THE BODY IS NOT A GIVEN: JOINT AGENT POLICY LEARNING AND MORPHOLOGY EVOLUTION

Anonymous authors
Paper under double-blind review

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 3D environment with simulated physics 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. [1]

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 (Mnih et al., 2015; Silver et al., 2016), robotics (Kober et al., 2013; Levine et al., 2016) and dialog systems (Li et al., 2016). Such algorithms were designed for stationary environments, but having multiple learning agents interact yields a non-stationary environment (Littman, 1994; Bernstein et al., 2002). Various approaches were proposed for continuous control, required for locomotion in an physics simulator environment (Heess et al., 2017; Schulman et al., 2017; Rajeswaran et al., 2017; Al-Shedivat et al., 2017). 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 (Simpson, 1953). Researchers refer to how individual learning can enhance evolution at the species level as the "Baldwin Effect" (Weber & Depew, 2003), 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

[1] A video is available at: www.youtube.com/watch?v=eei6Rgom3YY.

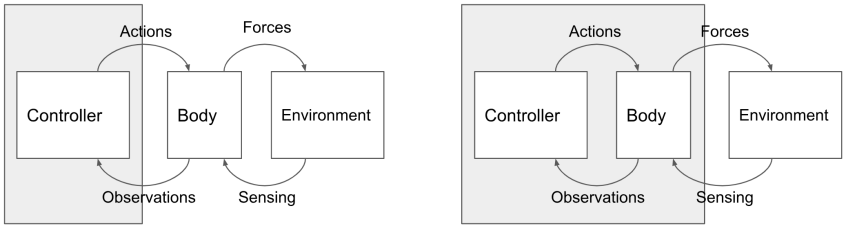


Figure 1: Left: conventional RL optimizes the controller. Right: this work aims to jointly optimize both the controller and the body. Components to be optimized shown in shaded boxes.

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 is proposing a method for uncovering strong agents, consisting of a good combination of a body and policy. This 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 combines RL 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 an environment similar to the Robo-Sumo task (Al-Shedivat et al., 2017), where agents in a 3D environment with simulated physics compete in pushing the opponent out of the arena or tipping it over. This environment is based on the MuJoCo physics simulator (Todorov et al., 2012), 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 physically simulated environments, without learning (Sims, 1994). EVCs have a genetically defined morphology and control system that are co-evolved to perform locomotion tasks (Sims, 1994; Dan Lessin, 2013; Cheney, 2013), with some methods using a voxel-based "soft-body" (Cheney, 2013; Hiller & Lipson, 2012; Micha Joachimczak, 2012). Most such attempts have yielded relatively simple behaviors and morphologies (Dan Lessin, 2013; Cheney, 2016). One approach to enable continually increasing complex behavior is using a curriculum (Dan Lessin & Miikkulainen, 2014). 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 (Cheney, 2016). 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 would 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 (Simpson, 1953; Weber & Depew, 2003).

Scenarios where learning is used only in the evaluation phase of evolved agents are referred to as Baldwinian evolution (Weber & Depew, 2003), where the results of learning are discarded when an offspring is generated. This is in contrast to "Lamarkian evolution" (Whitley et al., 1994; Jelisavcic et al., 2017), where the result of learning is passed on to offspring. Typically the adaption stage uses a genetic algorithm operating to evolve the controller (Cheney, 2016; Jelisavcic et al., 2017). 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 (Heess et al., 2017; Schulman et al., 2017; Rajeswaran et al., 2017; Al-Shedivat et al., 2017), 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 (Jaderberg et al., 2017; 2018) (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 physically simulated environment.

Preliminaries We apply **multi-agent reinforcement learning** in partially-observable Markov games (i.e. *partially-observable stochastic games*) (Shapley, 1953a; Littman, 1994; Hansen et al., 2004). In every state, agents take actions given partial observations of the true world state, and each obtains an individual reward. Agents learn an appropriate behavior policy from past interactions. In our case, given the physics simulator state, agents observe 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. Our agents have actuated hinges (one at the "knee" and one at the "hip" of every limb). The full specification for our environment (observations and actions) are given in the Appendix, and are similar to other simulated physics locomotion tasks (Heess et al., 2017). Every agent has its own experience in the environment, and independently learns a policy, attempting to maximize its long term γ-discounted utility, so learning is *decentralized* (Bernstein et al., 2002).

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 (Shapley, 1953b) is considered a fair allocation of the overall reward achieved by a team to the individual players in a team, reflecting the contribution of each individual player to the team's success (Gul, 1989; Straffin, 1988). It is the *unique* value exhibiting various fairness axioms (Dubey, 1975; Feltkamp, 1995), taking into account synergies between agents (see Section 8 in the Appendix for a detailed discussion and examples of how the Shapley value captures such synergies between body components). 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: $\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \Pi} [v(S_\pi(i) \cup \{i\}) - v(S_\pi(i))]$.

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 (as well as the learner's parameters). We thus simultaneously improve the agents' bodies, while improving and fitting each agent's controller to its current body. Specifically, we employ a variant of Population Based Training (Jaderberg et al., 2017; 2018)(PBT), 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-improving* 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-improving sub-populations), and also evolves the agent bodies in the body-improving population.

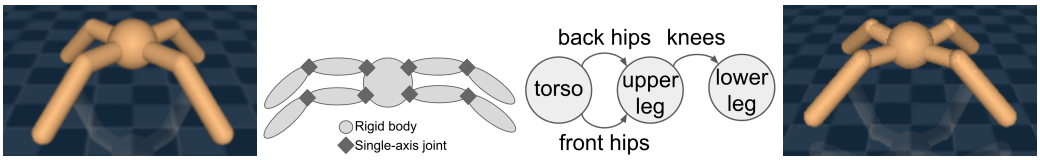
2.1.1 CONTROLLER (POLICY LEARNER): RL AGENTS

We examine continuous control RL agents, based on Stochastic Value Gradients (SVG) (Heess et al., 2015) and employing off-policy Retrace-correction (Munos et al., 2016). 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 (Munos et al., 2016) (several papers cover this in detail (Hausman et al., 2018; Riedmiller et al., 2018)).

2.1.2 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 contests, and Elo ratings (Elo, 1978) are used to measure agents' performance, and "evolve" them, with low-ranked agents copying the parameters of highly-ranked agents. Both the episode traces fed to the policy learner and the agent performance ratings that drive the evolution depend on the tournaments played. 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-improving population agents have different bodies, yielding different outcomes for the same action). POEM ensures that agents from both sub-populations constantly encounter one another, by applying random match-making: each agent faces another agent chosen uniformly at random from the *whole population* in a contest yielding a single episode. However, the evolution procedure differs between the two sub-populations; both use the evolution procedure to periodically copy policy parameters and copy and perturb learner hyperparameters, but only the body-improving agents also evolve the body parameters during evolution.

Evolution Agents in our framework face their peers in multiple contests, so each agent faces opponents that learn and change their policy, and also evolve and change their body. This makes the environment an agent faces highly non-stationary. Further, changes in the agent's *own* body also contribute to the non-stationary nature of the environment. The optimal body configuration and controller thus depend on the configuration of the other agents in the population (i.e. their bodies and policies, which are the result of the learning and evolution procedure). We apply PBT to optimize different parts of the agent configuration, including policy parameters, hyperparameters of the learner, and the shape of the agent's body. We now provide a short description of the PBT procedure we use (full details are given in the Appendix). The high-level pseudocode is given in Algorithm 1, and the subroutines are described below. **Fitness:** PBT uses ratings that quantify agents' relative performance based on the outcomes of previous contests. Following the original PBT work, we use the Elo rating system (Elo, 1978) which was originally introduced to rate chess players (the specific update details of the Elo procedure are given in the Appendix). **Evolution eligibility:** Agents are examined using *evolution eligibility criteria*, designed to avoid early conversion of the agent population. 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 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. **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. **Inheritance:** PEOM uses an *inheritance* procedure to mod-



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

Figure 2: Ant model

ify i's configuration to be more similar to j's, affecting three types of 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 2. **Mutation:** After inheritance, parameters undergo mutation, which modifies each parameter p by a factor f_p following a uniform distribution $f_p \sim \mathcal{U}(1-m, 1+m)$. After mutation, body parameters are capped at their individual upper and lower bounds (see Algorithm 4 in the Appendix for full details).

Algorithm 1 POEM PBT Procedure

A: set of population agents
procedure POEM-PBT
  for agent a_i in population **do**
    Fitness: Rank agents by ability
    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 procedure

Algorithm 2 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($\theta_i, \theta_j, \theta_i^h, \theta_j^h, \theta_i^b, \theta_j^b$)
  $\theta_i \leftarrow \theta_j$
  $\theta_i^b \leftarrow \theta_j^b$
  $\boldsymbol{m} \sim \mathcal{B}(0.5)$
  $\theta_i^h \leftarrow \boldsymbol{m}\theta_i^h + (1-\boldsymbol{m})\theta_j^h$
end procedure

3 EXPERIMENTS

We now describe our experiments for evaluating the performance of the POEM framework. We examine 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 physics simulator (Todorov et al., 2012). We focus on the Robo-Sumo task (Al-Shedivat et al., 2017), 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 2a. 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 the rigid bodies have unit density. A schematic configuration is shown in Figure 2b. In our experiments, the morphology is represented as a directed graph-based genotype where the nodes represent physical component specifications and the edges describe relationships between them (Sims, 1994). An agent's morphology is expressed by parsing its genotype.

Each node describes the shape of a 3D rigid body (sphere or capsule), and the limits of the hinge joint attaching it to its parent (see Figure 3). Edges contain parameters used to position, orient and

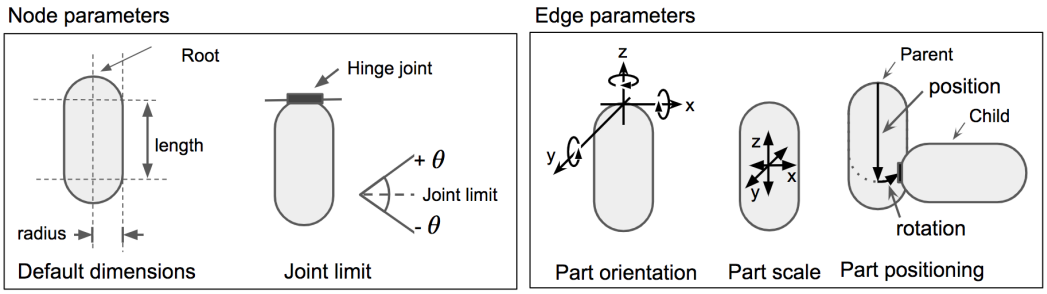


Figure 3: Ant body configuration parameters

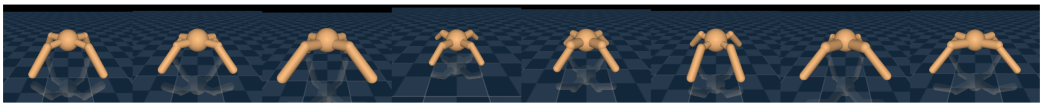


Figure 4: Selection of initial bodies for the body-improving population

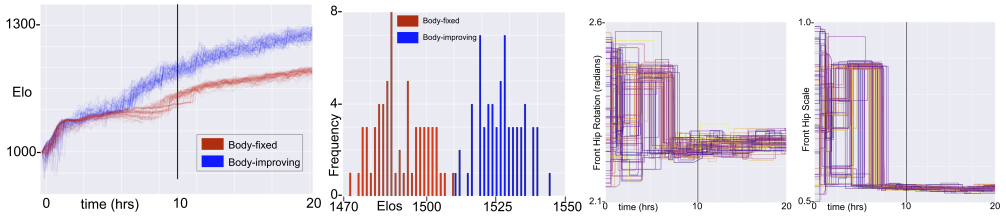
scale the child node, illustrated in Figure 3. Edges have a "reflection" parameter to facilitate body symmetry; when enabled, the body of the child node is created twice: once in its specified position and orientation, and a second time reflected across the parent's Z-Y plane. No control systems are specified by the genotype, instead all the actuators are made available to the RL algoithm as its actions. The genotype constructed for our ant consists of three nodes and three edges connected as shown in Figure 2c. 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 physical structure of the body is determined by 25 parameters in these nodes and edges (detailed in the Appendix).

Population Configuration Our experiments are based on two sub-populations, a *body-fixed* and *body-improving* subpopulations. Each of these consists of $n = 64$ agents. In the body-fixed sub-population all agents have the same body configuration, but have different policy parameters and learner hyper-parameters, whereas the body-improving sub-population has agents with different bodies. Both sub-populations are initialized with random policy parameters and the same hyper-parameters. The body-fixed agents are all initialized to the same standard body-configuration (as shown in Figure 2a, with full details given in Appendix). The body-improving agents are each initialized with a different body by sampling body configurations around the original body configuration as detailed in the Appendix. Figure 4 shows example initial bodies for the body-improving population (more examples appear in Figure 7 in the Appendix).

3.1 Comparing body-fixed Populations with Body-Evolving Population

Our experiment is based on data from $k = 50$ runs of the POEM method of Section 2.1, with two sub-populations (a body-fixed and a body-improving sub-population), each with $n = 64$ agents. POEM matches agents for contests uniformly at random across the entire population, so the body-fixed agents to adapt the controller so as to best match the body-improving agents, making them increasingly stronger opponents. Note that finding a good controller for the body-improving 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 5a shows agent Elo ratings over time, in one run, where agents of the body-improving sub-population outperform the body-fixed agents (body-fixed agents are shown in red, and body-improving agents in blue). Both populations start with similar Elo scores, but even early in training there is a gap in favor of the body-improving agents.

To determine whether POEM results in a significant advantage over optimizing only the controller, we study outcomes in all $k = 50$ runs. We run POEM for a fixed number of training steps, 10 training hours (equivalently, $2e9$ training steps), and analyze agent performance. At the evaluation time, each



(a) Agent Elo scores over time. (b) Elo histogram. (c) Convergence of front hip's rotation (left) and scale (right) across agents over time.

Figure 5: Elos progression and evolution of two body parameters during training. Vertical bars represents 10 hours/2e9 steps of training.

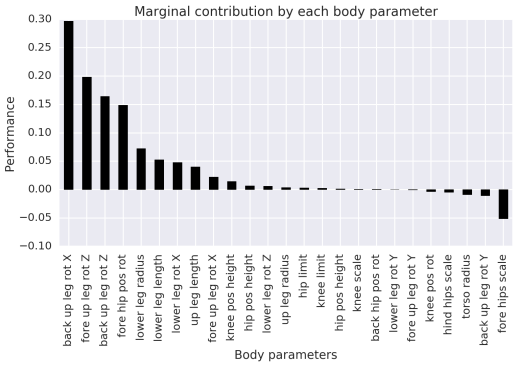
agent (in either sub-population) has its own Elo rating, reflecting its win-rate against others. As our goal is to identify the strongest agents, we examine the highest Elo agent in each sub-population. Figure 5b shows a histogram of Elo scores on the run of Figure 5a, at evaluation time, showing that in this run all body-improving agents outperform all body-fixed agents. We examine the proportion of runs where the highest Elo agent is a body-improving agent (rather than a body-fixed one). In over 70% of the runs, the top body-improving agent outperforms the top body-fixed agent. A binomial test shows this to be significant at the $p < 0.001$ 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 2d shows an example evolved ant from the body-improving population. On average the evolved body is 15% wider and 6% higher (thus 25% heavier), and has a lower center of gravity; the caps on parameters during evolution allow the body to evolve to be much heavier, so the advantage is not only due to mass. Figure 5c shows how some body parameters evolve over time within the population. Initially the variance is high, but by 1.5e9 steps it is negligible. This shows the population converges early in training, to a possibly sub-optimal body.

Using Body-Improving 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-improving controllers fit into the *unevolved* body. Controllers were taken after 10 hours of training. The results show that in 64% of the runs, the controllers taken from the body-improving population outperform those of the body-fixed population, *when used in the unevolved body* (similar to recent observations in EVCs by Jelisavcic et al. (2018)). This shows POEM can find strong controllers even for the original body, and is thus useful even when we cannot modify the physical body of agents.

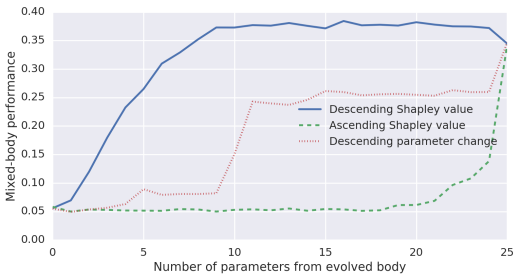
3.2 Identifying Influential Body Changes

Section 3.1 shows we can find a set of changes to the original body, that allow significantly improving its performance (given an appropriate controller). However, which of these changes had the most impact on agent's performance? This question is not completely well-defined, as the performance of agent is not a simple sum of the "strengths" of individual body changes. The different body components depend on each other, and may exhibit synergies between components. For instance, changing the orientation of the leg may only be helpful when changing its length. We thus 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 thus 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. [2]

[2]Our approach is akin to using the Shapley value to measure the importance of features in predictive models (Cohen et al., 2005; Datta et al., 2016). Appendix Section 8 contains a discussion and a motivating example.



(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 6: Shapley analysis of importance of evolved body parameters.

We define the value $v(S)$ of a subset S of body changes as follows. Given the original body b and the evolved body b' and a set S of body parts, we define the body $b(S)$ as the body where each body part $p \in S$ takes the configuration as in the evolved body, and where each part $p \notin S$ takes the configuration as in the unevolved body b. The body $b(S)$ is a hybrid body with some parameters configured as in the original body and some as in the evolved body. To evaluate the performance of a hybrid body we use the evolved controller discussed in Section 3.1. This controller has been learned over many different bodies, and can thus likely handle a hybrid body well. 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, so we can use the formula in Section 1 to compute the Shapley value of each body part. For computational reasons, we settle for an approximation (Bachrach et al., 2010) (see Appendix for details). Figure 6a shows Shapley values measuring relative importance of body changes (for the top body-improving agent from Section 3.1), showing that body changes are unequal in their contribution to agent performance. The high impact body parameters are the orientation of the front and rear upper leg, whereas changing the body radius and scale parameters have a lower impact.

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 twice; once descending through the ranking starting with the highest Shapley-valued parameters, and a second time in an ascending 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 6b 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.

4 Conclusion

We proposed a framework for jointly optimizing agent body and policy, combining continuous control RL agents with an evolutionary procedure for modifying agent bodies. Our analysis shows that this technique can achieve stronger agents than obtained by optimizing the controller alone. We also 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 (Liu et al., 2017)? 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

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

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

D. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of markov decision processes, 2002.

Bongard J. SunSpiral V. Lipson H. Cheney, N. 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.

MacCurdy R. Clune J. Lipson H. Cheney, N. Unshackling evolution: Evolving soft robots with multiple materials and a powerful generative encoding. In *Genetic and Evolutionary Computation Conference (GECCO'13), Amsterdam, The Netherlands.*, 2013.

Shay B Cohen, Eytan Ruppin, and Gideon Dror. Feature selection based on the shapley value. In *IJCAI*, volume 5, pp. 665–670, 2005.

Don Fussell Dan Lessin and Risto Miikkulainen. Adapting morphology to multiple tasks in evolved virtual creatures. In *The Fourteenth International Conference on the Synthesis and Simulation of Living Systems (ALIFE 14)*, 2014.

Risto Miikkulainen Dan Lessin, Don Fussell. Open-ended behavioral complexity for evolved virtual creatures. In *Genetic and Evolutionary Computation Conference (GECCO'13), Amsterdam, The Netherlands.*, 2013.

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

Pradeep Dubey. On the uniqueness of the shapley value. *International Journal of Game Theory*, 4 (3):131–139, 1975.

Arpad E. Elo. *The rating of chessplayers, past and present*. Arco Pub., New York, 1978. ISBN 0668047216 9780668047210. URL `http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216`.

Vincent Feltkamp. Alternative axiomatic characterizations of the shapley and banzhaf values. *International Journal of Game Theory*, 24(2):179–186, 1995.

Faruk Gul. Bargaining foundations of shapley value. *Econometrica*, 57(1):81–95, 1989.

Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pp. 709–715. AAAI Press, 2004.

Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

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

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. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017. URL `http://arxiv.org/abs/1707.02286`.

Jonathan Hiller and Hod Lipson. Automatic design and manufacture of soft robots. 2012.

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

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. Human-level performance in first-person multiplayer games with population-based deep reinforcement learning. *CoRR*, abs/1807.01281, 2018.

Milan Jelisavcic, Rafael Kiesel, Kyrre Glette, Evert Haasdijk, and A. E. Eiben. Analysis of lamarckian evolution in morphologically evolving robots. In *Proceedings of the 14th European Conference on Artificial Life*, pp. 214–221. MIT Press, 2017. ISBN 978-0-262-34633-7.

M.J. Jelisavcic, D.M. Roijers, and A.E. Eiben. Analysing the relative importance of robot brains and bodies. In *ALIFE 2018 Proceedings of the Artificial Life Conference 2018*, Artificial Life Conference Proceedings, pp. 327–334, United States, 2018. MIT Press Journals. doi: 10.1162/isal_a_00063.

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

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

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

Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *In Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163. Morgan Kaufmann, 1994.

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

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

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. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

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

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

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Volodymyr Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing-solving sparse reward tasks from scratch. *arXiv preprint arXiv:1802.10567*, 2018.

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

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

Lloyd S Shapley. A value for n-person games. 1953b.

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. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

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

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

P Straffin. The shapleyshubik and banzhaf power indices as probabilities. *The Shapley value. Essays in honor of Lloyd S. Shapley*, pp. 71–81, 1988.

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

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

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

5 APPENDIX: ANT STRUCTURE AND GENOTYPE

We now describe the full details regarding the structure of the ant, and the genotype containing the parameters governing the shape of the body. Section 3 in the main text describes the configuration of the ant body as a graph-based genotype consisting of three nodes and three edges (shown in Figure 2c), with a root node specifying the spherical torso and two edges specifying the rear and front sets of legs. The parameters for the body-fixed ant nodes and edges are shown in Table 1 and Table 2 respectively. In the body-improving ant all these parameters are mutable with the exception of a node's Shape and the edge's Reflection. These are immutable to keep the number of mutable parameters in the genotype constant and ensure a fixed topology for the ant.

	Shape	Radius (m)	Length (m)	Joint Limit (radians)
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 rotation	y rotation	z rotation	Scale	Parent position	Parent rotation	Reflection
Front hip	$\pi/2$	0	$\pi/2$	1.0	0.5	$3\pi/4$	True
Rear hip	$\pi/2$	0	$\pi/2$	1.0	0.5	$\pi/4$	True
Knee	$\pi/4$	0	0	1.0	1.0	$-\pi/2$	False

Table 2: Edge parameters

6 APPENDIX: FORMAL DEFINITION OF MULTI-AGENT RL AND FULL ENVIRONMENT SPECIFICATION

Our analysis is based on independent multi-agent RL, in a physics simulator environment. At a high level, we use multiple independent learners in a partially-observable Markov game, often called *partially-observable stochastic games* (Hansen et al., 2004). In every state, agents take actions given partial observations of the true world state, and each agent obtains an individual reward. Agents receive a shaping reward to encourage them to approach their opponent, $+100$ if they win and -100 if they lose. 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 $\mathcal{M}$ (Shapley, 1953a; Littman, 1994) 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 $\mathcal{O}^i = \{o^i \mid s \in \mathcal{S}, o^i = O(s, i)\}$ 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 $\pi^i : \mathcal{O}^i \rightarrow \Delta(\mathcal{A}^i)$ (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:

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

6.1 ACTIONS AND OBSERVATIONS IN OUR ENVIRONMENT

Given the above definitions, to fully describe the environment in which our agents interact, we must define the partial observations they receive of the true world state, and the actions they may take. As discussed in Section 2 and Section 3, the physics simulator holds the true world state, but

agents only receive partial observations in the form of an egocentric view. 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, 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. Each limb of the ant body has two hinges, one at the "hip" (attaching it to the spherical 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).

7 APPENDIX: FULL PBT IMPLEMENTATION DETAILS

Our POEM framework uses population based training (Jaderberg et al., 2017; 2018) (PBT) to evolve agents. Algorithm 1 in Section 2 presents a high level view of the procedure, applying 5 sub-procedures: measuring fitness, checking whether an agent is eligible for evolution, selection, inheritance and mutation. As discussed in Section 2, we use Elo ranking (Elo, 1978) for fitness, based on the outcomes of previous contests. The specific Elo computation we use is given in Algorithm 3.

Algorithm 3 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)
    $s \leftarrow (\text{sign}(s_i - s_j) + 1)/2$
    $s_{elo} \leftarrow 1/(1 + 10^{(r_j - r_i)/400})$
    $r_i \leftarrow r_i + K(s - s_{elo})$
    $r_j \leftarrow r_j - K(s - s_{elo})$
end procedure

Periodically *evolutionary events* take place, whereby selected agents are potentially updated with the parameters of the other agents. Each evolutionary event consists of a pairwise comparison between agents within each sub-population. Each pair consist of an eligible *recipient* and an eligible *donor*. To be eligible both agents need to have processed 1×10^8 frames for learning since the beginning of training or the last time they received parameters (whichever is most recent). Further, a recipient agent needs to have also processed 4×10^6 frames for learning since the last time it was involved in an evolutionary event.

POEM's pairwise-matching procedure is simple. For each sub-population, recipient-donor pairs (i, j) are uniformly sampling from the eligible recipients and eligible donors within that sub-population. The Elo 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 t then the recipient agent i will be receive an update based on the donor j. We use a threshold of $t = 45\%$ (i.e. an agent i inherits only if its probability of winning against the target is 45% or less).

When the win-rate of a recipient i against a donor j is lower than the threshold, we change i to be more similar to j by applying two sub-procedures: an inheritance procedure and a mutation procedure. The inheritance procedure is given in Algorithm 2. We have opted to simply copy all of the donor's body configuration parameters to the recipient although other possible variants can be considered, such as taking only some parameters at random, or modifying the parameters to be closer to those of the donor. Following the inheritance procedure, we apply random mutations to the parameters. The mutation procedure is given in Algorithm 4. It 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 of the parameters. The caps avoid the body-improving morphology from diverging too far from the body-fixed morphology, and we impose upper and lower bounds on each mutable parameter at $\pm 10\%$ of the parameter's value in the body-fixed configuration.

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

Figure 7: Selection of initial bodies for the body-improving population

7.1 BODY-IMPROVING GENOTYPE INITIALIZATION

The initial body-improving agent population is generated by sampling body configurations around the body-fixed configuration. Specifically, a random genotype is generated by uniformly sampling a value for each mutable parameter within the mutable parameter's lower and upper bounds. Examples of an initial population of bodies is show in Figure 7.

8 APPENDIX: SHAPLEY VALUE APPROXIMATION FOR BODY PARTS

Section 3.2 defines 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 in theory sim-

ple compute the Shapley value of this game 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.

To motivate the use of the Shapley consider a simple example, where there are three possible changes that can be made to the body: a) increase the length of the leg, b) change the angle of the leg, and c) change the size of the torso. Further suppose that the changes a and b *in isolation* each increase the agent win-rate against a baseline from 50% to 56% (an increase of 6%), while applying c *in isolation* increases the win-rate from 50% to 54% (an increase of 4%). Based solely on this, one might claim that a and b are more impactful changes (as they increase the performance more, in isolation). However, suppose that a and b are *substitutes* so that applying *both* changes a and b only increases the win-rate to 56% (i.e. once one of these changes 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 when combined with either a or b, it improves performance by 5%; for instance, applying both a and c (or both b 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 of components in all permutations, as given in the table below. It would thus be 3% for a and b, and 4.67% for c (i.e. the fair impact of c is higher than of a or b, when taking synergies into account).

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

Section 3.2 analyzes the contribution of individual body changes using the Shapley value. It is based on computing the Shapley value similarly to the computation in Table 3 for the simple hypothetical example. Such a direct computation simply averages the marginal contribution of each component across all permutations (as given in the formula for the Shapley value in Section 1). Although this direct computation is straighforward, it is intractable in for two reasons. First, our definition of the cooperative game in Section 3.2 uses the probability of a certain agent (with a hybrid body and the evolved controller) beating another agent (with the original body and a baseline controller). However, given a set S of body changes, we do not know the win probability of the agent with body $b(S)$ against the baseline agent. Second, the direct formula for the Shapley value in Section 1 is a sum of $r!$ components (each being a difference between $v(A), v(B)$ for some two subsets of body parts A and B) where r is the number of body changes. As we have 25 body components, as opposed to the three components in the hypothetical example, this requires going over $25!$ permutations, which is clearly computationally intractable.

To overcome the above problems we settle for *approximating* the Shapley value in the above game, rather than computing it exactly. We estimate $v(S)$ by generating m episodes where agent $b(S)$ competes against the baseline agent using our simulator, and use the proportion of these where $b(S)$ wins as an estimate of its win-rate (in our experiments we use $m = 1000$ episodes). We then compute the Shapley value using a simple approximation method (Bachrach et al., 2010), which *samples* component permutations, rather than iterating over *all* such permutations.