Evolving an Optimal Morphology for Walking

Nathan Smith
Student number: 20105878

April 19th, 2022

1 Introduction

This paper details the creation of an evolutionary algorithm to evolve an optimal morphology for walking. This project involves three main components in order to work; a physics simulator to simulate the environment, an evolutionary algorithm to evolve the structure, and lastly a reinforcement learning algorithm to train the morphology to walk. We will delve into how each of these were chosen or made as well as the thought process behind any decisions that led to their implementation. The overall goal of the project was to produce morphologies that were adept at walking despite being given only a simple definition of the problem, and no guidance. The program would ideally generate a morphology that was adept at the task despite barely any information about the task being given to it. Only through evolution and evaluation by a reinforcement learning algorithm would it be able to obtain information about the task.

The motivation for producing this paper was to gain a better understanding into the workings of not only evolutionary algorithms as a whole, but to see their power when utilized with reinforcement learning. This combination applied to a simple, but complex task such as walking would hopefully pave the way for an exciting research opportunity to learn about more optimal morphologies for walking. The morphologies generated would be compared to a human-design morphology and could provide insight into fully autonomously designing not only an entire morphology for a task, but also training it. Code for the paper can be viewed here:
(`https://github.com/NathanS-Git/Evolving-Morphology-for-Walking`)

Going into the paper, we will begin by discussing the background and motivation behind why the task of evolving morphology for walking was chosen. After that, several research papers will be discussed, providing a brief outline of each, and detailing their strengths as well as their shortcomings in relation to this paper. Next the methods for each subsection of the algorithm will be discussed, as well as the methodology behind choosing them. The results will come after, showcasing how the overall algorithm performs. Lastly, the discussion and conclusion will detail what insights about the problem were discovered as well as what can be learned from the paper as a whole.

2 Background and motivation

The task of evolving morphology for a fundamental task such as walking mirrors the ethos that real life evolution follows. Many applications of evolutionary computing apply the benefits it provides to specific subtasks disconnected from the reality it was inspired from. The idea of simulation on a general level in order to evolve an agent suitable for a fundamental task from essentially no other given information not only poses an intriguing research question, but if successful, the applications of which could bring insight into newer methods of doing things that traditionally were taken for granted. Where evolution shines is in bringing out solutions never even considered, and applying these techniques to fundamental questions — of which could provide interesting and significant results. Additionally, the autonomous nature of the production of the morphology could introduce new methods disconnected from a human designer that could perform just as well.

The realm of both evolutionary genetics applied specifically to robotics/morphology generation is still quite new. Many generative applications have been done that express intriguing results, however, many of these explore how certain applications apply to pre-defined morphology or limit the changes allowed for the morphology generation. Few allow for dramatic changes in any morphological sense, and even less provide a broad goal for any agents to accomplish. The task of generating a morphology using loose rules to apply to a broad goal can in turn provide insight into

new methods of doing these tasks. With recent advancements not only in computation power but also reinforcement learning algorithms and machine learning as a whole, developing a completely autonomous method for intelligently designing morphology has only just recently become a viable option, and the algorithm discussed within this paper is still a basic implementation of such a task.

3 Literature review

The overarching goal of this project was to build an evolutionary algorithm that when paired with a suitable reinforcement learning algorithm would produce morphology that was best suited to accomplish a specific task, in this case walking. Papers discussed here delve into a similar topic; however, they differ in implementation. The first discusses producing a wide variety of morphology for general tasks, another that optimizes both the form and control of a microrobot through specific optimization techniques, and lastly one that discusses the co-evolution of both structure and controller for biped walking robots.

The first paper, *Task-agnostic morphology evolution* (TAME) is written by Donald Hejna, Pieter Abbeel, and Lerrel Pinto. It is about building an algorithm that focuses on producing morphology-agnostic designs. It is easily the source with the largest overlap with what is being done in this paper, and as such will be evaluated the most in comparison to the other sources. The paper by Donald et al. proposes a new approach to the typical task of co-adapting morphology and behavior through an information-theoretic objective that efficiently ranks agents by their ability to reach diverse states in the environment and the causality of their actions (Hejna, Abbeel, & Pinto, 2021). This method is far more computationally efficient than the one proposed in this document. However, instead of producing morphologies for one defined goal in particular, it is evolved for a wide range of tasks — this was done so intentionally. The paper has many strengths, however, one of the larger limitations of the paper mainly revolve around how they implement morphology generation. Morphologies are represented as graphs, and as such are limited by the certain restrictions this imposes. All morphologies are composed only of capsules; additionally, each joint is specified to be a hinge. This therefore means there are certain morphologies that this paper can construct that TAME can not. The morphologies TAME produces are limited in their variability through the organization of their limbs, rather than the parameters that go into them. For the task of generating morphology adept at general tasks it works well, however for more specialized tasks they may impede any learning algorithms applied to them. In contrast, this paper generates and evaluates morphologies hand-in-hand to produce a specialized morphology for the task of walking with few restrictions on what can be generated.

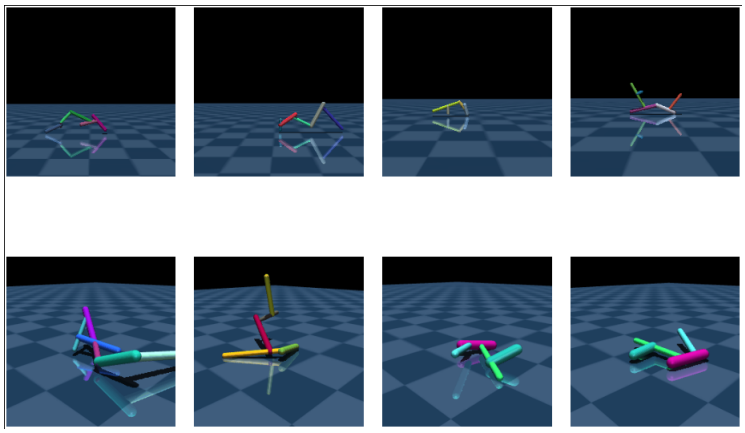


Figure 1: An array of generated morphology from the paper *Task-agnostic morphology evolution*

The second paper, *Data-efficient Learning of Morphology and Controller for a Microrobot* by Thomas Liao, Grant Wang, Brian Yang, Rene Lee, and several others discusses the creation of a microrobot and its controller for the task of walking. Developed through an iterative method by evaluating many variations of the microrobot at a time, further optimizing it with each generation through the use of hierarchical process constrained batch Bayesian optimization (HPC-BBO) (Liao et al., 2019). This method is the core of their paper and showcases the effectiveness of batch optimization applied to time and/or cost constrained robot morphology development. The specific lack of use of evolutionary techniques was due to the cost associated with needing to develop body plans from scratch as opposed to tuning existing designs that evolution requires. In contrast, this paper does not suffer from the cost restriction that Thomas Liao's did. The lack of utilization of evolutionary computing is disregarded, but ideas from the controller optimization are extended through incorporating reinforcement learning as opposed to Bayesian optimization. The double down on utilization of evolutionary techniques for this paper was due to the unconstrained nature of the simulator, and thus strict optimization of each morphology was unnecessary. Additionally, the introduction of a reinforcement algorithm allowed for a more accurate evaluation of each morphology's capabilities.

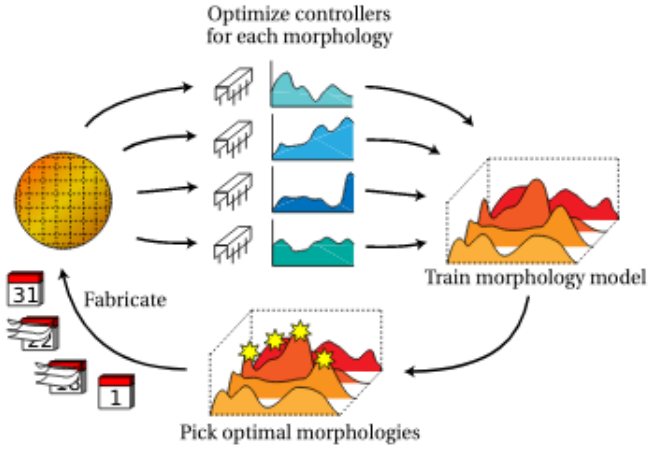


Figure 2: The optimization process from *Data-efficient Learning of Morphology and Controller for a Microrobot*

The final paper, *A Method for Co-Evolving Morphology and the Walking pattern of Biped Humanoid Robot*, by Ken Endo, Fuminori Yamasaki, Tkashi Maeno, and Hiroaki Kitano delves into the topic of robot morphology and controller co-evolution specifically for bi-pedal robots. Similarly to the paper discussed above, it focuses on producing a morphology and controller optimization. Ken Endo's paper accomplishes this through a two-step Genetic Algorithm. Producing morphology and neural oscillators simultaneously, utilizing an evolutionary approach to optimize these parameters. The limitations of Ken's paper however are that the morphologies are heavily limited. Evolution can only modify the length of each link, and the morphologies are restricted to always have four links, contained within two legs. The main insight the paper provides to us is the limitations of the morphology and why they were created to allow us to circumvent these restriction within our own implementation.

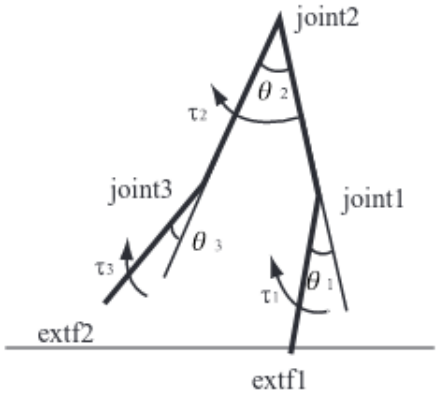


Figure 3: The limited morphology associated with *A Method for Co-Evolving Morphology and the Walking pattern of Biped Humanoid Robot*

Each paper achieves impressive results, centered mainly around simulating a robot that works in a general host of environments. They additionally provide mostly unique methods into producing both morphology and controllers. Their weakness lies in their ability to provide morphologies that are unrestricted in their evolution. Each paper develops morphologies constrained by some factor, either through time/cost constraints, or through limitations in how each morphology is represented internally. The methods utilized and learned from these helped in producing this paper by creating a focus on more unrestricted morphology generation as well as more computationally intensive evaluations of each morphology.

4 Methods

4.1 Introduction

When designing the overall algorithm, it was clear there were several main aspects that needed care in how they were created. Here we'll discuss exactly what those decisions were, and why they were made. To begin, the algorithm consisted of three main aspects — the simulator, the reinforcement learning algorithm, and the evolutionary algorithm.

4.2 Physics simulator

We'll begin with the selection of the physics simulator. The physics simulator that was chosen in the end was MuJoCo (Multi-Joint dynamics with Contact), the recently acquired simulator by Google's DeepMind. Several simulators were examined when deciding which one specifically would be ideal for the specific task we set out to accomplish. Among many others, Bullet, MuJoCo, and ODE were all candidates when selecting the simulator. Tom Erez's paper on *Simulation Tools for Model-Based Robotics* (Erez, Tassa, & Todorov, 2015) was the main paper used to evaluate each simulator's benefits, placing an importance on speed, as opposed to realism. The focus on speed was done to ensure the overall evolution would not take too much time, as the evaluation was already expected to be a large time-sink. With this in mind, MuJoCo became the obvious answer. ODE was considered as a decent second choice, but with MuJoCo's python bindings and its large support in reinforcement learning applications such as within OpenAI's gym, it solidified it as the main candidate for the simulation. The implementation of the simulator within the algorithm was essentially an extension to the already mature OpenAI gym environment. A unique wrapper was built around OpenAI's fully implemented Ant-v3 gym. It was modified to allow for custom morphology, as well as more general hyperparameters, to function better with the diversity of the evolving morphology.

4.3 Reinforcement learning algorithm

Moving on to the reinforcement learning algorithm. Similar to the simulator, several implementations were examined. The paper *Addressing Function Approximation Error in Actor-Critic Methods* (Fujimoto, 2018) by Scott Fujimoto et al. not only compares the performance of many of the state-of-the art reinforcement learning algorithms, but additionally proposes its own, called TD3 (Twin Delayed Deep Deterministic policy gradient algorithm), a direct improvement over DDPG (Deep Deterministic Policy Gradient). Within Scott's paper, TD3 is thrown into the ring with other heavily used algorithms such as DDPG, PPO (Proximal Policy Optimization), and TRPO (Trust Region Policy Optimization). TD3's performance in almost all OpenAI gym tasks was second to none, making it the clear choice for training the morphologies within the evolutionary algorithm. TD3 relies on two Q-networks building on double deep Q-learning, and implements these through the PyTorch library. The implementation for TD3 used within the evolutionary algorithm of this paper is that of the one from Scott's paper.

4.4 Evolutionary algorithm

Finally, arguably the main focus for the paper — the evolutionary algorithm. We'll discuss how morphologies are represented within the program, as well as the overall general inner workings of the algorithm.

When it comes to how morphologies are defined within the code, it was crucial it was not only compatible with the simulator, but also lent itself to being mutated in predictable and logical ways. MuJoCo relies on XML files to load morphologies within it — however, raw XML files do not lend themselves well to storing/manipulating data for the evolutionary aspect of the program. To circumvent this, a special representation of morphology was developed that allowed it to be generated into an XML file through the implementation of a morphology generation method that takes in a morphology representation and produces an XML file. The morphology representation used within this project was a list of lists, each sub-list containing a dictionary. Each sub-list was considered a leg of the morphology, and each dictionary within them — a segment of the leg. The use of dictionaries for the segments allowed for them to have attributes such as length, joint type,

joint rotation plane, diameter, and many more. The first segment in every leg dictates where it is positioned in space, relative to the main body, and all subsequent segments are attached to the starting one. The initial position of the first segment in each leg is initialized to a random point in space on a sphere surrounding the main morphology body. When it comes to generating the XML file, a fixed leg is created going from the main body to the initial position of the leg for graphical continuity. The main body of each morphology is defined at the beginning of the list, in its own separate dictionary, consisting of its body type, and the subsequent parameters that go with it, such as length or radius, as well as parameters that apply to the entire morphology, like name. With the outline of how each morphology is represented within the code, this then allows us to discuss how these function with the overall evolution of the program.

The evolutionary algorithm consists of five methods: population initialization, micro/macro mutation, recombination (breeding), and parent/replacement selection.

4.4.1 Population initialization

Population initialization is one of the largest methods within the program. It generates the starting population of morphology, with random attributes ranging from main body type, to limb count. Throughout the building process, we create a random amount of legs, and per leg, a random amount of segments. These random values have ranges given within the function parameters. After the morphology is completed, its associated XML file is created and both the file name and the morphology itself are combined into a tuple and added to the population. A list containing these tuples is what is returned as the population.

4.4.2 Mutation

Each mutation method modifies slightly different aspects of the morphology. There exists both micro, and macro mutations within the program. Macro mutations modify the degree of freedom of the joints, the joint hinge plane, minor length changes, as well as main body size adjustments. These changes are all miniscule in value, and are meant to nudge a morphology slightly, making no large changes. On the other hand, macro mutations does the opposite, attempting to make as large of changes as possible, these include joint type changes, removing/adding entire segments, removing/adding entire legs, and major length changes. This results in a larger diversity even if two parents continually dominate the gene pool, allowing for their children to have the opportunity to become vastly different from them.

4.4.3 Breeding

Breeding was crucial for the evolutionary algorithm, and was the hardest to implement due to the nature of the program. The implementation creates a list of both parent's legs, shuffles this list, and randomly distributes these legs among the two children. Each child inherits the respective parent's main body type. The shuffling of legs enables the children to have legs in configurations that would not have been generated on startup, as the absolute leg position is taken from one body type and potentially given to the other.

4.4.4 Parent/Survivor selection

Both parent selection and replacement selection were very basic, and function almost identically. They simply return the highest/lowest fit individuals for breeding/replacement. The lowest fit indi-

viduals are flagged for replacement by the parents' children, and the parents (most fit individuals) are guaranteed to move on to the next generation, ensuring the population never devolves.

4.5 Conclusion

Overall, when it came to designing the algorithm for evolving morphology, specific care and attention was made towards ensuring every aspect would function not only well alone, but in conjunction with each other. These restrictions resulted in the simulation engine MuJoCo, and the reinforcement learning algorithm TD3, both being chosen and being combined within Python into the evolutionary algorithm used within this paper. In the next section we'll discuss the results of our implementation.

5 Results

5.1 Introduction

To begin, the algorithm functions by first generating a population of 10 individual morphologies. By default, each morphology is generated for 50k iterations multiplied by the generation. This decision was made to evaluate morphologies for longer as the program progresses — with the idea being that the program would spend less time on the less mature morphology, while allowing for more advanced morphology to form as time progresses. After each morphology is evaluated, the two best are bred, and their children replace the six worst, producing the next generation.

5.2 First simulation (without macro mutations)

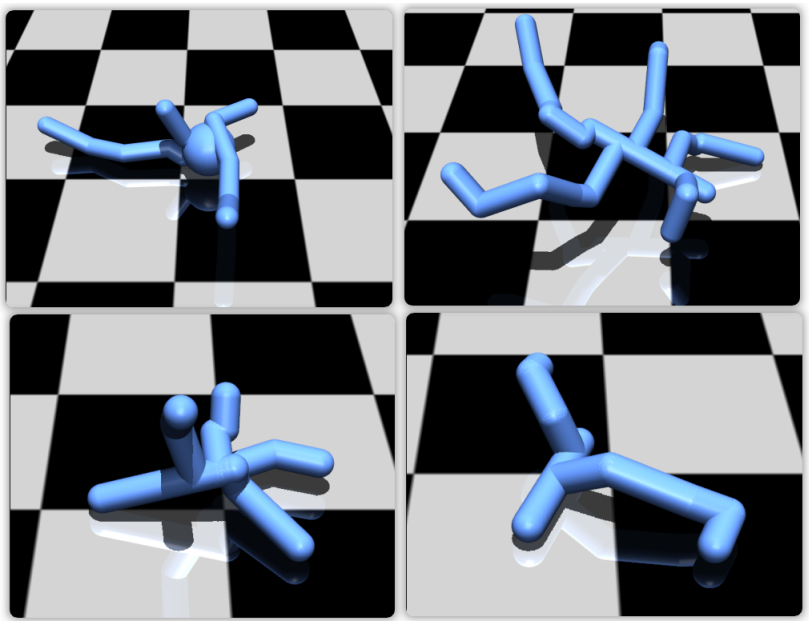


Figure 4: Examples of generated morphology

Figure 4 showcases the capabilities of the population initializer — producing a wide array of unique configurations. Many of the morphology produced are varied, and this is crucial because it shows

that the space of all possible morphologies is large, and subsequently ripe for discovering complex structures that are adept at a unique and specific task.

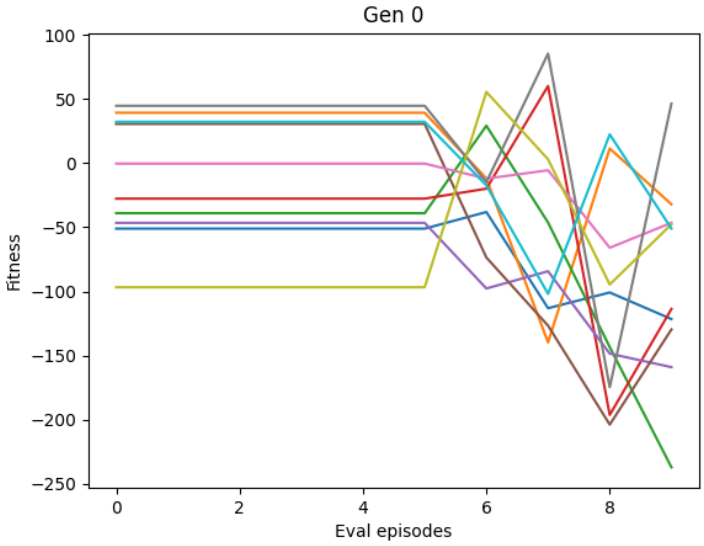
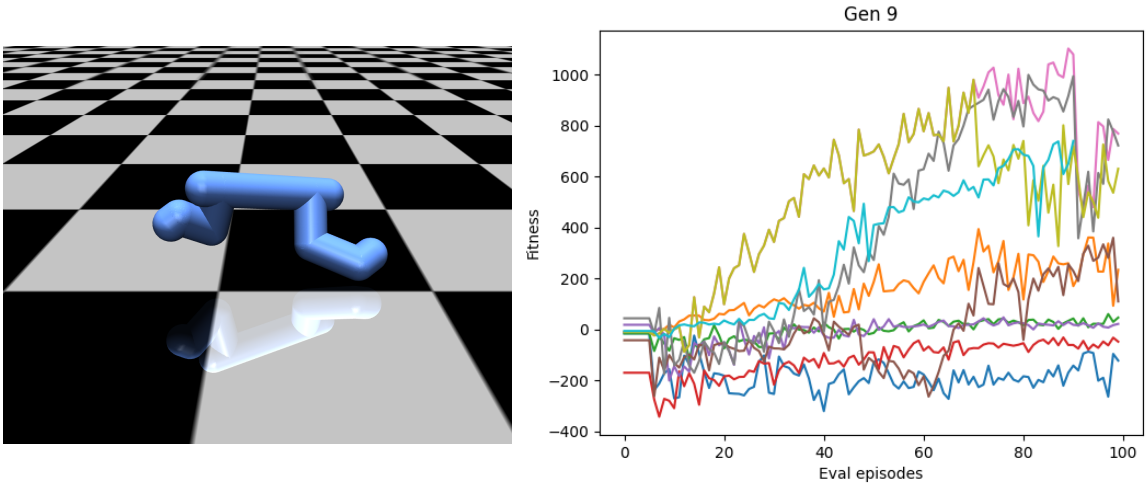


Figure 5: The fitness for each morphology within the initial generation

Figure 5 shows the initial generation results from one of the older runs of the algorithm. The large flat line at the beginning is when the reinforcement learning algorithm is in its exploration phase, making the first 5 evaluations on an untrained policy. The algorithm evaluates every 5k iterations, so for the first 50k iterations, there are 5 evaluations dedicated to exploration, and the other 5 to begin learning. The results here are to be expected, as the first generation will see mainly random noise. The slight tendency towards negative values is also expected as minor negative rewards are added in addition to the main reward for walking, such as large contact forces and control cost weight. These negative rewards are miniscule, but encourage the morphology to not rely so much on jumping.



(a) Best morphology

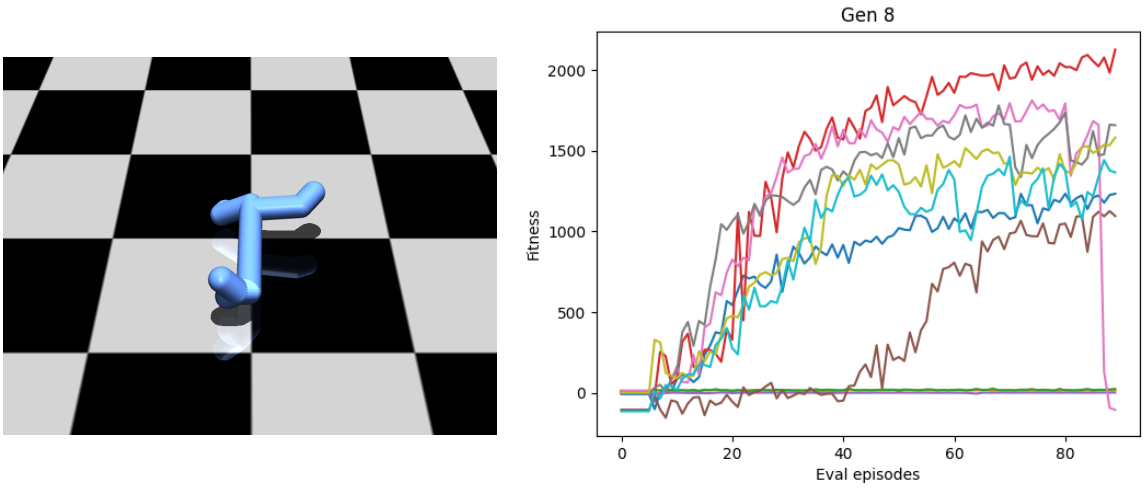
(b) Fitness for entire generation

Figure 6: The best morphology from this simulation after 9 generations as well as the fitness for each morphology within the generation. *It should be noted that the colors given in each graph are meant for ease of viewing, and are not consistent between graphs and/or representative of specific morphology.*

These results within Figure 6 are from the same population as Figure 5, however this is nine generations into the future. It is clear to see that the evolutionary algorithm accomplishes growth, and the results showcase the overall programs ability to evolve optimal morphologies for walking. The maximum fitness the most optimal morphology was able to reach within these nine generations was around 1100 after 100 iterations and as we shall soon see, is decent. Comparing the highest fitness levels from generation 0 to generation 9, growth is clear to see, as the program develops morphologies that are able to compete with one another and produce a competitive environment. The parameters for the above simulation only included micromutations, in order to see the contrast that macro mutations would provide.

Figure 6a is the best morphology for the data within Figure 6b. It was produced in generation 8 and was the 2nd child. It functions during simulation by constantly rotating its back appendage to fling itself forward by jumping, and then reorients itself when it lands using its front appendage.

5.3 Second simulation



(a) Best morphology

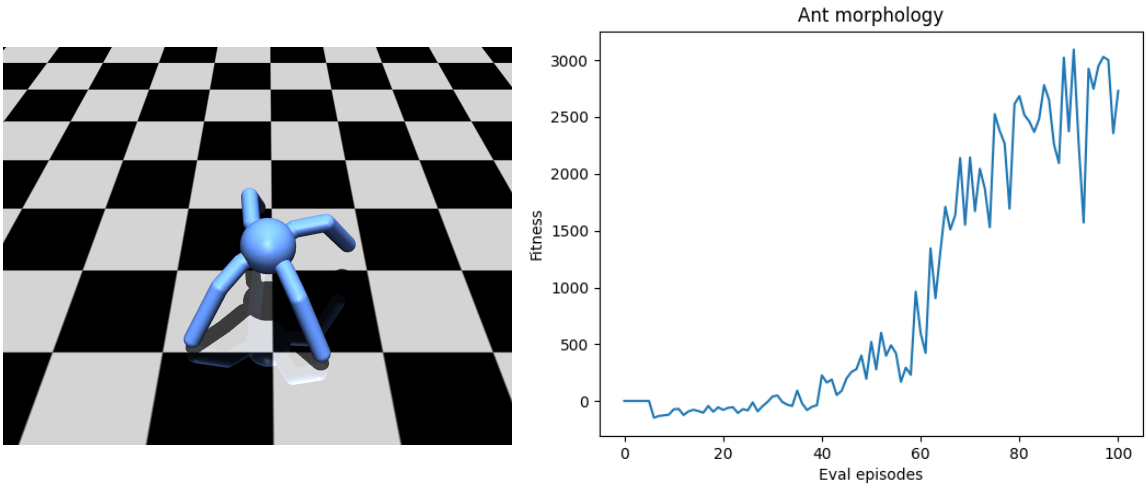
(b) Fitness for entire generation

Figure 7: The best morphology from this simulation after 9 generations as well as the fitness for each morphology within the generation.

Figure 7 was a later run of the algorithm with similar parameters as Figure 6 with macro mutations included. It is clear to see that this performed significantly better, as the larger diversity per generation allowed the program to reach a maximum fitness of about 2100 after 90 evaluations. This showcases not only the potential the algorithm has, but additionally, just how sensitive the algorithm can be to starting parameters, as well as the importance of both large and small mutation changes during the program's execution.

The above morphology within Figure 7a is the best morphology from the simulation data in Figure 7b. It was produced in generation 8 and was the 3rd child. It may not be clear to see from the image alone, but actual simulation of the morphology uncovers why it reigned supreme among the others; it had evolved what was essentially a wheel as its back leg, and used the two prongs at its front to balance itself as it propelled forward. In contrast to the previous best morphology, it relies more so on an intelligently designed motor rather than rapid jumping — which would explain its vastly superior fitness value.

5.4 Baseline



(a) Pre-defined morphology

(b) Fitness results

Figure 8: The human designed "Ant" morphology as well as its respective fitness values.

All the values from Figures 6 and 7 are useless unless we have some baseline to compare them to. When comparing the results to a pre-defined morphology such as the ant morphology given above in Figure 8, we can gain a greater insight into how well the overall algorithm performs. As can be seen, the predefined ant morphology obtains a max fitness of about 3000 after 100 iterations. This makes the best morphology from the first population one third the performance of a well-defined morphology, and the latest population a solid two thirds in terms of performance. Additionally, these scores were achieved in only 8-9 generations; it remains to be seen that running the algorithm for longer could potentially achieve results even better than the predefined morphology. These results showcase the algorithm's functions, and can provide unique insight into the practicality of evolving morphologies to complete specific tasks.

5.5 Conclusion

Overall the results suggest the evolutionary algorithm falls only slightly behind a human designed morphology, and could potentially produce superior results, if given enough time. The breadth in the configurations the morphology generation can produce showcase its unrestricted ability — this in conjunction with the TD3 algorithm was able to produce morphologies adept at the task of walking. It was able to create morphologies capable of consistently reaching over 1000 fitness in 100 iterations without macro mutations after only 9 generations. Introducing macro mutations produces significantly better morphologies capable of reaching 2000 fitness within the same time-frame. These are only just behind a human define morphology at 3000 fitness.

6 Discussion

The applications of this paper are mainly conceptual in nature. It produced an answer to the question of what an entity could look like if it was optimized for walking and produced largely

unrestricted morphology through evolution, utilizing reinforcement learning for evaluation. However, the paper is not without its limitations. The morphologies generated throughout the process developed inhuman methods of walking. In order to obtain more realistic walking patterns some small environmental parameters could be tweaked; potentially involving reward shaping for the reinforcement algorithm. Modifying the evolutionary algorithm to restrict such configurations could work as well, however the algorithm as a whole may be stunted by this change — and so this would not be recommended. Additionally, this task was applied strictly to the 3D case; it would be interesting to see the results applied to simpler environments, perhaps 2D — this was never tried within this paper as it went against the desired complexity of the environment. Lastly, the issues with the current implementation of the algorithm is that is suffers heavily from performance — taking several days to compute less than 10 generations – this could potentially be remedied through parallelization of the algorithm.

The main contribution this paper provides is an insight into how morphologies may be formed through evolutionary means, providing a greater insight into the evolutionary process. It additionally showcases that more automatic methods of creating morphology configurations are possible; not relying on human creators. The main takeaway is the creation of the algorithm discussed within this paper. Producing morphology capable of specific physically complex tasks defined by simple rules. The applications of this lend itself well to walking, however other implementations could be tried, such as jumping, turning, etc.

The methods discussed within the paper could be utilized by future researchers to develop general morphologies for a wide range of applications that are optimal for a given task. These tasks are restricted to being modeled through reinforcement learning rewards and lend themselves to more general goal definitions. The morphology generation could provide new insights into what may not have been considered — potentially evolving morphologies that are superior to hand designed configurations. Another technique that could be tried is further optimizing an already hand-made morphology by introducing it into the starting population, and having it compete against slightly mutated versions of itself. This could further improve semi-optimal designs through evolutionary means. The foundations for the overall program are basic on a fundamental level, but could provide a powerful tool — marrying a feature-rich morphology evolutionary algorithm with a state-of-the art reinforcement learning algorithm. All in all, any one wishing to build upon the technology discussed within this paper has many avenues to follow with a large selection of methods to extend upon.

7 Conclusion

Overall, a method to produce morphology designed for the task of walking was developed through the implementation of the TD3 algorithm combined with an evolutionary algorithm within the MuJoCo physics simulator. These united to produce an environment from which morphologies were generated, evaluated and subsequently evolved to become adept at the task of walking.

This paper set out to develop a morphology generator that could produce robot-inspired configurations that were adept at a defined task without relying on any human intervention. It succeeded in doing this, and provided a foundation that other researchers could further improve — producing new methods for doing tasks that minimized the amount of design decisions made through humans alone. The paper was far from perfect, producing results that relied too heavily on unrealistic methods of movement as well as suffering from a heavy computational debt. However, despite these limitations a fundamental implementation for combining genetic evolution with reinforcement learning was developed, and subsequently, brought to light a host of knowledge on not only

what these are capable of now, but their potential for the future.

References

DeepMind. (2017). *Emergence of locomotion behaviors in rich environments.* Retrieved from https://arxiv.org/pdf/1707.02286.pdf

Endo, K., Yamasaki, F., Maeno, T., & Kitano, H. (2009). *A method for co-evolving morphology and walking pattern of biped humanoid robot.* Retrieved from http://lab.sdm.keio.ac.jp/maenolab/previoushp/paper/ICRA2002_Endo.pdf

Erez, T., Tassa, Y., & Todorov, E. (2015). *Simulation tools for model-based robotics: Comparison of ...* Retrieved from https://homes.cs.washington.edu/~todorov/papers/ErezICRA15.pdf

Fujimoto, S. (2018). *Addressing function approximation error in actor-critic methods.* Retrieved from https://proceedings.mlr.press/v80/fujimoto18a/fujimoto18a.pdf

Hejna, D., Abbeel, P., & Pinto, L. (2021). *Task-agnostic morphology evolution.* Retrieved from https://arxiv.org/abs/2102.13100

Liao, T., Wang, G., Yang, B., Lee, R., Pister, K., & Levine, S. (2019, May). *Data-efficient learning of morphology and controller for a microrobot.* Retrieved from https://arxiv.org/abs/1905.01334

OpenAI. (2017). *Proximal policy optimization.* Retrieved from https://openai.com/blog/openai-baselines-ppo/

OpenAI. (2018). *Background for twin delayed ddpg.* Retrieved from https://spinningup.openai.com/en/latest/algorithms/td3.html#background

Risi, S., & Stanley, K. (2013). *Confronting the challenge of learning a flexible neural controller for a diversity of morphologies.* Retrieved from https://eplex.cs.ucf.edu/papers/risi_gecco13b.pdf