The Body is Not a Given: Joint Agent Policy Learning and Morphology Evolution

Dylan Banarse
DeepMind
dylski@google.com

Yoram Bachrach
DeepMind
yorambac@google.com

Siqi Liu
DeepMind
liusiqi@google.com

Guy Lever
DeepMind
guylever@google.com

Nicolas Heess
DeepMind
heess@google.com

Chrisantha Fernando
DeepMind
chrisantha@google.com

Pushmeet Kohli
DeepMind
pushmeet@google.com

Thore Graepel
DeepMind
thore@google.com

ABSTRACT

Reinforcement learning (RL) has proven to be a powerful paradigm for deriving complex behaviors from simple reward signals in a wide range of environments. When applying RL to continuous control agents in simulated physics environments, the body is usually considered to be part of the environment. However, during evolution the physical body of biological organisms and their controlling brains are co-evolved, thus exploring a much larger space of actuator/controller configurations. Put differently, the intelligence does not reside only in the agent's mind, but also in the design of their body. We propose a method for uncovering strong agents, consisting of a good combination of a body and policy, based on combining RL with an evolutionary procedure. Given the resulting agent, we also propose an approach for identifying the body changes that contributed the most to the agent performance. We use the Shapley value from cooperative game theory to find the fair contribution of individual components, taking into account synergies between components. We evaluate our methods in an environment similar to the the recently proposed Robo-Sumo task, where agents in a software physics simulator compete in tipping over their opponent or pushing them out of the arena. Our results show that the proposed methods are indeed capable of generating strong agents, significantly outperforming baselines that focus on optimizing the agent policy alone.

A video is available at: https://youtu.be/CHlecRim9PI

KEYWORDS

Reinforcement Learning; Evolutionary Computation

ACM Reference Format:
Dylan Banarse, Yoram Bachrach, Siqi Liu, Guy Lever, Nicolas Heess, Chrisantha Fernando, Pushmeet Kohli, and Thore Graepel. 2019. The Body is Not a Given: Joint Agent Policy Learning and Morphology Evolution. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, IFAAMAS, 9 pages.

1 INTRODUCTION

Reinforcement Learning (**RL**) uses a simple reward signal to derive complex agent policies, with recent progress on representing the policy using deep neural networks leading to strong results in game playing [32, 41], robotics [26, 27] and dialog systems [28]. Such algorithms were designed for stationary environments, but having multiple learning agents interact yields a non-stationary environment [6, 29]. Various approaches were proposed for continuous control, required for locomotion in software physics simulator environments [1, 19, 34, 37]. Although very successful, such approaches consider the body of the agent to be *fixed*, simply a part of the environment. However, during evolution the physical body of *biological* organisms is constantly changing; thus, the controlling brain and physical body are jointly optimized, exploring a larger space of actuator-controller configurations.

The interaction of evolution with learning by individual animals over their lifetime can result in superior performance [42]. Researchers refer to how individual learning can enhance evolution at the species level as the "Baldwin Effect" [46], where learning guides evolution by smoothing the fitness landscape. In learning agents, the physical shape of the body plays a double role. First, a good body has the capability of effectively exerting many forces in the environment. Second, a well-configured body is *easier to learn to control*, by making it simpler to identify good policies for exerting the forces. Consider a physical task which requires exerting certain forces at the right time, such as locomotion. Some bodies can exert the required forces, while others cannot. Further, some bodies exert the required forces only under a small set of exactly correct policies, whereas others have a wide range of policies under which they exert the required forces (at least approximately). In other words, some bodies have a wide "basin of attraction" where a learner can find a policy that exerts at least a part of the required forces; once discovering a policy in this wide basin, the learner can optimize the policy to exert the required forces. This indicates that the intelligence of agents resides not only in their mind (the controller), but also in the design of their body.

Our contribution: we propose a method for uncovering strong agents, consisting of a good combination of a body and policy. Our

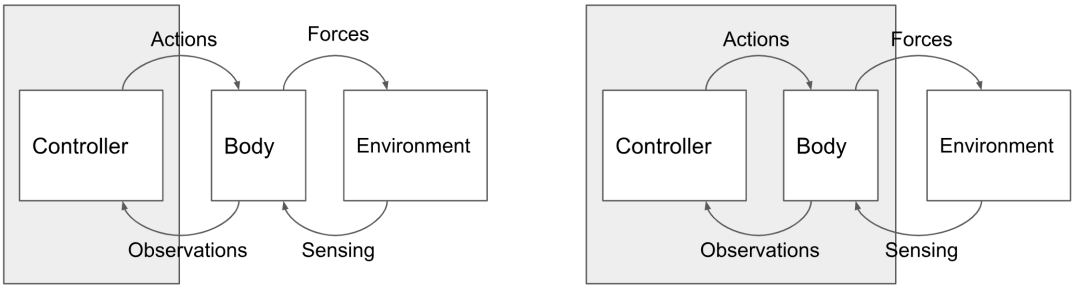


Figure 1: Left: conventional RL optimizes the controller. Right: this work aims to jointly optimize both the controller and the body. Components to be optimized shown in shaded boxes.

approach is based on Policy Optimization while Evolving Morphology, or **POEM** for short. This approach stands in contrast to the traditional paradigm, which takes the body as a given (i.e. a fixed part of the environment), as shown in Figure 1. Our technique relies on applying reinforcement learning algorithm combined with an evolutionary procedure. We also show how to identify the body changes that contributed the most to agent performance, taking into account synergies between them. We demonstrate our method in the Robo-Sumo task [1], where agents in a software 3D physics simulator compete in pushing the opponent out of the arena or tipping it over. This environment is based on the MuJoCo software physics simulator [45], allowing us to easily modify the agent's body. Our results show that the proposed methods are indeed capable of generating superior agents, significantly outperforming baselines that focus on optimizing the agent policy alone.

Related Work. Evolving virtual creatures (EVCs) work uses genetic algorithms to evolve the structure and controllers of virtual creatures in physics simulators, without learning [43]. EVCs have a genetically defined morphology and control system that are co-evolved to perform locomotion tasks [8, 11, 43], with some methods using a voxel-based "soft-body" [8, 21, 31]. Most such attempts have yielded simple behaviors and morphologies [7, 11]. One approach to enable increasingly complex behavior is using a curriculum [10]. Researchers hypothesized that embodied cognition, where a controller expresses its behavior through a body, may cause morphological changes to have an immediate detrimental impact on a behavior [7]. For example, a mutation generating longer legs may harm performance with a controller optimized for shorter legs. This results in pressure to converge on a body design early in evolution, to give the controller a stable platform to optimize. This interdependence can be mitigated by giving the controller time to adapt to morphological changes, so bodies that are easier to learn to control have an evolutionary advantage, and learning would smooth the fitness landscape; this may speed up body evolution, with the extent of learning required for new bodies decreasing over time [42, 46].

Scenarios where learning is used only in the evaluation phase of evolved agents are referred to as Baldwinian evolution [46], where the results of learning are discarded when an offspring is generated. This is in contrast to "Lamarkian evolution" [24, 47], where the result of learning is passed on to offspring. Typically the adaption stage uses a genetic algorithm operating to evolve the controller [7, 24]. In contrast, we use an RL algorithm to learn to control an evolving body. RL has achieved complex behaviours in continuous control tasks with fixed morphology [1, 19, 34, 37], and has the potential to adapt to morphological changes. Our experiments evaluate the potential of evolving the bodies of a population of learning agents. We leverage Population Based Training [22, 23] (PBT), originally proposed to evolve parameters of the controller. To our knowledge, this is the first attempt at evolving the body of continuously controlled RL agents in a software physics simulator.

Preliminaries. We apply **multi-agent reinforcement learning** in partially-observable Markov games (i.e. *partially-observable stochastic games*) [17, 29, 39], in an environment based on a software physics simulator. In every state, agents take actions given partial observations of the true world state, and each agent obtains an individual reward. Through their individual experiences interacting with one another in the environment, agents learn an appropriate behavior policy. More formally, consider an N-player partially observable Markov game $\mathcal{M}$ [29, 39] defined on a finite state set $\mathcal{S}$. An observation function $O : \mathcal{S} \times \{1, \ldots, N\} \rightarrow \mathbb{R}^d$ gives each agent's d-dimensional restricted view of the true state space. On any state, the agents may apply an action from $\mathcal{A}^1, \ldots, \mathcal{A}^N$ (one per agent). Given the joint action $a^1, \ldots, a^N \in \mathcal{A}^1, \ldots, \mathcal{A}^N$ the state changes, following a transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \Delta(\mathcal{S})$ (this is a stochastic transition, and we denote the set of discrete probability distributions over $\mathcal{S}$ as $\Delta(\mathcal{S})$). We use $O^i = \{o^i \mid s \in \mathcal{S}, o^i = O(s, i)\}$ to denote the observation space of agent i. Every agent gets an individual reward $r^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \times \mathcal{A}^N \rightarrow \mathbb{R}$ for player i. Every agent has its own experience in the environment, and independently learns a policy $\pi^i : O^i \rightarrow \Delta(\mathcal{A}^i)$ (denoted $\pi(a^i|o^i)$) given its own observation $o^i = O(s, i)$ and reward $r^i(s, a^1, \ldots, a^N)$). We use the notation $\vec{a} = (a^1, \ldots, a^N)$, $\vec{o} = (o^1, \ldots, o^N)$ and $\vec{\pi}(.|\vec{o}) = (\pi^1(.|o^1), \ldots, \pi^N(.|o^N))$. Every agent attempts to maximize its long term γ-discounted utility:

$$V^i_{\vec{\pi}}(s_0) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r^i(s_t, \vec{a}_t) | \vec{a}_t \sim \vec{\pi}_t, s_{t+1} \sim \mathcal{T}(s_t, \vec{a}_t)\right] \quad (1)$$

Actions and Observations in our Environment: The software physics simulator holds the full true world state, but agents only receive partial observations, in the form of an egocentric view consisting of the positions and velocities of their and their opponent's bodies (end effectors and joints) and distances from the edges

of the pitch. The full list of observed variables include the 3D positions of each end effector of the body, the 3D positions of each joint, the velocities and acceleration of the joints, and distances (on 3 axes) to the corners of the pitch. The agents observe all of these variables for both their own body, and the relative ones of the opponents body. The action space of the agents relates to the actuated hinges. Our agents have multiple limbs, connected by actuated hinges. Our experiments use an "ant" body, with a spherical torso connected to 4 limbs, each having two hinges, one at the "hip" (attaching it to the torso), and one at the "knee". Each of these is a single degree of freedom (DoF) joint, responding to a continuous control signal. The full action space is thus the Cartesian product of the allowed action for each of the hinges (8 hinges in total, with a single DoF each). The above observations and actions are similar to other locomotion tasks based on a physics simulator [19].

Our analysis of the relative importance of the body changes uses **cooperative game theory**. We view the set of body changes as a "team" of players, and quantify the impact of individual components, taking into account synergies between them. Game theory studies players who can form teams, looking for *fair* ways of estimating the impact of individual players in a team. A *cooperative game* consists of a set A of n players, and a *characteristic function* $v : 2^A \rightarrow \mathbb{R}$ which maps any team $C \subseteq A$ of players to a real value, showing the performance of the team as a whole. In our case, A consists of all changes to body components resulting in the final body configuration. The marginal contribution of a player i in a team C that includes it (i.e. $i \in C$) is the change in performance resulting from excluding i: $\Delta_i^C = v(C) - v(C \setminus \{i\})$. We define a similar concept for permutations. Denote by π a permutation of the players (i.e. $\pi : \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, n\}$ where π is a bijection), and by Π the set of all player permutations. We refer to the players occurring before i in the permutation π as the *predecessors* of i in π, and denote by $S_\pi(i)$ the predecessors of i in π, i.e. $S_\pi(i) = \{j | \pi(j) < \pi(i)\}$. The marginal contribution of a player i in a permutation is the change in performance between i's predecessors *and including* i, and the performance of i's predecessors alone: $\Delta_i^\pi = v(S_\pi(i) \cup \{i\}) - v(S_\pi(i))$. The Shapley value [40] is considered a fair allocation of the overall team reward to the individual players in a team, reflecting the contribution of each individual player to the team's success [16, 44]. It is the *unique* value exhibiting various fairness axioms [13, 15], taking into account synergies between agents.

The Shapley value is the marginal contribution of a player, averaged across all player permutations, given by the vector $\phi(v) = (\phi_1(v), \phi_2(v), \ldots, \phi_n(v))$ where:

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(S_\pi(i) \cup \{i\}) - v(S_\pi(i))] \quad (2)$$

2 METHOD

We consider agents who compete with one another in a physical environment, and propose a method for optimizing both the agent's policy and its physical characteristics. We refer to the agent's policy as the *controller*, and to the configuration of its physical structure as the *body*. Our method begins with an initial agent body and a random policy and repeatedly competes agents with each other, identifying beneficial changes to both the agent's policy and the agent's body. Finally, the procedure outputs high performing agents, consisting of both a body configuration and a controller.

Our high level approach combines a reinforcement learning procedure that optimizes each agent's controller with an evolutionary procedure that optimizes the agent's body. We thus simultaneously improve the agents' bodies, while improving and fitting each agent's controller to its current body. Specifically, we use a variant of Population Based Training (PBT) [22, 23], which maintains a *population* of RL agents and leverages an evolutionary procedure to improve their controllers, except we apply evolution to not only the policy learner, but also to the physical agent body. Further, given a final agent body, we *decompose* the overall agent performance to the contribution of each individual body change.

2.1 Our Approach: Policy Optimization while Evolving Morphology (POEM)

POEM maintains a population of agents and lets them participate in contests with each other. It uses the data from the contests in two ways: first, it uses the experience from these episodes to improve the controller by applying RL; second, it analyzes the outcomes of the contests to rank agents by their performance, and uses this ranking to apply an evolutionary process to improve the agents' bodies (and controllers). POEM retains two sub-populations of agents, a *body-fixed* population where only the agent policy is optimized, and a *body-evolving* population, where the agent body as well as the controller are improved over time. The individual agents, in both sub-populations, are continuous policy agents. For the evolutionary procedure, POEM uses a variant of PBT which improves model parameters and learner hyper-parameters (in both body-fixed and body-evolving sub-populations), and also evolves the agent bodies in the body-evolving population.

Controller (Policy Learner): RL Agents: We use continuous control RL agents, based on Stochastic Value Gradients (SVG) [20] and employing off-policy Retrace-correction [33]. SVG is a policy gradient algorithm that learns continuous control policies, allowing for stochastic control by modelling stochasticity in the Bellman equation as a deterministic function of external noise. Our implementation augments SVG with an experience replay buffer for learning the action-value function with k-step returns, applying off-policy Retrace-corrections [33] (several papers cover this in detail [18, 35]). We chose to use SVG rather than alternative learners such as TRPO and PPO [36, 37] as the reparameterization trick it applies typically achieves lower variance estimates, which are particularly important in continuous control settings [20].

Evolutionary Procedure: POEM uses an evolutionary procedure jointly with policy learners to evolve agent bodies and learner parameters, adapting PBT. In PBT, agents play against each other in multiple matches, and Elo ratings [14] are used to measure agents' performance, and "evolve" them, with low-ranked agents copying the parameters of highly-ranked agents. Match episode trajectories are used by the RL algorithm to modify agent policy, and the match outcomes also affect agent fitness ratings, which drive the evolution procedure. The original PBT procedure is designed for "monolithic" agents, but we maintain two sub-populations with an important *asymmetry* between them; the *action space* is the same for all agents, but the outcome of taking the same action depends on the body of

the agent (agents in the body-fixed population are identical, but the body-evolving population agents have different bodies, yielding different outcomes for the same action).

POEM Procedure at a High Level: As discussed above, POEM combines SVG [20] which is a single agent reinforcement learning algorithm with an evolutionary procedure based on PBT [22, 23]. POEM maintains two sub-populations of agents, with the set of body evolving agents denoted as $E = \{e_i\}_{i=1}^{N}$ and the set of body fixed agents denoted as $F = \{f_i\}_{i=1}^{N}$. Each agent i consists of a representation of it physical body characteristics called the *genotype* and denoted as θ_i^b, as well as a neural network representing its policy with parameters (weights) denoted as θ_i, and learner hyper-parameters denoted as θ_i^h (in our case θ_i^h are hyper-parameters of the SVG reinforcement learning procedure, which determine how the agent policy is updated given its experience in previous episodes). Each agent i in both sub-populations E and F is initialized with random policy parameters θ_i and learner hyper-parameters θ_i^h sampled from an initialization distribution. All agents in the body fixed sub-population F are initialized to the same fixed initial body configuration θ_{init}^b, whereas agents in the body-evolving sub-population are initialized with parameters sampled from a body initialization distribution D_{init}^b (and thus have different bodies). In addition to the agent parameters, POEM maintains a **fitness score** for each agent, reflecting its relative ability to win against others. The fitness rating r_i of agent i is initialized to a fixed rating r_{init}.

Following initialization, PBT repeatedly engages agents in matches. A **match-making** method randomly chooses pairs (i, j) of agents to compete with one another. POEM ensures agents from both sub-populations constantly encounter one another, by having each agent face another agent chosen uniformly at random from the *whole population* $E \cup F$. Each match m is an episode trajectory consisting of sequence of the states, actions and rewards of the agents: $m = \{(s_t, a_t^i, a_t^j, r_t^i, r_t^j)\}_{t=1}^{h}$, where s_t denotes the true world state at time t, where a_t^x denote the action taken by agent x at step t and where r_t^x denote the reward obtained by agent x at step t. Following each match, every agent i who participated in the match updates its policy parameters θ_i using the reinforcement learning algorithm; We use SVG [20] to update θ_i given the match trajectory m (and the hyper-parameters θ_i^h). Given the outcome of the contest m, we also update the fitness ratings of the participating agents i, j, using a **rating update** procedure (POEM applies Elo updates [14]).

Following matches between agents, which result in updates to the agent policy and ratings, we employ an evolutionary procedure to adapt agent bodies, policy parameters and learner hyper-parameters. In POEM, the evolution procedure differs between the two sub-populations; agents in both E and F use the procedure to periodically copy policy parameters θ_i and copy and perturb learner hyper-parameters θ_i^h, but only the body-evolving agents E also evolve the body parameters θ_i^b. To guarantee that parameters are not updated too frequently, the procedure first checks that the agent is **evolution-eligible**, using a test based on the number of steps since their parameters were last updated. Each agent i eligible for evolution is compared against a peer j randomly chosen from the population, using a *selection* procedure, which compares i's rating against j's, and only evolves i if j's rating exceeds i's by a certain threshold. If i is selected to evolve, we apply an **inheritance** procedure where i copies j's policy and body and a random subset of learner hyper-parameters, and a **mutation** procedure which randomly perturbs the parameters within some bounds.

The high level description of POEM is given in Algorithm 1. Below we describe each of the sub-procedures: measuring fitness, determining which agents to evolve (Evolution-Eligible and Selection), and how to evolve agents (Inherit and Mutate).

Algorithm 1 POEM PBT Procedure

```
E: population of body-evolving agents
F: population of body-fixed agents
procedure POEM-PBT
    for agent a_i in population E ∪ F do (Initialize)
        Initialize rating r_i ← r_init
        Initialize policy and learner hyper-parameters θ_i, θ_i^h
    end for
    for agent a_i in F do
        Initialize fixed body θ_i^b ← θ_init^b
    end for
    for agent a_i in E do
        Initialize random body θ_i^b ~ D_init^b
    end for
    while true do
        Agents play matches M = {m_x = (i, j)}_{x=1}^{q}
        for match m = (i, j) ∈ M do
            Apply SVG to update θ_i, θ_j using m's trajectory
            Update fitness scores r_i, r_j using m's results (Elo)
        end for
        for agent a_i in population do (Evolution)
            if Evolution-Eligible(a_i) then
                Choose random target a_j
                if Selection(a_i, a_j) then
                    Inherit(a_i, a_j)
                    Mutate(a_i)
                end if
            end if
        end for
    end while
end procedure
```

POEM Sub-Procedures: We now describe the sub-procedures of Algorithm 1. First, note that the body and learner hyper-parameters θ_i^b, θ_i^h are only updated through evolution (the Inherit and Mutate sub-procedures), whereas the policy parameters θ_i are updated both through the RL procedure (SVG) and through evolution.

Fitness: Our procedure uses fitness scores $\{r_i\}_{i=1}^{N}$ to drive evolution, as ratings are used to decide which agents are replaced by mutated copies of others. Following the original PBT work, we use the Elo rating system [14] which was originally introduced to rate chess players. The fitness ratings are based on the outcomes of a set M of past matches, as given in Algorithm 2.

Evolution eligibility: Agents are examined using *evolution eligibility criteria* to avoid early convergence of parameters. Initially there is a warm-up period, during which only the RL learner is used and no evolution steps are performed. Following the warm-up period, agents are only considered for evolution if they meet these

Algorithm 2 Iterative Elo rating update.

Initialize rating r_i for each agent in the agent population.
K: step size of Elo rating update given one match result.
s_i, s_j: score for agent i, j in a given match.
procedure UpdateRating(r_i, r_j, s_i, s_j)
    $s \leftarrow (sign(s_i - s_j) + 1)/2$
    $s_{elo} \leftarrow 1/(1 + 10^{(r_j - r_i)/400})$
    $r_i \leftarrow r_i + K(s - s_{elo})$
    $r_j \leftarrow r_j - K(s - s_{elo})$
end procedure

criteria: a certain number of steps must have passed since they last became eligible for evolution, and a certain number of steps must have passed since their parameters were modified by the evolution. In our experiments, we use a warm-up period of 1×10^8 frames, and require 4×10^6 to have passed since last evolution step.

Selection: Not every agent eligible for evolution immediately modifies its parameters: eligible agents are examined using a *selection procedure* to determine whether the agent would modify its parameters. Each eligible agent i is compared to another agent j sampled uniformly at random from the sub-population, and the ratings are used to compute $s_{i,j}$, the probability of agent i to win in a contest against j. If this probability (win-rate) is lower than a certain threshold, an agent undergoes inheritance and mutation. Formally, for each sub-population E, F, recipient-donor pairs (i, j) are uniformly sampled from the evolution eligible agents. The Elo ratings r_i, r_j are used to compute $s_{i,j}$; if $s_{i,j} > t$ (for a threshold t) then the recipient i will be undergo inheritance and mutation, making it more similar to j. We use a threshold of $t = 45\%$.

Inheritance: POEM uses an *inheritance* procedure to modify i's configuration to be more similar to j's, affecting three types of parameters: policy parameters (neural network weights), learner hyper-parameters, and body configuration parameters. Neural network parameters and body configuration parameters are set to the target j's configuration. Each hyper-parameter is taken either from the evolving agent i or from the target j depending on a (possibly-biased) coin-flip. The inheritance procedure is given in Algorithm 3.

Algorithm 3 Agent i inherits from agent j by cross-over.

Agent i, j with respective network parameters θ_i, θ_j, hyper-parameters θ_i^h, θ_j^h, and body configuration parameters θ_i^b, θ_j^b.
procedure Inherit($\theta_i, \theta_j, \theta_i^h, \theta_j^h, \theta_i^b, \theta_j^b$)
    $\theta_i \leftarrow \theta_j$
    $\theta_i^b \leftarrow \theta_j^b$
    $m \sim \mathcal{B}(0.5)$
    $\theta_i^h \leftarrow m\theta_i^h + (1 - m)\theta_j^h$
end procedure

Mutation: After inheritance, parameters undergo a mutation procedure, which multiplies each parameter by a factor sampled uniformly at random from the range $[1-m, 1+m]$ (we use $m = 0.01$), but maintains caps for each parameter. The caps avoid the body-evolving morphology from diverging too far from the body-fixed morphology. We impose upper and lower bounds on each parameter at ±25% of the parameter's value in the body-fixed configuration. The mutation procedure is given in Algorithm 4.

Algorithm 4 Mutate

P set of mutable parameters in genotype
m mutation level
procedure Mutate(P, m)
    for mutable parameter p_i in P **do**
        b_u upper bound for p_i
        b_l lower bound for p_i
        $r \sim \mathcal{U}(1 - m, 1 + m)$
        $p_i \leftarrow p_i \cdot r$
        if $p_i > b_u$ **then**
            $p_i \leftarrow b_u$
        end if
        if $p_i < b_l$ **then**
            $p_i \leftarrow b_l$
        end if
    end for
end procedure

Body-Evolving Random Initialization: The PBT procedure in Algorithm 1 samples a body configuration θ_i^b for each agent i in the body-evolving population E from the distribution D_{init}^b. In our implementation, we sampled body configurations around the body-fixed configuration: we sample each parameter θ_i^b uniformly at random between the caps used for the mutation procedure.

3 EXPERIMENTS

We now describe our experiments for evaluating POEM, examining several research questions. First, does POEM allow obtaining high performing agents (in controller and body)? Second, is the advantage achieved by the resulting agent due solely to their improved body, or does the process allow us to obtain superior controllers even for the original agent body? Finally, which body changes are most influential in achieving an improved performance?

Environment: Our experiments involve contests between agents, conducted using the MuJoCo software physics simulator [45]. We focus on the Robo-Sumo task [1], where ant shaped robots must tip their opponent over or force them out of the arena.

Agent Body: We use a quadruped body, which we refer to as the "ant body", an example of which is shown in Figure 4a. The body is composed of a root sphere and 8 capsules (cylinders capped by hemispheres) connected via hinges (single DoF joints), each of which are actuated. All rigid bodies have unit density.

In our experiments, the morphology is represented as a graph-based genotype where nodes represent physical components and edges describe relations between them [43]. The morphology is expressed by parsing the genotype. A node describes the shape of a 3D body (sphere or capsule), and the limits of the joint attaching it to its parent (see Figure 2). Edges contain parameters to position, orient and scale child node, shown in Figure 3. Edges have a "reflection" parameter to express symmetry; when enabled, the body of the child is cloned, reflected across the parent's Z-Y plane.

A schematic description of the ant's body is shown in Figure 4b. The genotype for our ant consists of three nodes and three edges,

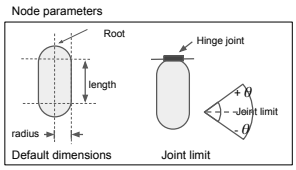


Figure 2: Nodes

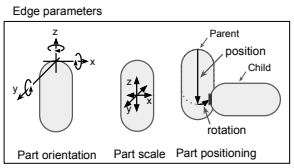


Figure 3: Edges

shown in Figure 4c. The root node specifies the spherical torso, with two edges connected to an "upper leg" node, one for the upper segment of the rear legs, and one for the front legs. The lower segments of the ant's legs are all specified by the same "lower leg" node. The full body structure is determined by 25 parameters in these nodes and edges. Parameters for the body-fixed ant nodes and edges are shown in Tables 1 and 2.

	Shape	Radius (m)	Length (m)	Joint Limit (rad)
Torso	Sphere	0.2	n/a	n/a
Upper leg	Capsule	0.08	0.28	0.52
Lower leg	Capsule	0.08	0.56	0.35

Table 1: Node parameters

	x rot	y rot	z rot	Scale	Parent pos	Parent rot
Front hip	$\pi/2$	0	$\pi/2$	1.0	0.5	$3\pi/4$
Rear hip	$\pi/2$	0	$\pi/2$	1.0	0.5	$\pi/4$
Knee	$\pi/4$	0	0	1.0	1.0	$-\pi/2$

Table 2: Edge parameters

Population Configuration: We maintain a *body-fixed* and *body-evolving* sub-populations, each consisting of $n = 64$ agents. Both sub-populations are initialized with random policy parameters and the same hyper-parameters. As discussed in Section 2, body-fixed agents are all initialized to the body configuration shown in Figure 4a, and body-evolving agents are each initialized with a different body sampled around the original body configuration. Figure 5 shows example initial bodies for the body-evolving population.

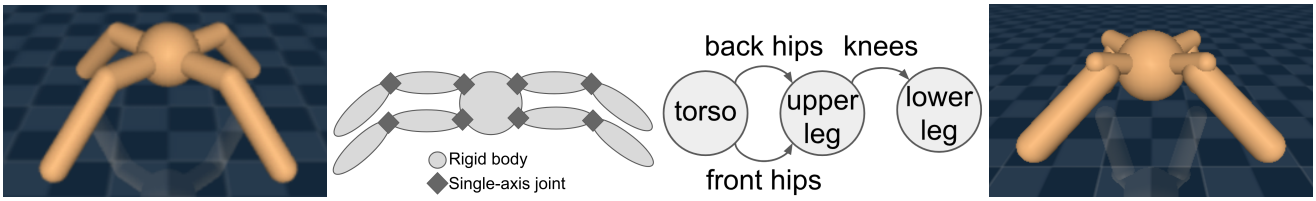
3.1 Comparing Body-Fixed and Body-Evolving Populations

Our experiment is based on data from $k = 70$ runs of the POEM method of Section 2.1, with two sub-populations (a body-fixed and a body-evolving sub-population), each with $n = 64$ agents. POEM matches agents for contests uniformly at random across the entire population, so the body-fixed agents adapt the controller so as to best match the body-evolving agents, making them increasingly stronger opponents. Finding a good controller for the body-evolving population is challenging, as the controller must cope with having many different possible bodies it may control (i.e. it must be robust to changes in the physical body of the agent). We examine agent performance, as reflected by agent Elo scores. Figure 6 shows agent Elo ratings over time, in a typical run, where agents of the body-evolving sub-population outperform the body-fixed agents (body-fixed agents are shown in red, and body-evolving agents in blue). Both populations start with similar Elo scores, but even early in training there is a gap in favor of the body-evolving agents.

To determine whether POEM results in a significant advantage over optimizing only the controller, we study outcomes in all $k = 70$ runs. We run POEM for 36 training hours (equivalently, $1e10$ training steps), and analyze agent performance. At the evaluation time, each agent (in either sub-population) has its own Elo rating, reflecting its win-rate against others. As our goal is to identify the strongest agents, we examine the highest Elo agent in each sub-population. Figure 7 shows a histogram of Elo scores on the run of Figure 6, at evaluation time, showing that in this run all body-evolving agents outperform all body-fixed agents. We examine the proportion of runs where the highest Elo agent is a body-evolving agent (rather than a body-fixed one). In over 95% of the runs, the top body-evolving agent outperforms the top body-fixed agent. A binomial test shows this to be significant at the $p < 0.0001$ level. When one can change physical traits of agents, this shows that POEM can find the configuration of strong agents (a body and matching controller), typically outperforming agents with the original body and a controller optimized for that body. Figure 4d shows an example evolved ant from the body-evolving population. On average the evolved body is wider and heavier, and has a lower center of gravity; the caps on parameters during evolution allow the body to evolve to be even heavier, so the advantage is not only due to mass. Figure 8 shows how one body parameter evolves over time within the population. Initially the variance is high, but by 1e10 steps it is negligible. This shows the population converges early in training, possibly to a sub-optimal body.

Body-Evolving Controllers in the Original Body: POEM uncovers good *combinations* of a body and controller. One might conjecture that the advantage stems from the agent's modified body, rather than from the controller. As the overall structure of the ant remains the same, with only sizes, locations and angles of joints modified, we can use any controller in any body. Thus we can test the performance of the controller discovered for the evolved body in the *original*, unevolved body. We compared the win-rate of the body-fixed population against that of the body-evolving controllers fit into the *unevolved* body. Controllers were taken after 36 hours of training. The results show that in 23% of the runs, the controllers taken from the body-evolving population outperform those of the body-fixed population, *when used in the unevolved body* (similar to recent observations in EVCs by [25]). This shows POEM may find strong controllers even for the original body, and may be useful even when we cannot modify the physical body of agents.

Original Body Controller in the Body-Evolved Body: Similarly, an indication of the performance improvement afforded by the body-evolving body can be gained by coupling it with the original-body optimized controller; as if the body and controller have been optimized separately. For all experiments, we competed the best body-evolving body against the original body-fixed body where both agents use the best body-fixed controller. The results showed a 70% win-rate for the body-evolving body, demonstrating that whilst the body-evolving body does improve performance, jointly optimizing the body and controller provides the best performance.



(a) Unevolved Ant body (b) Ant body schematic (c) Ant genotype (d) An evolved Ant Body

Figure 4: Ant model

Figure 5: Initial bodies in the body-evolving population

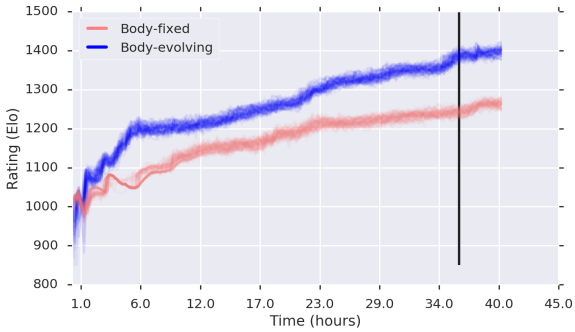


Figure 6: Agent Elo over time (SD=20). Across runs mean body-evolving Elo was 100 higher (SD=60) than body-fixed.

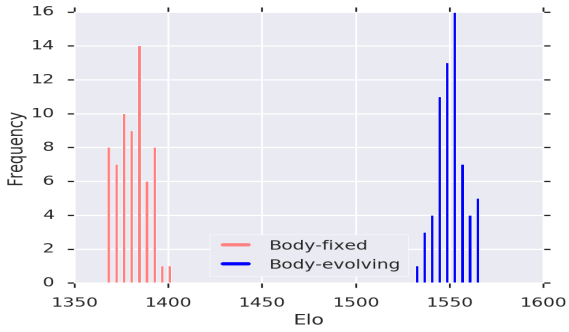


Figure 7: Elo histogram.

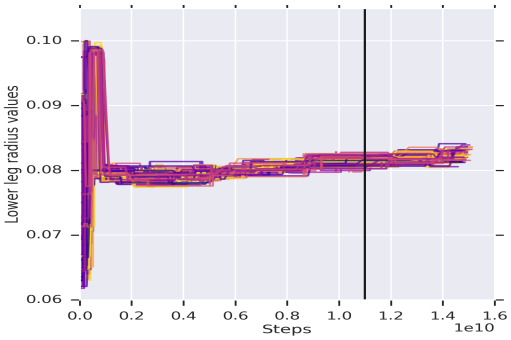


Figure 8: Lower leg radius across agents during training

3.2 Identifying Influential Body Changes

Section 3.1 discusses making multiple body changes to improve performance. But which change had the most impact? The agent's performance isn't simply the sum of the "strengths" of individual changes, as different body components depend on each other. For instance, changing the orientation of the leg may only be helpful when changing its length. We view the set of body changes as a "team" of components, and attempt to fairly attribute the improvement in performance to each of the parts, taking into account synergies between components, using the Shapley value. We define a cooperative game where "players" are the changes to body parts. As we have 25 body configuration parameters, we obtain a cooperative game with 25 players. [1] Our analysis is based on defining a cooperative game where the players are the body parts changes converting the original unevolved body into the final evolved body. Given this game, one can compute Shapley values using the formula in Section 1, to obtain the vector $\phi = (\phi_1, \ldots, \phi_n)$, reflecting the fair contribution of each body change, taking into account the interdependence and synergies between components.

A Motivating Example: consider three possible body changes: a) increase leg length, b) change the leg's angle, c) increase torso size. Suppose that changes a and b *in isolation* each increase the win-rate from 50% to 56%, while applying c *in isolation* increases the win-rate from 50% to 54%. Based solely on this, one might

[1] This is akin to using Shapley values to measure feature importance in prediction [9, 12], or for measuring power in voting and networking domains [3–5, 38].

claim that a and b are more impactful. However, suppose that a and b are *substitutes* so applying *both* increases the win-rate to 56% (i.e. once one has been applied, applying the other change does not further improve the win-rate). In contrast, while applying c in isolation only increases performance by 4%, it is synergetic with a and b, so combined with either a or b, it improves performance by 5%; for instance, applying both a and c result in a win rate of 50% + 6% + 5% = 61%. Finally, applying all three changes (a,b,c) still achieves a win-rate of 61%. As a and b are substitutes, their fair contribution should be lower than one would assume based on applying changes in isolation. Similarly, as c complements the other changes, it contribution should be higher than one would assume based on applying changes in isolation. The Shapley value examines the average marginal contribution (MC) of components in all permutations, as given in Table 3, to reflect these considerations.

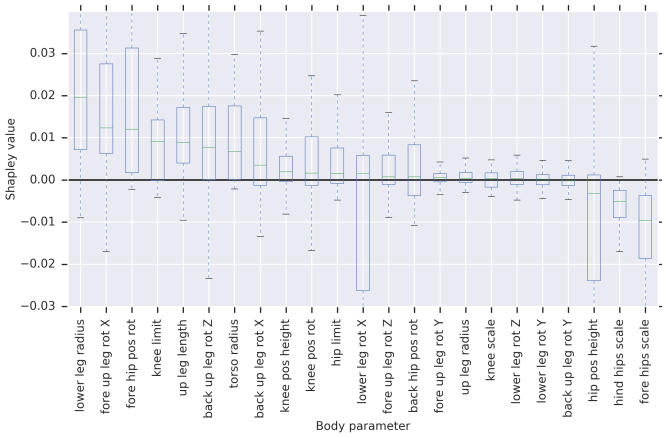
Permutation	MC(a)	MC(b)	MC(c)
abc	0.06	0.0	0.05
acb	0.06	0.0	0.05
bac	0.0	0.06	0.05
bca	0.0	0.06	0.05
cab	0.06	0.0	0.04
cba	0.0	0.06	0.04
average(Shapley)	0.03	0.03	0.0467

Table 3: Shapley computation for hypothetical example

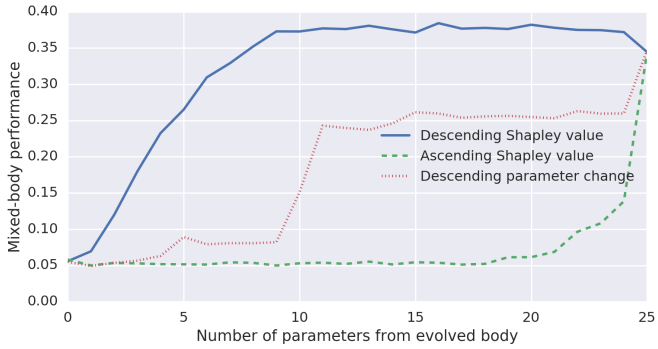
A direct computation of the Shapley value is computationally intractable, so we use an approximation method [2].

The Cooperative Game: We define the value $v(S)$ of a subset S of body changes as follows. Given the original body b and evolved body b' and a set S of body parts, we define the body $b(S)$ as one where each part $p \in S$ takes the configuration as in the evolved body, and where $p \notin S$ takes the configuration as in the unevolved body b. To evaluate the performance of the hybrid body $b(S)$ we use the evolved controller discussed in Section 3.1. Given an (evolved) controller c and a fixed baseline agent d (consisting of a fixed body and a fixed policy), we define the value $v(S)$ of a set S of body changes as the win probability of an agent with the body $b(S)$ and controller (policy) c against the baseline agent d. $v(S)$ defines a cooperative game over the body parts, allowing us to compute the Shapley value of each body part. Figure 9a shows Shapley values measuring relative importance of body changes (for the top body-evolving agent), showing that body changes are unequal in their contribution to agent performance. The high impact parameters are lower leg radius and the rotation of upper leg and hip.

We conduct another experiment to confirm that high Shapley components indeed yield a bigger performance boost than low Shapley ones. We rank body parameters by their Shapley value and use the ranking to incrementally apply evolved-body parameter values to an unevolved body-fixed body. We do this once in decreasing Shapley values order, and a second time in an opposite order. This process generates 26 body variants, where the first variant has no evolved body parameters and the last has all 25. Each body variant competes against a fixed baseline agent (with fixed body and policy) in 25,000 matches to get the proportion of won matches, used as a performance measure. Figure 9b depicts the resulting agent performance. The curves show that introducing the highest Shapley valued parameters first has a large impact on performance. The figure also shows that the Shapley ranking also outperforms another baseline heuristic, which ranks parameters by the magnitude of their change from the unevolved body.



(a) Relative importance of body changes using the Shapley decomposition (single POEM run).



(b) Performance as body parameters are incrementally changed from unevolved to evolved body (descending and ascending Shapley value order, and descending through parameter-heuristic importance).

Figure 9: Shapley analysis of importance of body changes.

4 CONCLUSION

We proposed a method for jointly optimizing agent body and policy, combining continuous control RL with an evolutionary procedure for modifying agent bodies. Our analysis shows that this method can achieve stronger agents than obtained by optimizing the controller alone. We used game theoretic solutions to identify the most influential body changes. Several questions remain open. First, can we augment our procedure to also modify the neural network architecture of the controller, similarly to recent neural architecture optimizers [30]? Second, can we use similar game theoretic methods to *guide* the evolutionary process? Finally, How can we ensure the diversity of agents' bodies so as to improve the final performance?

REFERENCES

[1] Maruan Al-Shedivat, Trapit Bansal, Yuri Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. 2017. Continuous adaptation via meta-learning in nonstationary and competitive environments. *arXiv preprint arXiv:1710.03641* (2017).

[2] Yoram Bachrach, Evangelos Markakis, Ezra Resnick, Ariel D Procaccia, Jeffrey S Rosenschein, and Amin Saberi. 2010. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multi-Agent Systems* 20, 2 (2010), 105–122.

[3] Yoram Bachrach, Reshef Meir, Michal Feldman, and Moshe Tennenholtz. 2012. Solving cooperative reliability games. *UAI* (2012).

[4] Yoram Bachrach and Ely Porat. 2010. Path disruption games. In *AAMAS*.

[5] Yoram Bachrach, Jeffrey S Rosenschein, and Ely Porat. 2008. Power and stability in connectivity games. In *AAMAS*.

[6] D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. (2002), 819–840 pages.

[7] Bongard J. SunSpiral V. Lipson H. Cheney, N. 2016. On the Difficulty of Co-Optimizing Morphology and Control in Evolved Virtual Creatures. *ALIFE XV, The Fifteenth International Conference on the Synthesis and Simulation of Living Systems* (2016).

[8] MacCurdy R. Clune J. Lipson H. Cheney, N. 2013. Unshackling Evolution: Evolving Soft Robots with Multiple Materials and a Powerful Generative Encoding. In *Genetic and Evolutionary Computation Conference (GECCO'13), Amsterdam, The Netherlands*.

[9] Shay B Cohen, Eytan Ruppin, and Gideon Dror. 2005. Feature Selection Based on the Shapley Value.. In *IJCAI*, Vol. 5. 665–670.

[10] Don Fussell Dan Lessin and Risto Miikkulainen. 2014. Adapting Morphology to Multiple Tasks in Evolved Virtual Creatures. In *The Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*.

[11] Risto Miikkulainen Dan Lessin, Don Fussell. 2013. Open-Ended Behavioral Complexity for Evolved Virtual Creatures. In *Genetic and Evolutionary Computation Conference (GECCO'13), Amsterdam, The Netherlands*.

[12] Anupam Datta, Shayak Sen, and Yair Zick. 2016. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems. In *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE, 598–617.

[13] Pradeep Dubey. 1975. On the uniqueness of the Shapley value. *International Journal of Game Theory* 4, 3 (1975), 131–139.

[14] Arpad E. Elo. 1978. *The rating of chessplayers, past and present.* Arco Pub., New York. http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216

[15] Vincent Feltkamp. 1995. Alternative axiomatic characterizations of the Shapley and Banzhaf values. *International Journal of Game Theory* 24, 2 (1995), 179–186.

[16] Faruk Gul. 1989. Bargaining Foundations of Shapley Value. *Econometrica* 57, 1 (1989), 81–95.

[17] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. 2004. Dynamic Programming for Partially Observable Stochastic Games. In *Proceedings of the 19th National Conference on Artifical Intelligence (AAAI'04)*. AAAI Press, 709–715.

[18] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. 2018. Learning an Embedding Space for Transferable Robot Skills. In *International Conference on Learning Representations*.

[19] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. 2017. Emergence of Locomotion Behaviours in Rich Environments. *CoRR* abs/1707.02286 (2017). http://arxiv.org/abs/1707.02286

[20] Nicolas Heess, Gregory Wayne, David Silver, Tim Lillicrap, Tom Erez, and Yuval Tassa. 2015. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*. 2944–2952.

[21] Jonathan Hiller and Hod Lipson. 2012. Automatic Design and Manufacture of Soft Robots. (2012).

[22] Max Jaderberg, Wojciech Czarnecki, Iain Dunning, Luke Marris, Guy Lever, Antonio García Castañeda, Charles Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, David Silver, Demis Hassabis, Koray Kavukcuoglu, and Thore Graepel. 2018. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *CoRR* abs/1807.01281 (2018).

[23] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al. 2017. Population based training of neural networks. *arXiv preprint arXiv:1711.09846* (2017).

[24] Milan Jelisavcic, Rafael Kiesel, Kyrre Glette, Evert Haasdijk, and A. E. Eiben. 2017. Analysis of Lamarckian Evolution in Morphologically Evolving Robots. In *Proceedings of the 14th European Conference on Artificial Life*. MIT Press, 214–221.

[25] M.J. Jelisavcic, D.M. Roijers, and A.E. Eiben. 2018. Analysing the Relative Importance of Robot Brains and Bodies. In *ALIFE 2018 Proceedings of the Artificial Life Conference 2018 (Artificial Life Conference Proceedings)*. MIT Press Journals, United States, 327–334. https://doi.org/10.1162/isal_a_00063

[26] 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.

[27] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. 2016. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research* 17, 1 (2016), 1334–1373.

[28] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. 2016. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541* (2016).

[29] Michael L. Littman. 1994. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 157–163.

[30] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. 2017. Hierarchical representations for efficient architecture search. *arXiv preprint arXiv:1711.00436* (2017).

[31] Borys WrÃąbel MichaÂĆ Joachimczak. 2012. Co-evolution of morphology and control of soft-bodied multicellular animats. In *Genetic and Evolutionary Computation Conference (GECCO'12), Philadelphia, Pennsylvania, USA*.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[33] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. 2016. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*. 1054–1062.

[34] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

[35] Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. 2018. Learning by Playing-Solving Sparse Reward Tasks from Scratch. *arXiv preprint arXiv:1802.10567* (2018).

[36] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International Conference on Machine Learning*. 1889–1897.

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

[38] Abigail See, Yoram Bachrach, and Pushmeet Kohli. 2014. The cost of principles: analyzing power in compatibility weighted voting games. In *AAMAS*.

[39] L. S. Shapley. 1953. Stochastic Games. *Proceedings of the National Academy of Sciences of the United States of America* 39, 10 (1953), 1095–1100.

[40] Lloyd S Shapley. 1953. A value for n-person games. (1953).

[41] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484.

[42] George Gaylord Simpson. 1953. The baldwin effect. *Evolution* 7, 2 (1953), 110–117.

[43] K. Sims. 1994. Evolving virtual creatures. *21st annual conference on Computer graphics and interactive techniques, SIGGRAPH '94* (1994).

[44] P Straffin. 1988. The ShapleyâĂŤShubik and Banzhaf power indices as probabilities. *The Shapley value. Essays in honor of Lloyd S. Shapley* (1988), 71–81.

[45] Emanuel Todorov, Tom Erez, and Yuval Tassa. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 5026–5033.

[46] Bruce H Weber and David J Depew. 2003. *Evolution and learning: The Baldwin effect reconsidered.* Mit Press.

[47] Darrell Whitley, V Scott Gordon, and Keith Mathias. 1994. Lamarckian evolution, the Baldwin effect and function optimization. In *International Conference on Parallel Problem Solving from Nature*. Springer, 5–15.