our approach in an A-life setting for the reasons listed earlier in this section. We note that a similar approach can be applied in a synchronous evolution scenario with discrete generations such as a scenario where NEAT would be applied.

Note that this paper is an extended version of a two-page abstract [17].

II. Problem Formulation

We investigate how learning mate-selection strategies can increase the efficiency of evolutionary search. When using PCG techniques to evolve NPCs in video games, the evolutionary efficiency of the method used can be critical. For example, in Darwin's Demons NPC opponents are evolved based on what was challenging for the player [11]. If the evolution of the NPCs is too slow then the novelty of the evolved opponents would be lost. In addition to potentially providing

[0]Co-authors contributed equally.

[1]While A-life eliminates the basic genetic search's need for an explicitly designed fitness function (in A-life the fitness function is implicitly induced by the physics of the environment), it still requires a mechanism for mate selection for sexual agent reproduction.

978-1-7281-1884-0/19/$31.00 ©2019 IEEE

better evolutionary efficiency, learning rather than hand-coding mate-selection strategies is desirable as it reduces the potential burden on the individuals employing these methods, further increasing their applicability to games.

In this work, population extinction time serves as a proxy for evolutionary efficiency such that a high average population extinction time for one method would indicate that the evolutionary efficiency of that method is high. Our *objective function* T maps solution methods for mate-selection to the expected lifetime of the resulting population. Our *goal* is to maximize T. Thus, in our experiments, we test empirically whether using different mate selection strategies results in changes in an estimate of the expected survival time of a population. To get a sense of the robustness of mate selection choices, we also look to the distribution of survival times of populations when using different mate selection strategies. The hypothesis is that when agents can decide which other agents to mate with (i.e., when they are afforded the ability to learn a mate-selection strategy), the average population survival time increases.

III. Related Work

Previous work empirically evaluated several methods of performing mate selection in conventional genetic algorithms [18]. However, to the best of our knowledge, none of the evaluated methods examined involves learning to perform mate selection. If potential parents can predict the fitness of their offspring, then they can learn to use that information in selecting their mates [19]. However, obtaining a reliable such prediction may be difficult at the agent level.

Mate preference on the basis of songs has been explored [20]. Males' genomes are either their song itself or a neural network that generates songs. Females' genomes are transition matrices which rank the aesthetic quality of different note transitions. Mating occurs synchronously, and there are no survival pressures on the agents; males only seek to be appealing to females which may limit the applicability of this approach to an A-life environment.

Related to the idea of mate selection is the study of cooperation. In a previous study [21] agents on a lattice are forced to interact with other agents in their vicinity. However, the agents they interact with may cooperate with them or may defect. The authors show that when agents simply move away from partners that defect and cooperate with partners that cooperate with them, they are able to successfully increase the amount of time they spend with cooperative partners.

Also related to cooperation, previous work [22] also studied how humans select partners in a real-world cooperation-based game. It demonstrated that, in the context of the game, humans will put a considerable amount of effort into finding cooperative partners and that humans will put more effort into finding partners when the selection process is unidirectional rather than bidirectional.

IV. Proposed Approach

We will now present our solution approach, starting with some additional definitions and notation, followed by a short intuition describing the fundamental motivations behind our approach and then the specific algorithmic details.

A. Definitions and Notation

Here, we continue our discussion in Section II with additional definitions and notation for our solution method. Let us first generally consider how an agent behaves as mating is one of many agent behaviours. Biologically, agents can generally be described by their genomes. So, mathematically, suppose we look at an *agent* n from a *population* A. Let its genome be denoted $G_n \in G$, the *genome space* where all agents' genomes lie. G_n can be viewed as a vector that contains features that parameterize agent n's behaviour among other things like its appearance. The particularly important part here in terms of learning and evolution is that the behaviour policy of the agent is genetically encoded.

Now, genomes are much too large for general decision making in social interactions and are not directly observable. In selecting mates, humans for example describe themselves and observe others only through condensed *profiles* as opposed to an entire genome.

Suppose agents can each express themselves though their own *profile function* $\text{Prof} : G \rightarrow \Lambda$ where Λ is the set of all profiles. One view of mate selection or relationship assessment is to think about agents having the need to combine information from the profiles of other agents and information from their own profiles to figure out their compatibility. We define a *merging function* $\text{Mer} : \Lambda \times \Lambda \rightarrow M$ where M is an arbitrary set of vectors of real numbers which contain features about how two agents relate. Finally, for an agents' decision making about whether to mate, we let each agent have a *preference function* $\text{Pref} : M \rightarrow \mathbb{R}$ where the scalar represents the preference an agent has for a particular combination of agents (through the proxy of a vector describing features about how the two agents' profiles relate.)

In this work, we particularly consider situations where agents evaluate combinations with itself for the purpose of mating. Similar to allowing agents to design their own reward functions [10], giving agents a way to select mates can help in maximizing population survival time. With this in mind, we focus on the preference network and evolve it over time, while considering different merging functions and fixing the profile function. We consider parameterized neural networks for the preference networks, use relevant portions of genomes as profiles and consider various merging functions based on viewing it as a function of two vectors (profiles) that we either might want to be similar (to exploit similarities that might be useful for longer living offspring) or dissimilar (to be more diverse and consider how incorporating differences might be beneficial).

There are two intentional limitations in the problem formulation above. First, the preference function operates on information from two profiles and does not consider other attributes such as what the agents may have learned during their lifetimes such as their action functions in Evolutionary Reinforcment Learning [10]. Second, we do not allow choice

in how agents present themselves to other agents (by fixing the profile functions). For instance, different agents cannot sing different songs to other agents to attract them as it happens with real-life birds or certain computational simulations [20]. Future work will lift both limitations.

B. Intuition

Mate-selection affects learning across generations as the choices of whether to mate or not affects the composition of future populations and hence the mate-selection strategy can impact the overall survival time of populations. Instead of hand-coding a mate selection policy into A-life agents we propose to find it via genetic search. We do so by giving each agent a preference function which maps merged profiles of agents and candidate mates to a preference value. The function is encoded genetically in each agent's genes. This allows it to (i) be agent-specific and (ii) evolve over time. We conjecture that a preference function that leads to longer surviving offspring gives such individuals an advantage and thus will spread in the population and increase the population survival time relative to a baseline mate-selection mechanism.

C. Algorithmic Details

ERL Background. We formulate our agents and the environment in an Evolutionary Reinforcement Learning (ERL) setting, similar to the original [10]. The genomes that encode agent behaviour consist of the initial weights for an action function Q_μ and the weights for an evaluation function V_θ where the subscripts μ and θ denote the weights that parameterize functions Q and V.

$V_\theta(s)$ serves as an estimate for the value of state $s \in S$ (i.e., the expected cumulative reward given that an agent starts from s). This evaluation function is learned using evolution. Thus, θ and hence V_θ remains the same during the lifetime of an agent and only change as crossover and mutation operations cause offspring to be born with differing θ's.

On the other hand, Q_μ changes over an agent's lifetime. Particularly, agents learn Q_μ using an RL algorithm where the reward for a transition from state s to state s' when taking action a is given by $V_\theta(s') - V_\theta(s)$. The role of Q_μ can be thought of as encouraging action a in state s if $V_\theta(s') > V_\theta(s)$.

In A-life, while there is no explicit fitness function we define, by virtue of the environment being designed with certain dynamics, we can induce various desired behaviors. For example, in a predator-prey setting where the prey possess genomes encoding their behavior, following Darwin's theory of evolution [23], the fittest agents have the most chances to reproduce, and their evaluation functions will consequently be the ones most seen in the population. The resulting effect is, as time goes on, most agents will learn evaluation functions whose corresponding reward functions, if maximized correctly by all agents, causes the average population survival time to increase.

ERL A-life Simulation. Algorithm 1 describes the simulation for agents with preferential mate-selection capabilities. While these agents are the prey in predator-prey environments in our experiments, the general idea can be extended beyond such environments.

Algorithm 1: A-life Environment Simulation of Agents with Preferential Mate Selection

Input : number_of_initial_agents, max_energy, world_size, Mer, filter, epsilon
Output : (Int) population_survival_time

1 $N_a \leftarrow$ number_initial_agents, $t \leftarrow 0, A \leftarrow \emptyset$
2 randomly initialize networks: $V_\theta, Q_\mu, \text{Pref}_\rho$
3 *genome* $\leftarrow [\theta, \mu, \rho]$
4 *location* $\leftarrow$ random_location_in_world(world_size)
5 *agent* $\leftarrow$ Agent.init(genome, max_energy, location)
6 $A \leftarrow A \cup \{agent\}$
7 **while** $N_a > 0$ **do**
8 **for** $n^m \in A$ **do**
9 $s, n^o \leftarrow$ Env.get_state(n^m, filter(n^m, ...))
10 $\text{Prof}^m \leftarrow [\theta_m, \mu_m]$
11 $\text{Prof}^o \leftarrow [\theta_o, \mu_o]$
12 other_agent_pref $\leftarrow$ $n^m.\text{Pref}_\rho(\text{Mer}(\text{Prof}^m, \text{Prof}^o))$
13 $q_0, ..., q_{|\mathcal{A}|} \leftarrow n^m.Q_\mu([s, \text{other\_agent\_pref}])$
14 $a \leftarrow$ EpsGreedy($[q_0, ..., q_{|\mathcal{A}|}], \epsilon$)
15 **if** $a =$ Action Mate **then**
16 genome $\leftarrow$ Crossover(n^m.genome, n^o.genome) + Mutation()
17 $A \leftarrow A \cup$ Agent.init(genome, max_energy, location_nearby(n^m.location))
18 $N_a \leftarrow N_a + 1$
19 $s' \leftarrow$ Env.step(n^m, a)
20 $V_t \leftarrow n^m.V_\theta([s, \text{other\_agent\_pref}])$
21 $n^m.Q_\mu$.train($s, s', a, V_t - V_{t-1}$)
22 **for** $a \in A$ **do**
23 **if** a.energy ≤ 0 **then**
24 $N_a \leftarrow N_a - 1$
25 $A \leftarrow A \setminus \{a\}$
26 $t \leftarrow t + 1$
27 **return** t

First, we initialize N_a with the initial number of agents given as input to the algorithm. This variable is used to keep track of the number of agents. Then, we perform a loop to initialize the initial agents. Within this loop, we first initialize parametric functions for state evaluation, action selection and preference evaluation (line 2). Given that we use neural networks to parameterize these functions in our experiments, we refer to V_θ, Q_μ and Pref_ρ as the evaluation network, action network and preference network respectively. Function V_θ, the *evaluation network*, outputs a scalar given a state, $V_\theta : \mathcal{S} \rightarrow \mathbb{R}$, quantifying how good it is to be in a given state through a prediction of expected cumulative reward.

The *action network* Q_μ defines the agent's state-action value function and outputs a scalar for each action: $Q_\mu : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ that quantifies how good it is to to take action a in state s and thereafter follow the policy Q_μ, again through a prediction of the expected cumulative reward. Note that here the network outputs the action-value for all the actions given a state, thereby avoiding the need to do multiple forward passes. Finally, the *preference network* Pref_ρ outputs a scalar preference value given information about candidate agents through their profiles, in particular outputs of the merging function where one argument is their own profile. In particular, the myself agent n^m computes $\text{Pref}_\rho(\text{Mer}(\text{Prof}^m, \text{Prof}^o))$ using its preference network. Here, Prof^m is the myself agent's profile and Prof^o is the other agent's profile. The merging functions, Mer, we use are handcrafted distance functions like norms. They are described in more detail in Section V-B.

We initialize the genome of the agent with the weights of the different networks in line 3. As in the original ERL formulation [10], the evaluation network stays fixed over an agent's lifetime whereas the action network changes its weights as the agent learns by interacting with the environment. The preference network does not change during and agent's lifetime.

Initialization of the environment is completed in lines 4-6 by setting the agents to random locations and giving them a predefined maximum amount of energy. Next, the main evolutionary loop begins where agents observe and act in the world. Let agent n^m be the agent being considered in the loop. We first query the environment to get the state for agent n^m and a candidate agent based on a filter function (line 9). Without the loss of generality, the pseudocode assumes a single candidate agent, n^o.

Mate Selection. As formulated in Section II, our preference functions operate on agent profiles. In our experiments, we use profile functions that map to a subset of the genome. In particular, the profiles we use are vectors containing θ and μ, the weights of the value network and the initial weights of the action network respectively.[2] Hence, the merging function takes two vectors of length $|\mu| + |\theta|$ and outputs a vector in M which, as noted in Section IV, is a vector space consisting of vectors whose sizes can vary based on the choice of the merging function. The profile computation for the selecting agent n^m and the candidate agent n^o is done in lines 10 and 11. The resulting profiles are Prof^m and Prof^o.

Suppose we have two agents n^m (the "myself" agent) and n^o (the other agent being considered for mating) and let their profiles be Prof^m and Prof^o. If n^m observes the Prof^o and passes it through its preference network Pref_ρ without any transformation to it (i.e., $\text{Mer}(\text{Prof}^o) = \text{Prof}^o$), it could be problematic as Prof^o alone is decoupled from the profile of agent n^m, Prof^m. Conceivably, the difference between a potential mate's profile and the selecting agent's profile is important in deciding whether to mate. Thus, to allow an agent to take its own genome into account when computing its preference for a candidate mate, the merging function takes both profiles Prof^m and Prof^o as inputs. We consider several hand-coded merging functions: Euclidean distance, element-wise squared distance, element-wise absolute difference, and the identity transformation which simply returns the profile of the other agent (i.e., $\text{Mer}(\text{Prof}^m, \text{Prof}^o) = \text{Prof}^o$).

[2]For simplicity, we exclude the weights of the preference function from the profile to avoid second-order preferences.

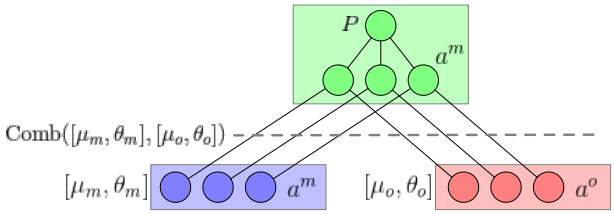

Fig. 1. How one agent calculates its preference for another agent. Here n^m represents weights maintained by the selecting agent and n^o represents weights maintained by the candidate agent.

In the pseudocode the use of this function, Mer, can be found in line 12. The schematic for calculating preferences is illustrated in Figure 1. An agent decides to mate perhaps based on the preference value passed to the state according to the action selection method presented below.

Action Selection and Reinforcement Learning. As the original ERL work [10] we support lifetime learning by updating agent's action-network weights as the agent interacts with the environment. However, instead of their complementary reinforcement backpropagation we use Q-Learning [24]:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a')]$$

where (s,a) is the state-action pair to be updated, s' is the next state, r is the reward, γ is the discount factor, and α is a step-size hyperparameter. To allow for exploration, we use ϵ-greedy action selection strategy whereby a random action is chosen with probability $\epsilon \in (0,1)$ and the greedy action, $\arg\max_a Q(s,a)$ is chosen with probability $1-\epsilon$ (line 14). To compute the reward r we use a difference in state evaluations at consecutive time steps. We use a squared loss as in DQN [25]:

$$\mu \leftarrow \mu - \alpha \nabla_\mu \mathbb{E}\Big[\Big(V_\theta(s') - V_\theta(s) + \\ + \gamma \max_{a'} Q_\mu(s',a') - Q_\mu(s,a)\Big)^2\Big]$$

Evolution. The evolution process takes place when the mate action is chosen by an agent (lines 16 to 18). We crossover the two agents' genomes and perform mutation as well by averaging the genomes of the parents and applying element-wise gaussian noise. We complete the mate action by initializing the agent with a location close to the parent agents, updating the agent set A and incrementing the agent counter.

The simulation keeps track of the energy of the agents, decrementing the agent counter and removing agents from the agent set when an agent's energy falls to 0 (lines 23 – 25).

V. Empirical Evaluation

Experiments are conducted in a wolf-sheep predation model similar to the predator-prey environment used by Ackley and Littman [10] and implemented in NetLogo [26].

A. Simulation Environment

Our tile-based simulation world (61×61 tiles) consists of tiles of grass, sheep and wolves (Figure 2). Grass is initialized with equal probability in either a grown or ungrown state. At initialization, one hundred sheep (shown as white icons) and one wolf (shown as black icons) are distributed throughout the world.

Sheep move around the environment and eat grass. Wolves move around the environment and eat sheep. Each agent always occupies a single tile. Several agents can share a tile. Grass is either present or absent. Grass present in a tile can be eaten by a sheep and will re-grow after a certain number of time steps (drawn uniformly randomly from a pre-specified range). Each sheep and wolf have a level of energy which is reduced with each agent's action except possibly eating. When the energy drops to 0 the agent dies. A wolf attacking a sheep decreases its energy by a certain amount. If the resulting sheep energy drops to 0, the sheep dies and the wolf eats it and replenishes its own energy. No energy is received for attacking a sheep. Wolves reproduce asexually when their energy is above a certain threshold. Sheep, on the other hand, require selecting a mate and taking the mating action to produce an offspring. When mating, both sheep and wolves incur a flat energy cost that is taken from the initiating parent and is used as the offspring's inital energy.

Sheep also die randomly according to a discrete Weibull distribution [27]. Its parameters were adjusted so that even if a sheep were to avoid being eaten, there was approximately 10% probability for a sheep to survive past 500 time steps. When all wolves die a new wolf is added to the simulation at a random location. Thus, there is always at least one wolf in the space, whereas if all sheep die then the simulation is terminated. Wolves follow a simple, fixed behaviour policy the enables them to chase and eat sheep consistently. In contrast, sheep must learn how to eat or to avoid wolves. The learning occurs at two temporal scales: across generations and within an agent's lifetime. Across generations the population evolves better evaluation functions encoded by the evaluation network weights and similarly evolves better initialization values for the action network weights. Recall, we average parents' genomes and then perform mutation by adding normally distributed random values to produce offspring genomes.

Learning within an agent's lifetime is done by updating the action network via Q-learning using evolved evaluation functions. At each time step, a sheep observes its current state and, using its action network, chooses an action to perform. In this simulation, a sheep's state

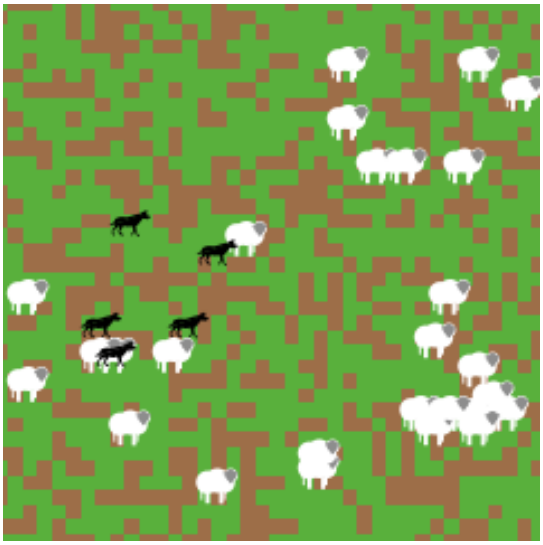

Fig. 2. A-life simulation with sheep, wolves and grass.

$s \in S$ is a nine-tuple of information from its surroundings: $(E, \theta_{\text{Sheep}}, \Delta_{\text{Sheep}}, \theta_{\text{Wolf}}, \Delta_{\text{Wolf}}, \theta_{\text{Grass}}, \Delta_{\text{Grass}}, P, A)$. The elements of this tuple are repsectively: the sheep's current energy; the angle and distance to the closest sheep, wolf, and grass tile; the preference score of the nearest sheep; and the age of the eldest child of the sheep that it has seen. Including this last element provides a rough means by which a sheep is able to gauge its reproductive success.

A sheep can select an action from the action set containing the actions move forwards, turn left, turn right, eat or mate. Performing this action produces a new state which in turn is used to generate a reward using the evaluation network.

We normalize the state features to ensure that the scale of all features falls within reasonable ranges. For example, we convert the angle (in degrees) to the closest entity to the range $[-180, 180]$ and then divide by 180 to get a value between $[-1, 1]$. Similarly, we normalize distances according to the size of the world, energy to $[0, 1]$ using the maximum energy, squash the scalar preferences using the hyperbolic tangent function, and normalize the child age by the oldest age of any sheep in the simulation. Because the world is quite small, sheep have sufficient vision to see any other agent in the world at each tick. However, for simplicity, we artificially blind agents to everything but the closest sheep, wolf, and tile of grass.

B. Baselines for Merging Functions

As baselines in our experiments, we use five hand-coded merging functions. As described earlier, the merging function Mer takes two profiles $\text{Prof}^m, \text{Prof}^o$ as its inputs and returns a vector in M. The baseline is a constant $\text{Mer}_\rho(\text{Prof}^m, \text{Prof}^o) = 0$ (which implies random mate selection), the absolute difference between the profiles $\text{Mer}(\text{Prof}^m, \text{Prof}^o) = |\text{Prof}^m - \text{Prof}^o|$, the squared difference $(\text{Prof}^m - \text{Prof}^o)^2$, the Euclidean distance $\|\text{Prof}_i^m - \text{Prof}^o\|$, and Prof^o itself.

TABLE I
POPULATION SURVIVAL TIMES AVERAGED OVER 125 TRIALS

Preference function	Mean population survival time
Other genome	2589.5 ± 100.0
Absolute difference	2521.0 ± 94.3
Squared difference	2340.4 ± 90.0
Euclidean distance	1878.9 ± 64.5
Random	1761.0 ± 57.2

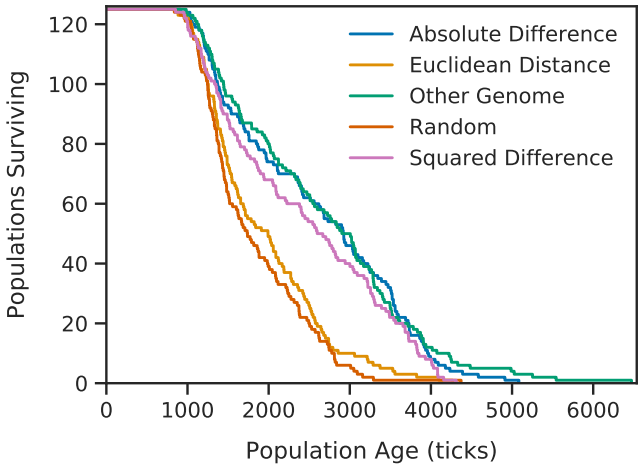

Fig. 3. Populations survival curves.

C. Results: Expected Population Survival Time

Figures 3, 4, and Table I show the results of experiments in which the four different merging functions as well as the random mating baseline were each run 125 times and the simulations were allowed to proceed until extinction (i.e., when the sheep population declines to 0).

Figure 3 shows the survival curves for each preference setting and Figure 4 shows the distribution of the survival times. Notably random mating and Euclidean distance perform quite poorly compared to the other four genome transformation functions, which perform roughly similarly. Table I shows the mean survival times under each of the genome transformation functions. Here there is a significant difference between the average survival time of each of absolute difference, other genome, and squared difference and each of Euclidean distance and random ($p < 0.01$ using two-tailed t-test with Bonferroni corrections). One reason for why using the Euclidean distance as a genome transformation functions performs so poorly is that the preference network under this genome transformation function is restricted to learning a single weight, the preference of an agent for another agent is likely to be dominated by irrelevant features of that agent's genome. The reason that irrelevant features will dominate the distance is that only the relevant features are likely to be selected for and so irrelevant features are likely to have much more variation.

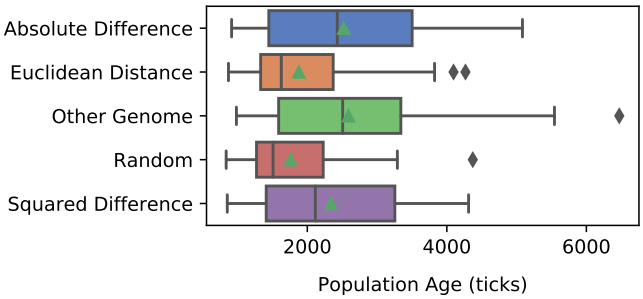

Fig. 4. Distribution of population survival times. Triangles denote mean values. Diamonds denote potential outliers.

D. Results: Preference Weights Analysis

For further analysis we inspected the weights ρ of the preference network Pref_ρ as it evolves. In particular, we looked at the longest run of one of the preferential mating variants that did the best across runs — the other-genome variant where agents only make use of a candidate's genome in calculating preference. A primary reason for this choice is that analysis becomes easier as compared to the other variants because we do not need to consider how a selecting agent's genome affects the preference value.

To see which genome features agents become interested in, we first calculate the unnormalized importance of features by multiplying an agent's preference network weights ρ by the weight from preference P to the Action Mate in the action network Q_μ and take the absolute values. Doing this, we get a vector $\mathbf{u}$ whose values are unbounded but whose sign is meaningful. In particular, the multiplication by the weight from the preference P to action mate in the preference network accounts for the fact that certain agents might weigh the preference in a negative manner, giving a high probability of mating when the preference is a large negative number.

We then normalize the positive and negative elements of the vector $\mathbf{u}$ separately by dividing the negative elements by the negative element with the largest magnitude and similarly for the positive elements. Thus, we get a normalized vector whose elements are between -1 and 1. By averaging such vectors over all agents in a generation, we get a vector of the average normalized weights for all the features of agents in a given generation. Here, the generation number of an individual is defined to be the maximum of the generation numbers of its parents plus one. This choice is motivated by noting that we can expect that agents with a higher ancestral order (i.e., sheep who have more ancestors), will survive better. The use of such discrete generations is also useful when comparing or consolidating agents across multiple runs as we do later. Taking the per-feature incremental average of the normalized $\mathbf{u}$ vectors across the generations and selecting the top-5 features at the last generation in the positive and negative weighting case separately then results in Figures 5 and 6 respectively.

As can be seen in Figure 5, there appear to be certain features that become positively correlated or, in other words,

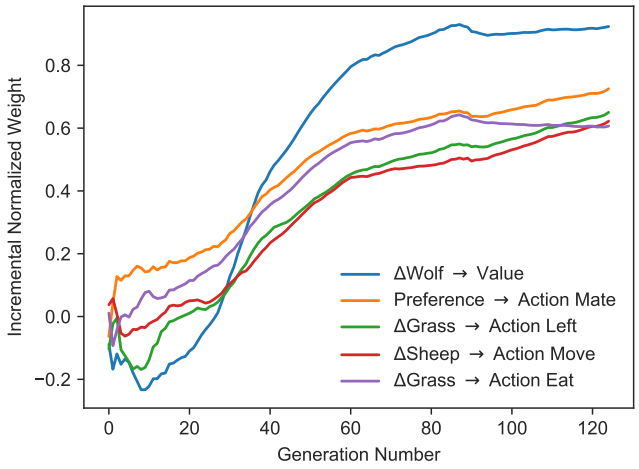

Fig. 5. Average normalized weight for the eventual top five most positively weighted genome features. Here the generation number of an individual is defined to be the maximum of the generation numbers of its parents plus one.

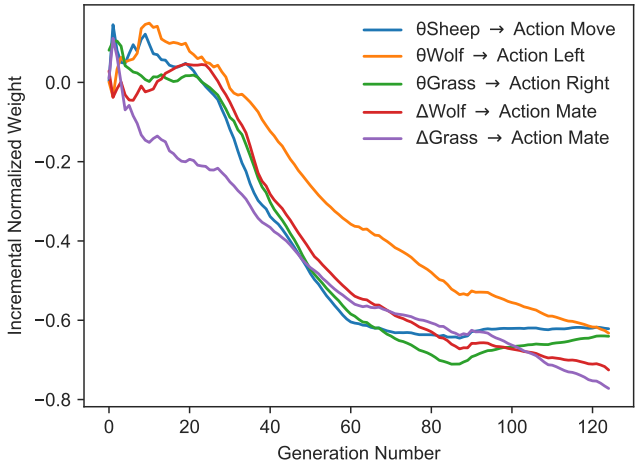

Fig. 6. Average normalized weight for the eventual top five most negatively weighted genome features.

preferred for among agents. For example, sheep learn, on average, to prefer other sheep who prefer being far from the closest wolf. This is because ΔWolf refers to the distance to the closest wolf and ΔWolf → action mate then refers to the weight a sheep gives to how far the closest wolf is.

Given that survival correlates to understanding that being near wolves is dangerous, it is interesting that indeed sheep have understood that in developing future populations, it is important that offspring prefer sheep who give a high value to wolves being far off. In other words, sheep learn to prefer sheep who have a high weight for ΔWolf → Value.

In a similar vein, sheep also learned to prefer mating with sheep who assign a high value to mating when they have a high preference for a candidate agent in their state. This is useful in ensuring that offspring make use of preference in selecting whether to mate and prefer other sheep who have a high weight for Preference → Action Mate.

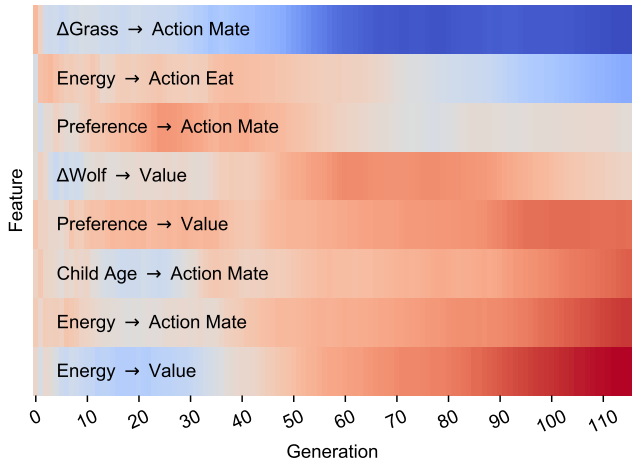

Fig. 7. Average normalized feature weight for meaningful features over the top 30 longest other-genome experiment runs.

Finally, sheep also learned to dislike other sheep who assign a high value to mating when grass is far off. In other words, sheep learn to dislike other sheep who assign a high preference to mate with sheep who have a high weight for ΔGrass → Action Mate. This makes intuitive sense in that mating consumes energy and, moreover, if a wolf were to appear and the closest food source is far off, the sheep could be in danger.

On the other hand, we are unable to explain why some weights being high or low were preferred such as the preference for sheep having a high weight for ΔGrass → Action Left. We note that, indeed, some of the features that become prominent seem to not have simple interpretations. This problem reduces when averaging over multiple runs as different features might end up being weighted more in different runs but any common genome features (i.e., different weights of the parametric function approximators), that sheep learn to prefer will remain prominent.

Finally, we averaged over the top 30 longest runs in the other-genome condition and looked for a set of meaningful features which we could interpret. The results of this investigation are shown in Figure 7. Many preference weights become as we would expect them to be. For example, considering the two ends of feature weight preference, on one end sheep learn to strongly prefer those who have a high value assigned to having a high energy (i.e., sheep who have a high weight for Energy → Value), and, on the other end, strongly disfavor those that more likely take the mate action when the closest grass is far off (i.e., sheep who have a high weight for ΔGrass → Action Mate).

VI. Current Shortcomings & Future Work

In this work, we allowed our agents to examine the genomes of potential mates to create a mating preference. Future work will give agents an ability to present themselves to others

(e.g., via a bird song or an online dating profile). Such profile formation ability can be genetic, co-evolving with their ability to read other profiles.

We prevented our agents from examining each other's preference functions to select a mate. Future work will investigate how including such information affects mate selection.

All our experiments were conducted in an autonomous A-life environment. Future work will implement these techniques in a video game with the player's actions informing the evolution.

VII. Conclusions

Evolutionary search has the potential to procedurally generate interesting non-playable characters in a video game. The A-life setting is particularly applicable to procedural content generation in a video game. We show that the way NPCs in an A-life environment select their mates to form offsprings has an effect on the population survival. In the spirit of PCG, we let evolution develop mate-selection strategies. The resulting evolved strategies increase the population survival time and have elements that make sense from the survival perspective.

Acknowledgment

We appreciate resources provided by Calcul Québec and Compute Canada.

References

[1] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 3, no. 3, pp. 172–186, 2011.

[2] J. Togelius, A. J. Champandard, P. L. Lanzi, M. Mateas, A. Paiva, M. Preuss, and K. O. Stanley, "Procedural content generation: Goals, challenges and actionable steps," in *Dagstuhl Follow-Ups*, vol. 6, 2013.

[3] P. D. Sørensen, J. M. Olsen, and S. Risi, "Breeding a diversity of super mario behaviors through interactive evolution," in *Procceedings of the 2016 IEEE Conference on Computational Intelligence and Games*, 2016, pp. 1–7.

[4] S. Risi and J. Togelius, "Neuroevolution in games: State of the art and open challenges," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 9, no. 1, pp. 25–41, 2017.

[5] A. S. Ruela and F. G. Guimarães, "Procedural generation of non-player characters in massively multiplayer online strategy games," *Soft Computing*, vol. 21, no. 23, pp. 7005–7020, 2017.

[6] G. N. Yannakakis and J. Togelius, *Artificial Intelligence and Games*. Springer, 2018.

[7] S. Risi, J. Lehman, D. B. D'Ambrosio, R. Hall, and K. O. Stanley, "Petalz: Search-based procedural content generation for the casual gamer," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 3, pp. 244–255, 2016.

[8] V. Bulitko, S. Carleton, D. Cormier, D. Sigurdson, and J. Simpson, "Towards positively surprising non-player characters in video games," in *Proceedings of the Experimental AI in Games Workshop at the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2017, pp. 34–40.

[9] V. Bulitko, M. Walters, M. Cselinacz, and M. R. Brown, "Evolving NPC behaviours in A-life with player proxies," in *Joint Proceedings of the AIIDE 2018 Workshops*, 2018.

[10] D. Ackley and M. Littman, "Interactions between learning and evolution," *Artificial life II*, vol. 10, pp. 487–509, 1991.

[11] T. Soule, S. Heck, T. E. Haynes, N. Wood, and B. D. Robison, "Darwin's demons: Does evolution improve the game?" in *Proceedings of the European Conference on the Applications of Evolutionary Computation*, 2017, pp. 435–451.

[12] Polymorphic Games, "Project Hastur," 2018.

[13] M. Buro, "Real-time strategy games: A new AI research challenge," in *Proceedings of the International Joint Conferences on Artificial Intelligence*, 2003, pp. 1534–1535.

[14] M. Gardner, "Mathematical games: The fantastic combinations of John Conway's new solitaire game "life"," *Scientific American*, vol. 223, no. 4, pp. 120–123, 1970.

[15] S. Wolfram, *A new kind of science*. Wolfram Media, 2002.

[16] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary Computation*, vol. 10, no. 2, pp. 99–127, 2002.

[17] D. Ashley, V. Chockalingam, B. Kuzma, and V. Bulitko, "Learning to select mates in artificial life," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019.

[18] C.-F. Huang, "A study of mate selection in genetic algorithms," Ph.D. dissertation, University of Michigan, 2002.

[19] L. M. Guntly and D. R. Tauritz, "Learning individual mating preferences," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2011, pp. 1069–1076.

[20] G. M. Werner and P. M. Todd, "Too many love songs: Sexual selection and the evolution of communication," in *Proceedings of the European Conference on Artificial Life*, 1997, pp. 434–443.

[21] C. A. Aktipis, "Know when to walk away: contingent movement and the evolution of cooperation," *Journal of Theoretical Biology*, vol. 231, no. 2, pp. 249–260, 2004.

[22] G. Coricelli, D. Fehr, and G. Fellner, "Partner selection in public goods experiments," *Journal of Conflict Resolution*, vol. 48, no. 3, pp. 356–378, 2004.

[23] D. Charles, "On the origin of species by means of natural selection," *Murray, London*, 1859.

[24] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] U. Wilensky, "NetLogo," Northwestern University, Evanston, IL, 1999.

[27] T. Nakagawa and S. Osaki, "The discrete Weibull distribution," *IEEE Transactions on Reliability*, vol. 24, no. 5, pp. 300–301, 1975.