

# Evolving Robotic Leg Shapes via Deep Reinforcement Learning

*Hayden Sheung  
Ronald S. Fearing, Ed.*



Electrical Engineering and Computer Sciences  
University of California at Berkeley

Technical Report No. UCB/EECS-2020-103

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-103.html>

May 29, 2020

Copyright © 2020, by the author(s).  
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

### Acknowledgement

I would like to thank my advisor, Professor Ronald Fearing, for all of his guidance and support over the past two years. I would also like to thank every member of the Biomimetic Millisystems Lab, especially Dr. Liyu Wang for his mentorship. I am fortunate to work in this laboratory. I am also grateful for Professor Kristofer Pister for agreeing to be the second reader of this thesis.

I would also like to express my gratitude to Alma Chen, who has been an invaluable and supportive friend in the last two years. Finally, I want to thank my parents and sister for their unconditional and unlimited support throughout all the years.

Evolving Robotic Leg Shapes via Deep Reinforcement Learning

by

Hayden Yui Sheung

A thesis submitted in partial satisfaction of the

requirements for the degree of

Masters of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ronald S. Fearing, Advisor  
Professor Kristofer S. J. Pister, Second Reader

Spring 2020

---

# **Evolving Robotic Leg Shapes via Deep Reinforcement Learning**

by Hayden Yui Sheung

---

## **Research Project**

Submitted to the Department of Electrical Engineering and Computer Sciences,  
University of California at Berkeley, in partial satisfaction of the requirements for the  
degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

### **Committee:**



---

Professor Ronald S. Fearing  
Research Advisor

May 28, 2020

---

(Date)

\* \* \* \* \*



---

Professor Kristofer S. J. Pister  
Second Reader

5/28/2020

---

(Date)

## Abstract

Evolving Robotic Leg Shapes via Deep Reinforcement Learning

by

Hayden Yui Sheung

Masters of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ronald S. Fearing, Advisor

In the current state of the art for deep reinforcement learning in robotics, the primary focus is on maximizing an objective function by controlling an unchanged agent. On the other hand, there is a subfield called evolutionary robotics. As implied by the name, it involves using evolutionary algorithms to develop better hardware for robots' performances. However, the field is a little outdated. The core of the aged evolutionary algorithm is that given a set number of shapes, a subset of them (called "parents") is chosen, based on their performances in the environment. They are the foundations for the next generation's shapes, and the program mutates the "parents" set to acquire the next set of shapes to examine. The experiment is then run for a fixed number of generations, and the best shapes are picked from the last generation.

An exciting crossover between the two fields is using the new advanced deep reinforcement learning algorithms in evolutionary robotics. Namely, instead of merely picking the best  $N$  shapes in each generation based on their performances in the environment, deep reinforcement learning methods are used to identify the most optimal shape. Then, some mutations are applied to this shape, and it serves as the starting position for the next generation.

The design of the agent is parameterized and allows the agent to learn its body parts during the simulation. The primary objective of the robot is to crawl on the desired path on a chosen terrain. REINFORCE and actor-critic are the deep reinforcement learning optimization algorithms applied. The results indicate that, with the new proposed approach, agents can learn body parts to facilitate their movements in the given environments.

Dedicated to the memories of my grandparents, who always believed in me.

# Contents

<b>Contents</b>	<b>ii</b>
<b>List of Figures</b>	<b>iv</b>
<b>List of Tables</b>	<b>vi</b>
<b>1 Background</b>	<b>1</b>
<b>2 Related Work</b>	<b>3</b>
2.1 OpenRoACH . . . . .	3
2.2 Previous work . . . . .	3
<b>3 Methods</b>	<b>5</b>
3.1 Data source . . . . .	5
3.2 Metrics . . . . .	5
3.3 Shapes parameterization . . . . .	7
3.4 Proposed evolutionary algorithm . . . . .	9
3.5 REINFORCE . . . . .	10
3.6 Actor-critic . . . . .	10
<b>4 Experiments and Setup</b>	<b>12</b>
<b>5 Results</b>	<b>15</b>
5.1 Flat surface, straight path . . . . .	15
5.2 Hilly surface, straight path . . . . .	20
5.3 Flat surface, curved path . . . . .	24
5.4 Hilly surface, curved path . . . . .	28
<b>6 Analysis</b>	<b>32</b>
<b>7 Conclusion</b>	<b>35</b>
<b>8 Future Work</b>	<b>36</b>



**Bibliography**

# List of Figures

3.1	The robot with the coordinate system. The red arrow represents the x-axis, the green arrow represents the y-axis, and the blue arrow represents the z-axis. This axis system is in the robot's body frame. . . . .	6
3.2	The robot with the standard leg shape, created by Andrew Pullin. . . . .	7
3.3	The left table displays how the legs look with no joint. The right table contains examples of the legs with one joint each. The variables are the sizes of the tips of the leg, the spring constants of the joints that are connecting to the tips, the numbers of joints per leg, and the shapes of the tips. . . . .	8
3.4	An example of the robot used in the experiment with a 0-joint design for its legs.	8
4.1	The flat and the hilly surfaces for the robot to crawl on. . . . .	13
4.2	The straight paths followed by the robot in the flat and hilly scenes. . . . .	13
4.3	The curved paths followed by the robot in the flat and hilly scenes. . . . .	14
4.4	The red lines are the ideal paths on the hilly surfaces, plotted in the contour maps for both scenes. . . . .	14
5.1	Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a straight path over 50 generations. . . . .	15
5.3	The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a flat surface and a straight path. . . . .	17
5.4	Metrics 3.1 (left) and 3.2 (right) for the last generation, on a flat surface and a straight path. . . . .	18
5.5	Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a straight path over 50 generations. Initialized with five randomly generated configurations for each optimization method.	18
5.6	Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a straight path over 50 generations. For these experiments, Equation 3.3 replaces Equation 3.1 in Algorithm 2. . . . .	19

5.7	Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a straight path over 50 generations. . . . .	20
5.9	The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a hilly surface and a straight path. . . . .	22
5.10	Metrics 3.1 (left) and 3.2 (right) for the last generation, on a hilly surface and a straight path. . . . .	22
5.11	Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a straight path over 50 generations. Initialized with five randomly generated configurations for each optimization method. . . . .	23
5.12	Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a curved path over 50 generations. . . . .	24
5.14	The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a flat surface and a curved path. . . . .	26
5.15	Metrics 3.1 (left) and 3.2 (right) for the last generation, on a flat surface and a curved path. . . . .	26
5.16	Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a curved path over 50 generations. Initialized with five randomly generated configurations for each optimization method. . . . .	27
5.17	Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a curved path over 50 generations. . . . .	28
5.19	The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a hilly surface and a curved path. . . . .	30
5.20	Metrics 3.1 (left) and 3.2 (right) for the last generation, on a hilly surface and a curved path. . . . .	30
5.21	Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a curved path over 50 generations. Initialized with five randomly generated configurations for each optimization method. . . . .	31

# List of Tables

5.1	The final parameters after 50 generations on a flat surface with a straight path.	16
5.2	The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a straight path on a flat surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs. . . . .	16
5.3	The ranges of the final parameters for each optimization method with the robot crawling on a flat surface and a straight path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.	19
5.4	The final parameters after 50 generations on a hilly surface with a straight path.	20
5.5	The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a straight path on a hilly surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs. . . . .	21
5.6	The ranges of the final parameters for each optimization method with the robot crawling on a hilly surface and a straight path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.	23
5.7	The final parameters after 50 generations on a flat surface with a curved path. .	24
5.8	The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a curved path on a flat surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs. . . . .	25

5.9	The ranges of the final parameters for each optimization method with the robot crawling on a flat surface and a curved path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.	27
5.10	The final parameters after 50 generations on a hilly surface with a curved path.	28
5.11	The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a curved path on a hilly surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs. . . . .	29
5.12	The ranges of the final parameters for each optimization method with the robot crawling on a hilly surface and a curved path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.	31

## Acknowledgments

First and foremost, I would like to thank my advisor, Professor Ronald Fearing, for his guidance and support over the past two years. I would also like to thank every member of the Biomimetic Millisystems Lab, especially Justin Yim for his help with lab logistics, Anusha Nagabandi for her assistance with machine learning-related questions, and Dr. Liyu Wang for his mentorship. I am fortunate to work in this laboratory. I am also grateful for Professor Kristofer Pister for agreeing to be the second reader of this thesis.

I am incredibly appreciative of everyone I have met in Berkeley. Each of them has helped me through this journey and becoming the person I am today. In particular, I would like to express my gratitude to Simon Lau, who has ignited my interest in research, and to Alma Chen, who has been an invaluable and supportive friend in the last two years. Finally, I want to thank my parents for their unconditional and unlimited support, including immigrating to the United States, driving me to Berkeley for countless times, and paying for my tuition. I also want to thank my sister for editing my essays, including this thesis, and for being my confidant during my time at Berkeley.

# Chapter 1

## Background

Designing robotic parts takes a lot of staff-hours and expertise knowledge. Besides, hand-crafted designs may not be the most optimal, as the search space can be too broad for humans to pick out the best shapes for a specific task.

Evolutionary robotics offers a convenient alternative to designing the parts by hand [Nolfi et al., 2000]. It involves having the algorithm to search for the best shape iteratively. Algorithm 1 summarizes the high-level steps of the iterative approach for finding the best shapes.

---

**Algorithm 1:** Summary of the iterative algorithm to pick the best robotics design in evolutionary robotics

---

candidates = Randomly initialize shapes

**for**  $G$  generations **do**

    Perform the tasks for the shapes in the candidates shape set

    parents = The best  $N$  shapes based on the chosen metric out from the candidates

    Mutate the parents set by slightly changing the parameters that determine the shapes

    candidates = parents

**end**

Output the candidates shapes

---

However, with the advances in machine learning in the last twenty years, the iterative ap-

proach may be outdated. Specifically, the way to choose the best shapes can be expensive, as each shape requires an iteration to determine the reward, and there can be many shapes to test for in each iteration.

The fact that organisms can adapt to their specific goals and environments, such as athletes performing weightlifting for enhancing their performances in sports [Tricoli et al., 2005], inspires the possibility for the agent to learn and adapt its body parts. The idea of artificial evolution on body parts was proposed as earliest as 1994 [Sims, 1994a,b].

This report suggests a new evolutionary algorithm, in which other machine learning algorithms replace the iterative genetic method. The new algorithm is still evolutionary in the sense that there is a fixed number of generations. Due to the time constraints, this report focuses on the evolution of the body parts only, instead of a co-evolutionary approach that improves the designs and the control simultaneously. The initial plan was to perform a majority of the learning in simulations powered by a physics engine named V-REP, and the results would be validated through actual robotic experiments. Due to the complications of COVID-19 and the closure of physical laboratories, all investigations are accomplished via simulations instead.



# Chapter 2

## Related Work

### 2.1 OpenRoACH

The simulated robot used for this research is derived from the OpenRoACH [Wang et al., 2019]. It was developed by Liyu Wang, a former postgraduate researcher in my group. OpenRoACH is a cheap and durable robot with the Robot Operating System (ROS) onboard. One of the ultimate goals of OpenRoACH is to become multiuse. For example, in the case of earthquakes, OpenRoACHs can be used to deliver the necessary resources, even on uneven surfaces. To do so, OpenRoACHs need to have the ability to adapt their body parts and control to move on various surfaces by switching their robotic parts at the repairing stations. This objective is the origin of this project.

### 2.2 Previous work

In 1994, Karl Sims proposed the idea of artificial evolutions that are similar to the ones organisms go through in nature [Sims, 1994a,b]. There is also some research done on automatically revolutionizing soft robots for the aquatic and terrestrial environments via a physics simulation engine [Corucci et al., 2018]. A more recent publication attempts to improve legs with Bezier splines, which gives the algorithm an ability to produce complex

shapes with a high degree of freedom [Collins et al., 2018].

With the recent advances in robotics and machine learning, there are some studies on optimizing body designs and control simultaneously. There is a paper that examines the possibility of learning locomotion with deep reinforcement learning for sim-to-real robots [Tan et al., 2018]. Similarly, in [Ha, 2019], David Ha utilizes REINFORCE as the optimization algorithm to modify the agent’s parts to accomplish missions such as walking on terrains with holes. There is a study, perhaps the closest to the full version of this project, that attempts to optimize control and designs alternately with reinforcement learning [Schaff et al., 2019]. In these analyses, they are all leveraging the regular reinforcement learning algorithms. In contrast, this study tries to combine genetic algorithms and deep reinforcement learning algorithms.

# Chapter 3

## Methods

This chapter discusses the various metrics and methods used in this project.

### 3.1 Data source

All of the data in this report are from simulations rendered in V-REP.

### 3.2 Metrics

Two objective functions are used in this project – the deviated distance (Equation 3.1) and the cost of energy (Equation 3.2). These two metrics capture the main goal OpenRoACH wants to accomplish in this project – to move along a specific path with an efficient body part design. In the training phase, a weighted sum of them serves as the objective function.

$$\sum_{t=1}^T [|(y_t - y_{t-1})| + |(z_t - z_{t-1})|] \quad (3.1)$$

3.1: Deviated distance along the y- and z- axes

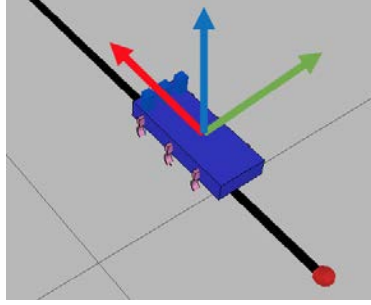


Figure 3.1: The robot with the coordinate system. The red arrow represents the x-axis, the green arrow represents the y-axis, and the blue arrow represents the z-axis. This axis system is in the robot's body frame.

$$\frac{\sum_{t=1}^T \sum_{i=1}^6 |F_{i,t}| \cdot |(\theta_{i,t} - \theta_{i,t-1})|}{mgd} \quad (3.2)$$

### 3.2: Cost of transport

The deviated distance (Equation 3.1) is a cumulative sum of the distance of the robot's center of mass from the x-axis (where the desired path lies on) during the whole simulation,  $T$  timesteps. The vertical (z-axis) and horizontal (y-axis) deviations are included in the metric as the robot should make an effort to stay on the floor and the path while crawling. The lower this deviated distance is, the better.

The cost of transport (Equation 3.2) is a ratio between the energy required to move the robot and the product of the distance traveled ( $d$ ), the mass of the robot ( $m$ ), and the gravitational constant ( $g$ ). Similar to the deviated distance, the lower the cost is, the better. The energy produced is calculated in discrete timesteps. It is the product between the force at a joint and the displacement of the joint in that timestep. Finally, the result is from summing these products across all six legs. The energy is the cumulative sum across the whole simulation.

$$\frac{\sum_{t=1}^T [|(y_t - y_{t-1})| + |(z_t - z_{t-1})|]}{d} \quad (3.3)$$

### 3.3: Normalized deviated distance along the y- and z- axes

The normalized deviated distance (Equation 3.3) is another proposed metric. It is utilized for validating if the deviated distance (Equation 3.1) is a credible metric for this study, since Equation 3.1 is not normalized and may be unbounded with a growing path length,  $d$ .

### 3.3 Shapes parameterization

For the algorithms to learn the designs of the body parts during the simulation, this project represents the shapes of the tips of the legs using a learnable parameter vector. This vector determines the primitive shape (cubes, cylinders, spheres), the height, the width (in the case of a sphere, the height is the same as the width), the mass, the spring constant of the joint that connects the tip with the leg, and the number of joints each leg has. The number of joints, which is best explained as the number of “knees” in each leg, is an intriguing factor to examine because there are variations among different organisms. For example, humans have one joint in each limb, but some insects have up to three joints per leg [Cruse et al., 2009]. Finally, the volume of each leg is constrained to be less than 25% of the main body, and the mass of each leg is limited to be less than 20% of the main body’s mass. These restrictions exclude impractical designs from the search space. Then, the performances of the shapes are compared to the standard shape developed by a former lab member, Andrew Pullin. For simplicity, only the front two legs are modified in this study.

To illustrate, the following are the standard shape and examples of learned leg shapes.

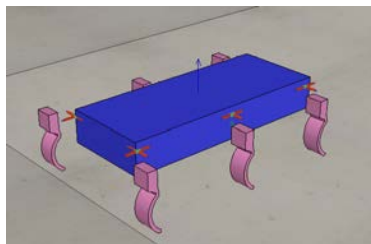


Figure 3.2: The robot with the standard leg shape, created by Andrew Pullin.

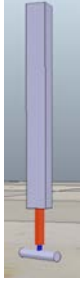

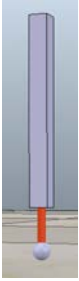

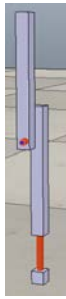
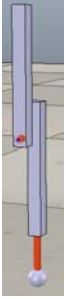
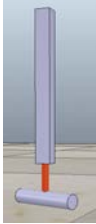
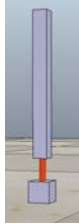
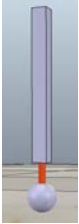



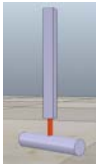
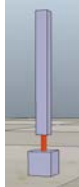
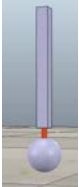


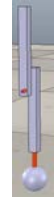
	Cylinder	Cube	Sphere		Cylinder	Cube	Sphere
<b>Small</b>				<b>Small</b>			
<b>Medium</b>				<b>Medium</b>			
<b>Large</b>				<b>Large</b>			

Figure 3.3: The left table displays how the legs look with no joint. The right table contains examples of the legs with one joint each. The variables are the sizes of the tips of the leg, the spring constants of the joints that are connecting to the tips, the numbers of joints per leg, and the shapes of the tips.

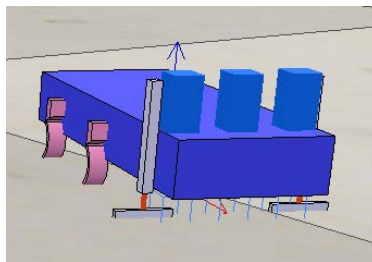


Figure 3.4: An example of the robot used in the experiment with a 0-joint design for its legs.

### 3.4 Proposed evolutionary algorithm

There are many evolutionary algorithms [Arias-Montano et al., 2012]. To summarize, there are three main steps in an evolutionary algorithm: creating a new generation, mutating the current generation, and selecting the best candidate.

The proposed algorithm progresses as follows. First, the starting design of the legs is initialized with a random vector. Then, the design is evaluated in the given environment, and a weighted sum of the two metrics is computed. The weights are chosen such that both quantities have roughly the same influences in the objective function. For this study, (0.025, 1) are chosen to be the multipliers for the cost of transport and the deviated distance, respectively. A deep reinforcement learning algorithm is then applied to the initial vector to reach an optimal value. All the steps described count as one generation. For the next generation, a mutation (small perturbations to the weights in the shape vector) is applied to the current design. There are two goals for the mutation. First, the algorithm hopefully leaves the local optimum in the search space for the shape vector. Second, the mutation initiates the next generation of the evolutionary algorithm. Algorithm 2 summarizes the proposed algorithm.

---

**Algorithm 2:** The proposed evolutionary algorithm

---

candidate = Randomly initialize a shape parameter

**for**  $G$  generations **do**

    Optimal shape = Start at the candidate design, perform a deep reinforcement learning algorithm to find the optimal shape vector, with the reward function =  $0.025 \cdot \text{Equation (3.2)} + \text{Equation (3.1)}$

    Mutate the optimal shape by slightly perturbing the shape vector

    candidate = optimal shape

**end**

Output the candidate shape

---

### 3.5 REINFORCE

REINFORCE is one of the two deep reinforcement learning algorithms applied to find the optimal shape vector. The following is a rundown of the algorithm [Williams, 1992; Ha, 2019].

$w$  is the learnable vector representing the body designs.  $R$  is the expected cumulative reward.  $\pi$  is the policy.  $\theta$  is the variable the algorithm optimizing over.

$$J(\theta) = E_{\theta}[R(w)] = \int R(w)\pi(w, \theta)dw \quad (3.4)$$

Computing the gradient of  $J(\theta)$  with the log-likelihood trick,

$$\nabla_{\theta}J(\theta) = E_{\theta}[R(w)\nabla_{\theta}\log\pi(w, \theta)] \quad (3.5)$$

With an  $N$ -size population, the gradient is approximately,

$$\nabla_{\theta}J(\theta) \approx \frac{1}{N} \sum_{i=1}^N R(w^i)\nabla_{\theta}\log\pi(w^i, \theta) \quad (3.6)$$

Finally,  $\theta$  is optimized through gradient descent.  $\alpha$  is the step size.

$$\theta \rightarrow \theta + \alpha\nabla_{\theta}J(\theta) \quad (3.7)$$

The above steps are repeated until the algorithm finds the optimal  $\theta$ .

### 3.6 Actor-critic

There are two primary problems with REINFORCE: noisy gradients and high variance. In the actor-critic methods, these two issues are mitigated with two models to compete against each other [Konda and Tsitsiklis, 2000].



The actor function decides which action to take, and the critic function informs the actor how advantageous its action is and how it should adjust.

In this study, the baseline is used as the advantage function. The gradient now looks like this, with  $s_t$  as the state at time  $t$ , and  $\gamma$  as the discount factor:

$$\nabla_{\theta} J(\theta) = \sum_{i=0}^{\infty} \left( \sum_{i'=t}^{\infty} \gamma^{i'} R(w^{i'}) - b(s_t) \right) \nabla_{\theta} \log \pi(w^i, \theta) \quad (3.8)$$

A popular choice (and the one used for this project) for the baseline function is the value function  $b(s_t) = V(s_t)$ . In Equation 3.8, the policy  $\pi$  is the actor and  $V(s_t)$  is the critic.

# Chapter 4

## Experiments and Setup

All simulations are performed in V-REP, a physics engine that supports a Python API to modify its scenes. `garage` (previously known as RLLab) [Duan et al., 2016] is used to implement and evaluate the deep reinforcement learning algorithms. Unlike MuJoCo and OpenAI Gym, V-REP is not compatible with the newer machine learning frameworks. Thus, some infrastructure work is required to bridge the two platforms. All neural networks and infrastructures are implemented in Tensorflow and Python.

The experiments include running the simulations on a flat surface and a hilly surface. Each simulation consists of 800 time steps, with each simulated time step equals to 3 ms. The robot is equipped with a vision-sensing controller, such that it attempts to walk along the desired paths indicated on the floor. Each experiment contains  $G = 50$  generations. In the end, the results from the two policy gradient methods are compared with the standard leg shape’s performance.

Towards the submission deadline of this report, it is discovered that Equation 3.1 may grow unlimitedly when the distance traveled in the simulation increases. Equation 3.3 is suggested to replace Equation 3.1 in Algorithm 2 to address this issue. Due to the time constraints, Equation 3.3 is only experimented with actor-critic in the scene with a straight path on a flat surface, serving as an additional validator for the proposed algorithm.

Furthermore, for each of REINFORCE and actor-critic, five random starting points are

selected and run to test for how sensitive the proposed algorithm is to the initial vectors. For validation purposes, the same algorithm, trained by crawling on straight paths, is run on the flat and hilly surfaces again, but with curved paths. The hill-climbing optimization method acts as the baseline comparison against the proposed algorithm.

Finally, it is noteworthy that when testing for the number of joints, the more joints a configuration has, the bigger the standard-shaped legs. This is to accommodate the fact that the more joints each leg has, the farther raised from the floor the robot is.



Figure 4.1: The flat and the hilly surfaces for the robot to crawl on.



Figure 4.2: The straight paths followed by the robot in the flat and hilly scenes.

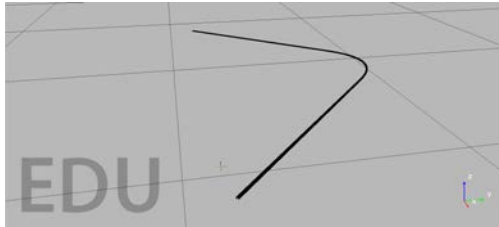


Figure 4.3: The curved paths followed by the robot in the flat and hilly scenes.

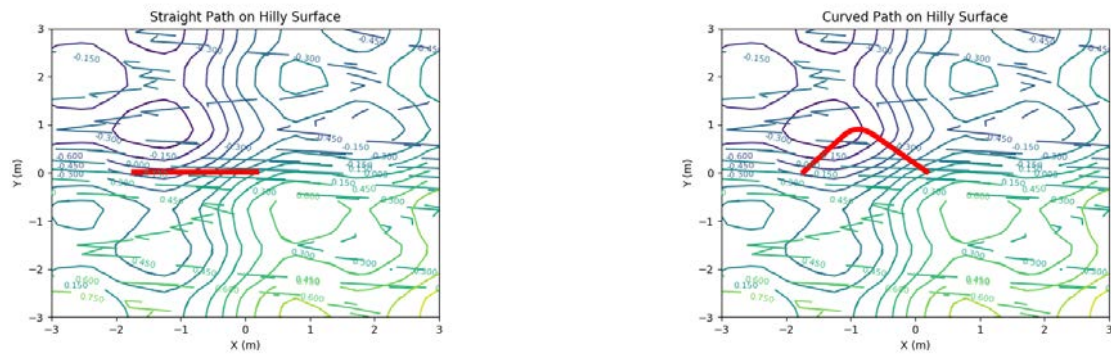


Figure 4.4: The red lines are the ideal paths on the hilly surfaces, plotted in the contour maps for both scenes.

# Chapter 5

## Results

### 5.1 Flat surface, straight path

#### First trial

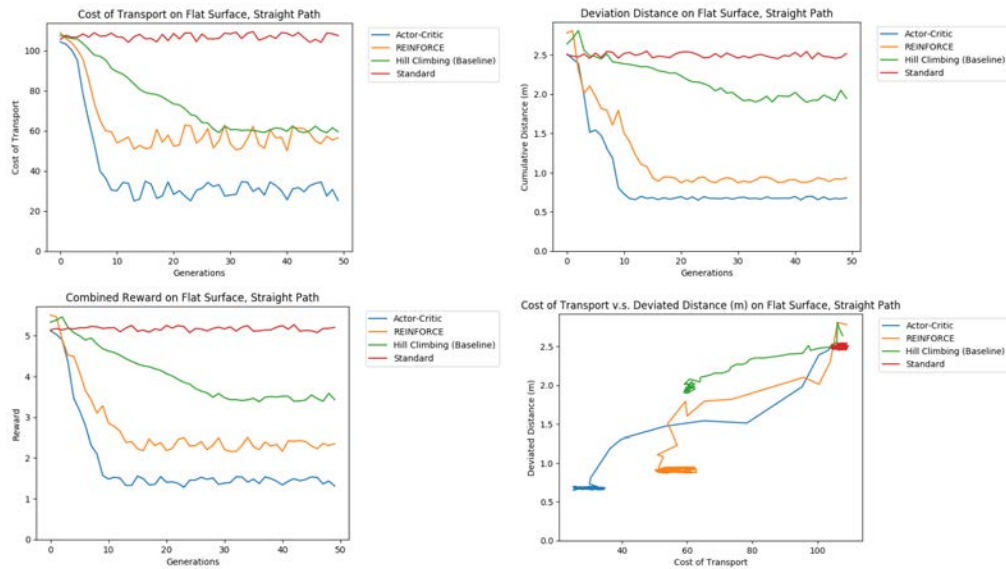


Figure 5.1: Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a straight path over 50 generations.

	Initial Shape	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.074	0.0635	0.0564	0.0831
Width (m)	0.0168	0.0635	0.0564	0.452
Mass (kg)	0.194	0.0081	0.0079	0.0138
Spring Constant (N/m)	0.019	0.0231	0.0296	0.043
Primitive Shapes	Cylinder	Sphere	Sphere	Cylinder
Number of Joints	0	1	1	1

Table 5.1: The final parameters after 50 generations on a flat surface with a straight path.

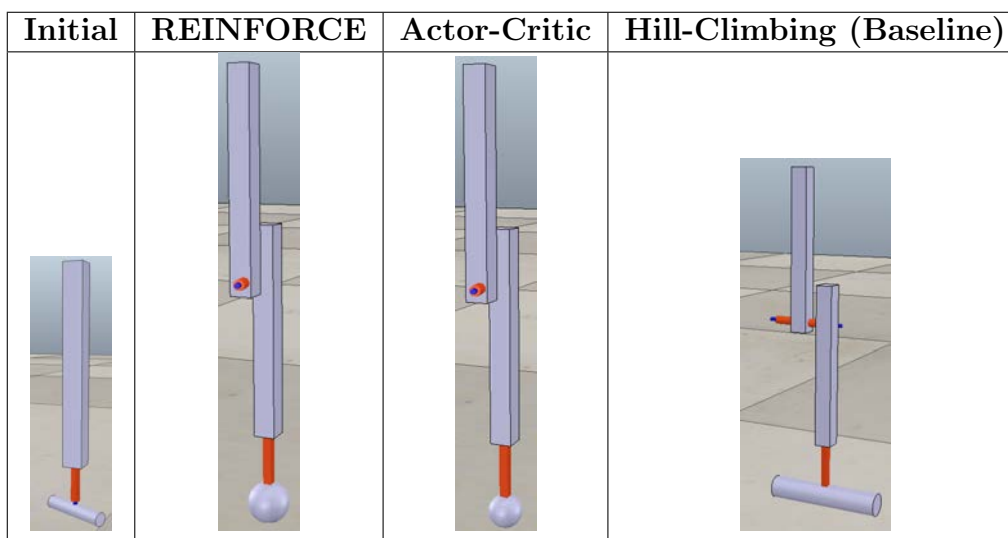


Table 5.2: The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a straight path on a flat surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs.

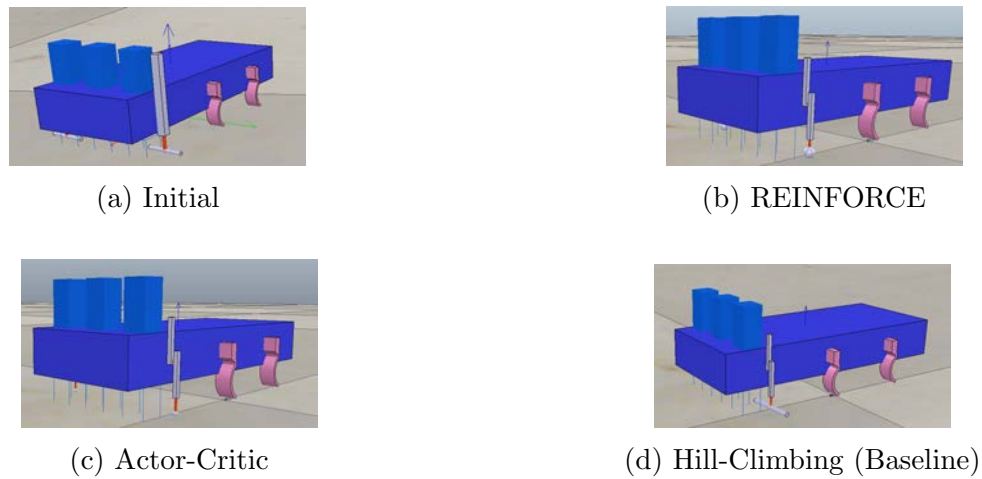


Figure 5.2: The same shapes from Table 5.2 on the robot, for illustration purposes.

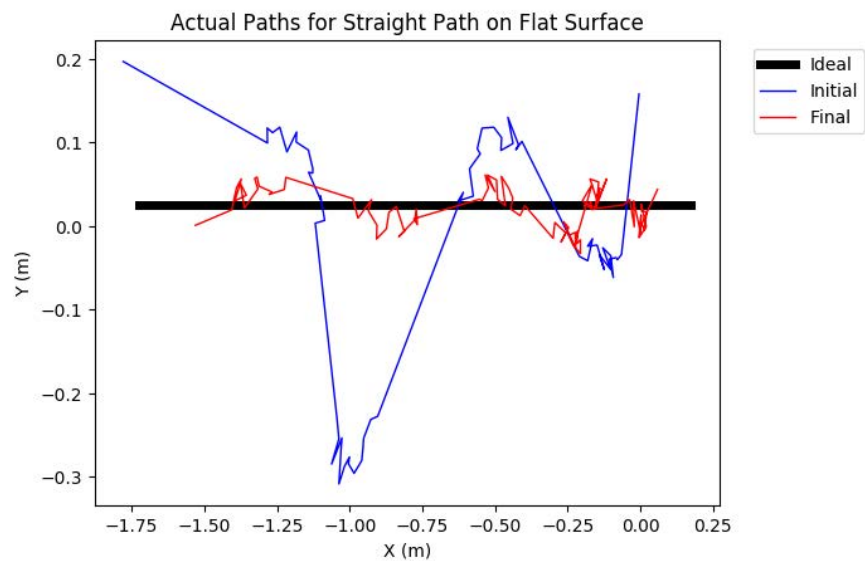


Figure 5.3: The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a flat surface and a straight path.

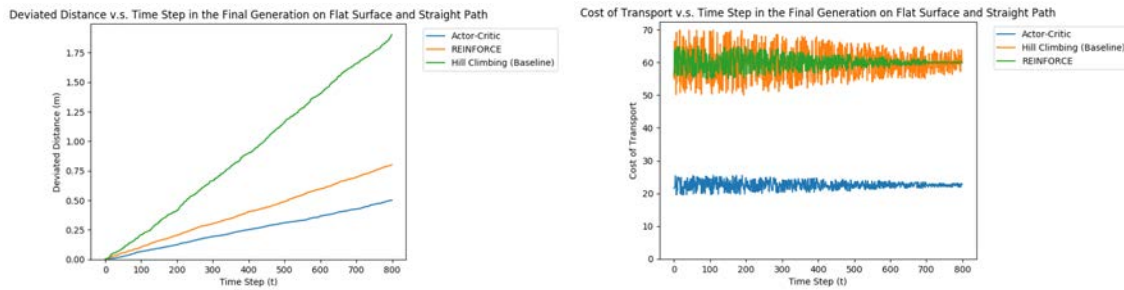


Figure 5.4: Metrics 3.1 (left) and 3.2 (right) for the last generation, on a flat surface and a straight path.

### Trials with five random starting points

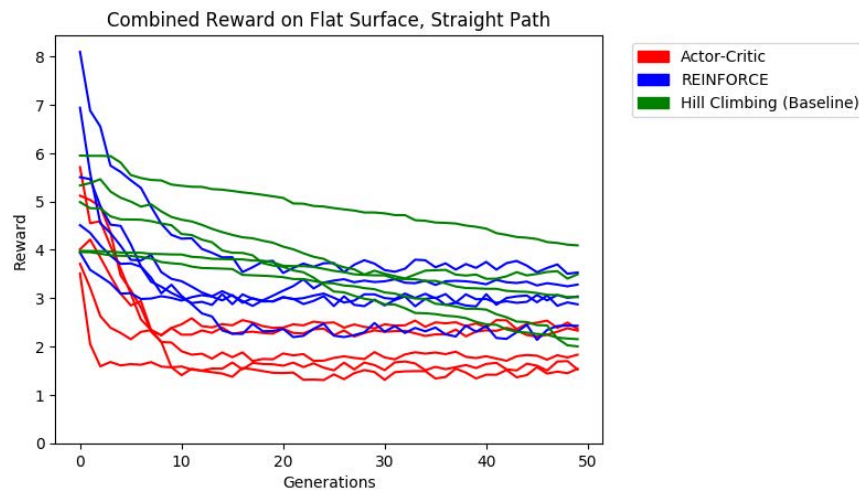


Figure 5.5: Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a straight path over 50 generations. Initialized with five randomly generated configurations for each optimization method.



	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.0478 - 0.07828	0.0361 - 0.0645	0.0579 - 0.187
Width (m)	0.0478 - 0.07828	0.0361 - 0.0645	0.109 - 0.557
Mass (kg)	0.0073 - 0.012	0.0068 - 0.0083	0.0102 - 0.0151
Spring Constant (N/m)	0.012 - 0.029	0.0105 - 0.0358	0.0352 - 0.0563
Primitive Shapes - Sphere	10	10	6
Primitive Shapes - Cube	0	0	0
Primitive Shapes - Cylinder	0	0	4
Number of Joints	1	1	0 - 1

Table 5.3: The ranges of the final parameters for each optimization method with the robot crawling on a flat surface and a straight path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.

### Validating with Equation 3.3 in place of Equation 3.1

Due to the time constraints, Equation 3.3 is only experimented with actor-critic in the scene with a straight path on a flat surface. It is used in place of Equation 3.1 in Algorithm 2.

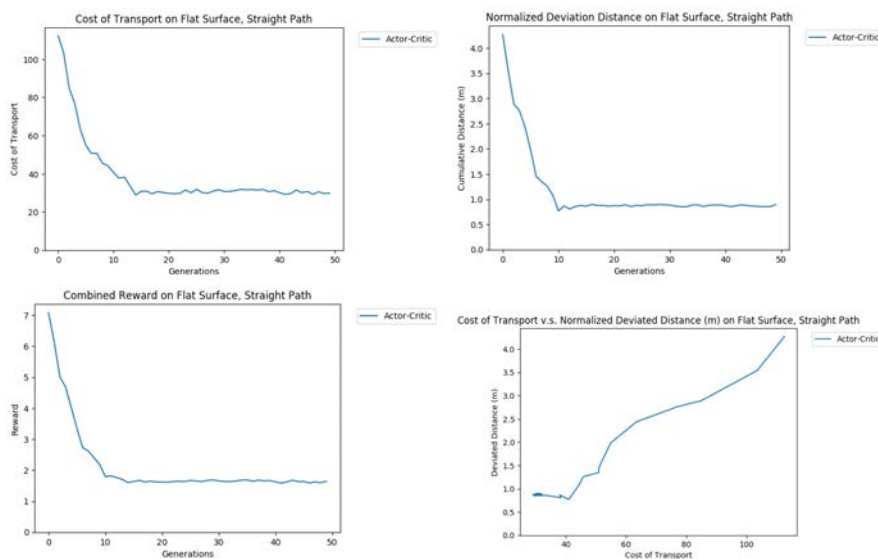


Figure 5.6: Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a straight path over 50 generations. For these experiments, Equation 3.3 replaces Equation 3.1 in Algorithm 2.

## 5.2 Hilly surface, straight path

### First trial

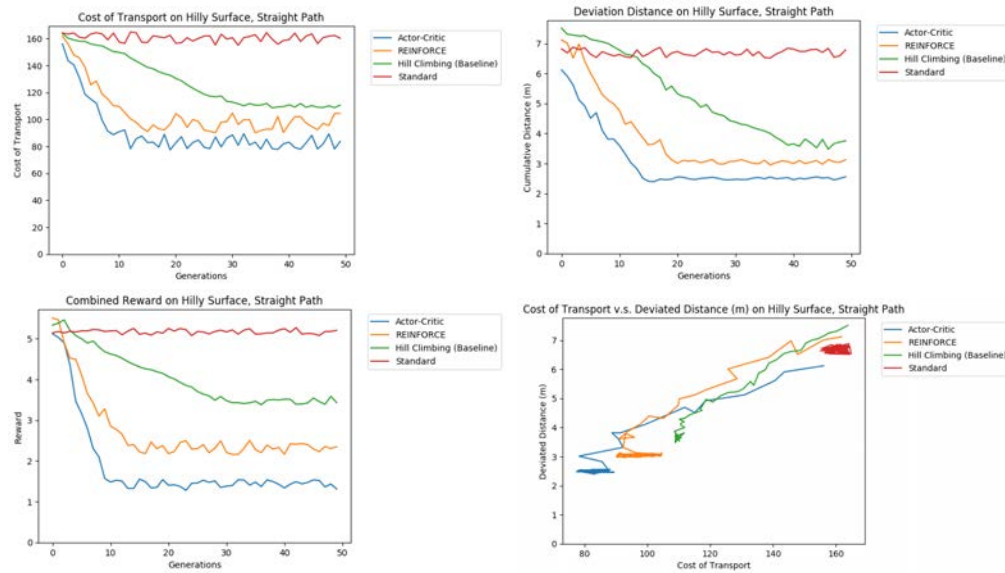


Figure 5.7: Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a straight path over 50 generations.

	Initial Shape	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.068	0.1418	0.0840	0.1791
Width (m)	0.21	0.1418	0.0840	0.240
Mass (kg)	0.179	0.1065	0.0873	0.153
Spring Constant (N/m)	0.001	0.0548	0.0674	0.0589
Primitive Shapes	Cube	Sphere	Sphere	Cylinder
Number of Joints	0	1	1	1

Table 5.4: The final parameters after 50 generations on a hilly surface with a straight path.

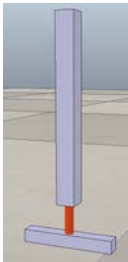


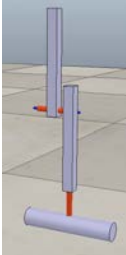
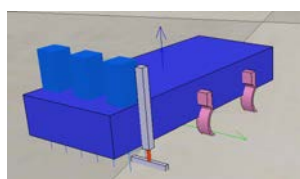
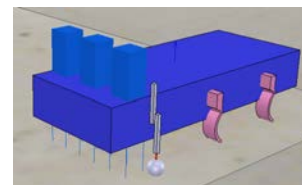
Initial	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
			

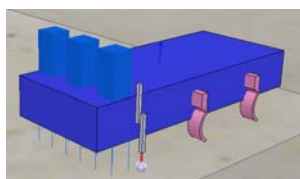
Table 5.5: The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a straight path on a hilly surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs.



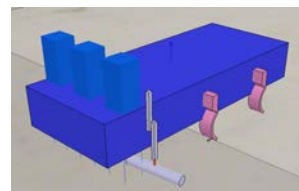
(a) Initial



(b) REINFORCE



(c) Actor-Critic



(d) Hill-Climbing (Baseline)

Figure 5.8: The same shapes from Table 5.5 on the robot, for illustration purposes.

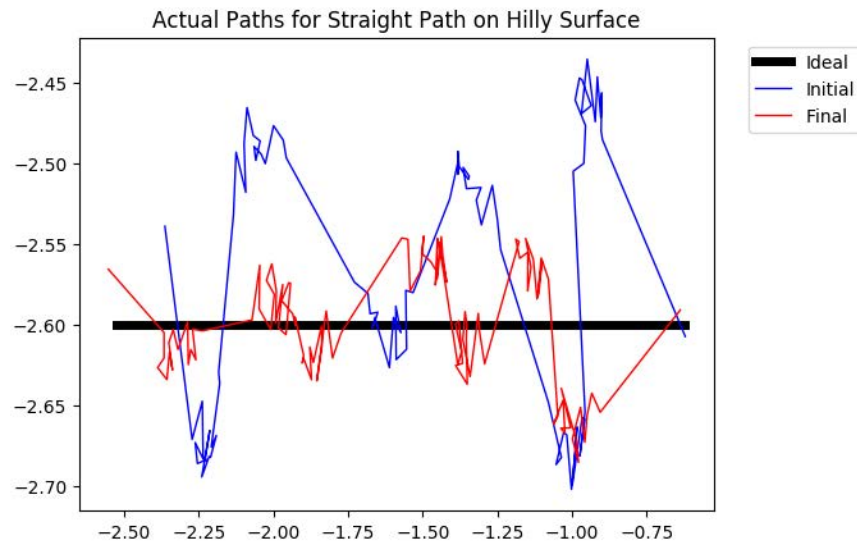


Figure 5.9: The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a hilly surface and a straight path.

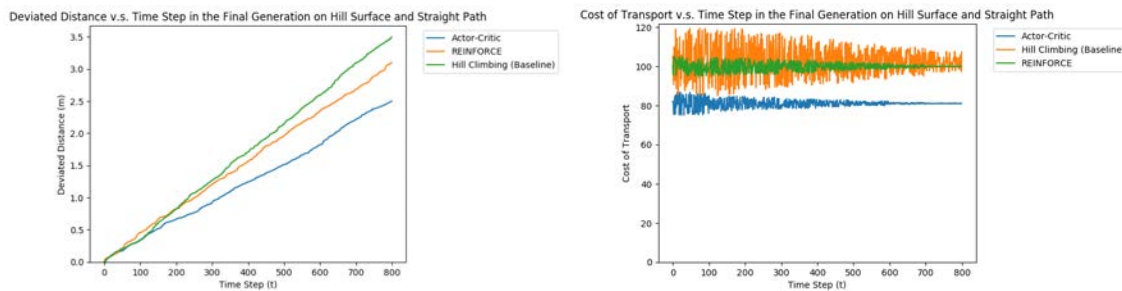


Figure 5.10: Metrics 3.1 (left) and 3.2 (right) for the last generation, on a hilly surface and a straight path.

## Trials with five random starting points

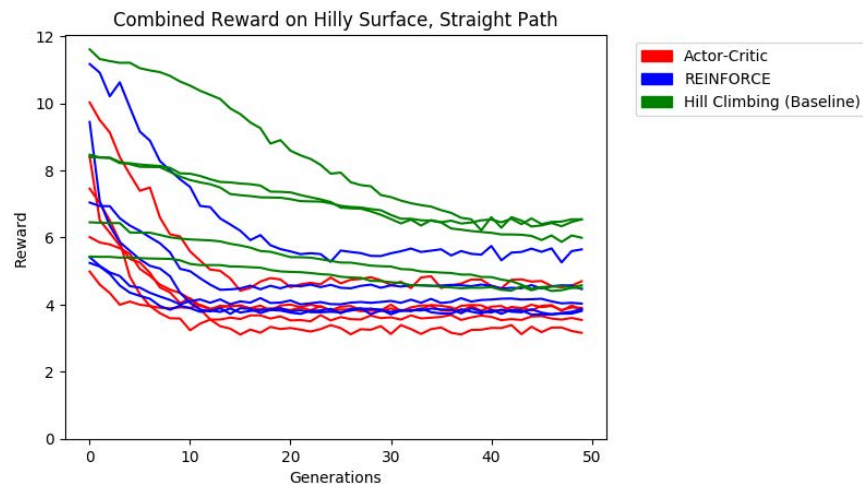


Figure 5.11: Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a straight path over 50 generations. Initialized with five randomly generated configurations for each optimization method.

	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.102 - 0.1418	0.0787 - 0.0844	0.1579 - 0.2384
Width (m)	0.102 - 0.1418	0.0787 - 0.0844	0.209 - 0.282
Mass (kg)	0.0781 - 0.1295	0.0798 - 0.0903	0.101 - 0.162
Spring Constant (N/m)	0.0452 - 0.0748	0.0602 - 0.0689	0.0352 - 0.0872
Primitive Shapes - Sphere	10	10	7
Primitive Shapes - Cube	0	0	0
Primitive Shapes - Cylinder	0	0	3
Number of Joints	0 - 1	0 - 1	0 - 2

Table 5.6: The ranges of the final parameters for each optimization method with the robot crawling on a hilly surface and a straight path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.

### 5.3 Flat surface, curved path

#### First trial

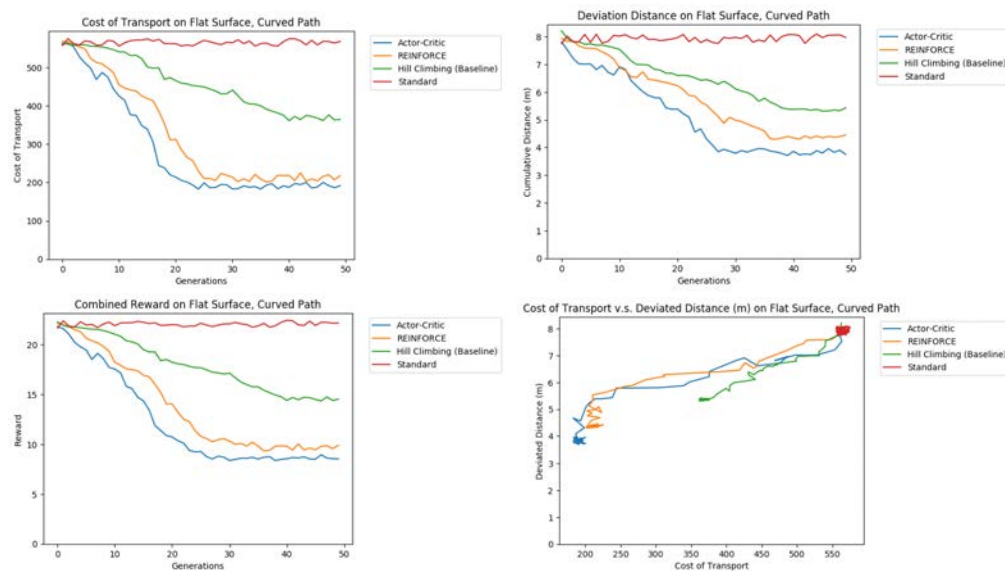


Figure 5.12: Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a curved path over 50 generations.

	Initial Shape	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.094	0.0504	0.0301	0.0651
Width (m)	0.023	0.0504	0.0301	0.0651
Mass (kg)	0.016	0.0051	0.0048	0.0108
Spring Constant (N/m)	0.043	0.016	0.013	0.036
Primitive Shapes	Cube	Sphere	Sphere	Sphere
Number of Joints	2	1	1	1

Table 5.7: The final parameters after 50 generations on a flat surface with a curved path.

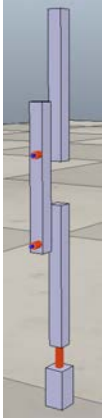

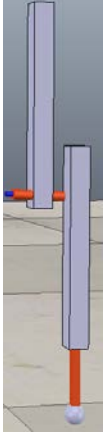
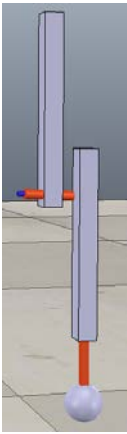
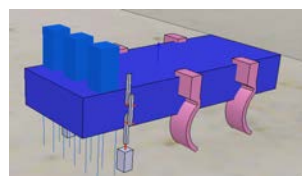
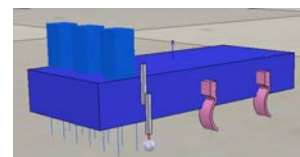
Initial	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
			

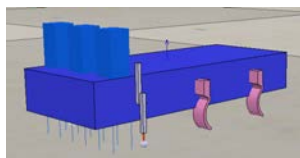
Table 5.8: The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a curved path on a flat surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs.



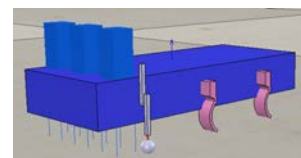
(a) Initial



(b) REINFORCE



(c) Actor-Critic



(d) Hill-Climbing (Baseline)

Figure 5.13: The same shapes from Table 5.8 on the robot, for illustration purposes.

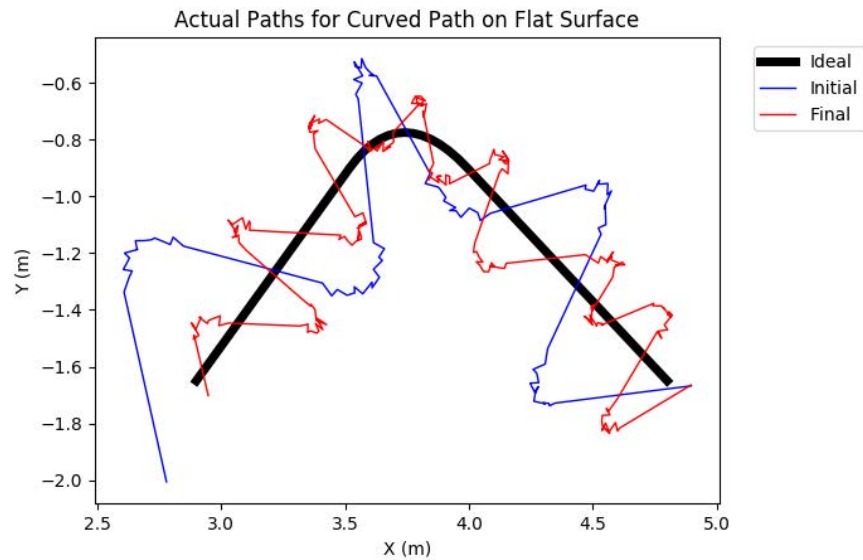


Figure 5.14: The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a flat surface and a curved path.

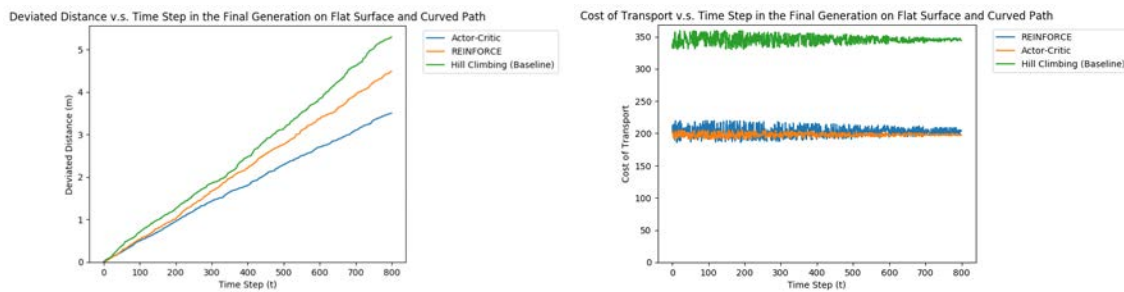


Figure 5.15: Metrics 3.1 (left) and 3.2 (right) for the last generation, on a flat surface and a curved path.



## Trials with five random starting points

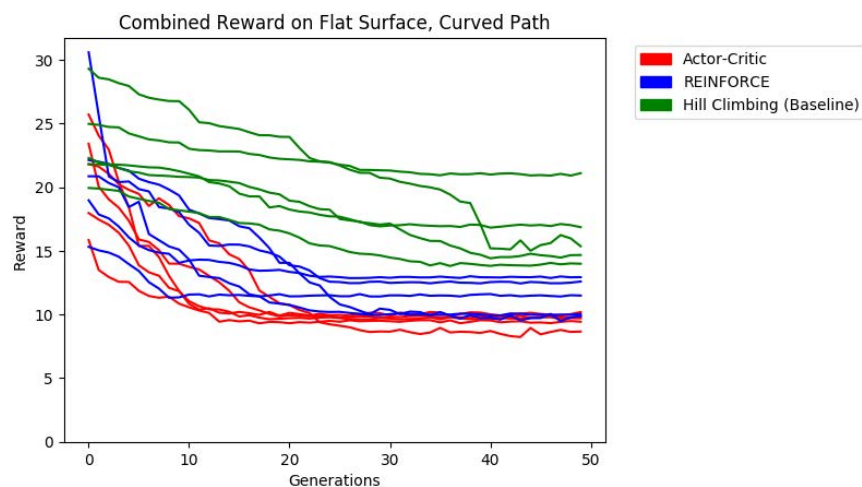


Figure 5.16: Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a flat surface, for the robot walking on a curved path over 50 generations. Initialized with five randomly generated configurations for each optimization method.

	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.0302 - 0.0728	0.0241 - 0.0572	0.0572 - 0.1361
Width (m)	0.0201 - 0.0912	0.0287 - 0.0839	0.0609 - 0.101
Mass (kg)	0.0032 - 0.0069	0.0035 - 0.0083	0.0095 - 0.0169
Spring Constant (N/m)	0.011 - 0.0223	0.002 - 0.0209	0.0203 - 0.0369
Primitive Shapes - Sphere	8	9	7
Primitive Shapes - Cube	0	0	1
Primitive Shapes - Cylinder	2	1	2
Number of Joints	0 - 1	0 - 1	0 - 2

Table 5.9: The ranges of the final parameters for each optimization method with the robot crawling on a flat surface and a curved path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.

## 5.4 Hilly surface, curved path

### First trial

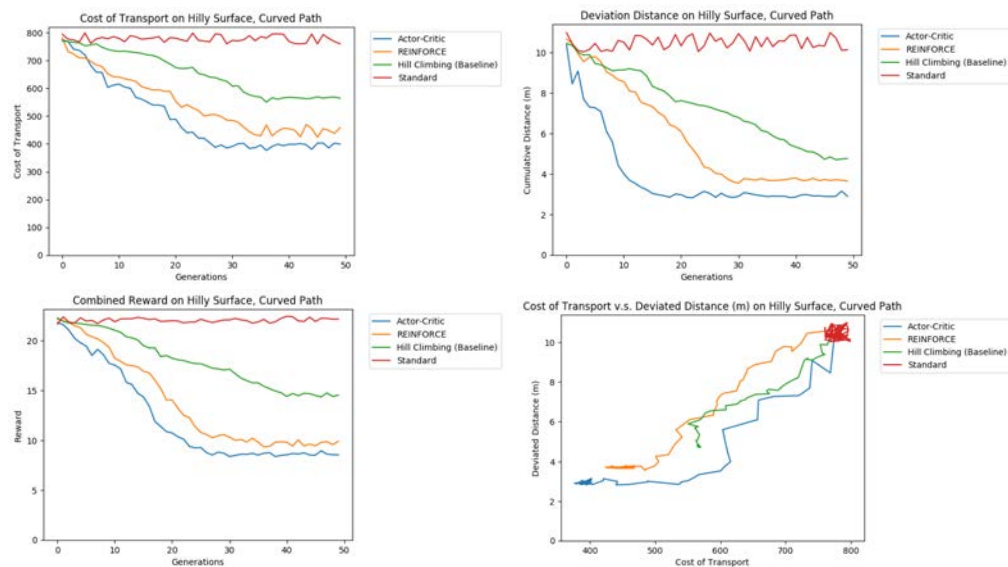


Figure 5.17: Learning curves for the standard shape, hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a curved path over 50 generations.

	Initial Shape	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.254	0.184	0.143	0.204
Width (m)	0.43	0.184	0.143	0.339
Mass (kg)	0.13	0.063	0.054	0.172
Spring Constant (N/m)	0.273	0.076	0.068	0.096
Primitive Shapes	Cylinder	Sphere	Sphere	Cube
Number of Joints	3	1	1	1

Table 5.10: The final parameters after 50 generations on a hilly surface with a curved path.

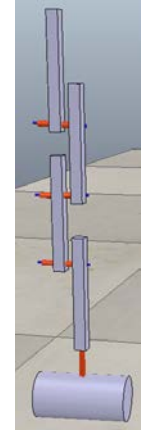
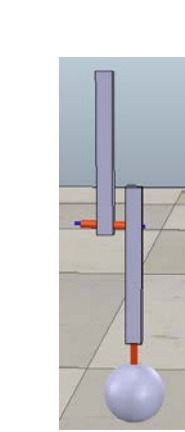
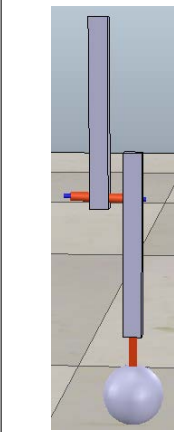
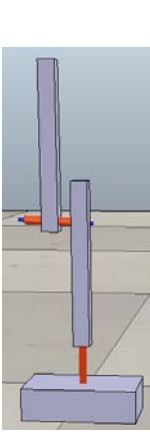
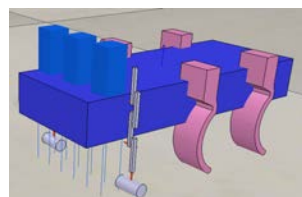
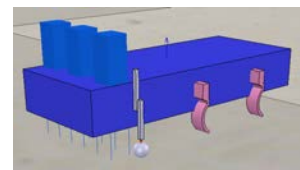
Initial	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
			

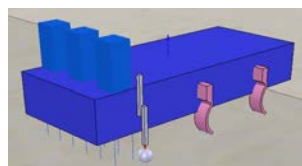
Table 5.11: The initial shape that serves as the starting point of the algorithm, and the shapes learned with crawling on a curved path on a hilly surface with REINFORCE, actor-critic, and hill-climbing (baseline), respectively, after 50 generations. The variables optimized are the shapes of the tips, the sizes of the tips, the spring constants of the joints connecting the tips and the legs, and the numbers of joints in the legs.



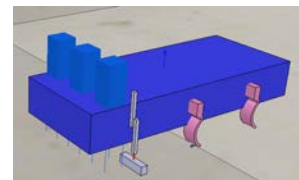
(a) Initial



(b) REINFORCE



(c) Actor-Critic



(d) Hill-Climbing (Baseline)

Figure 5.18: The same shapes from Table 5.11 on the robot, for illustration purposes.

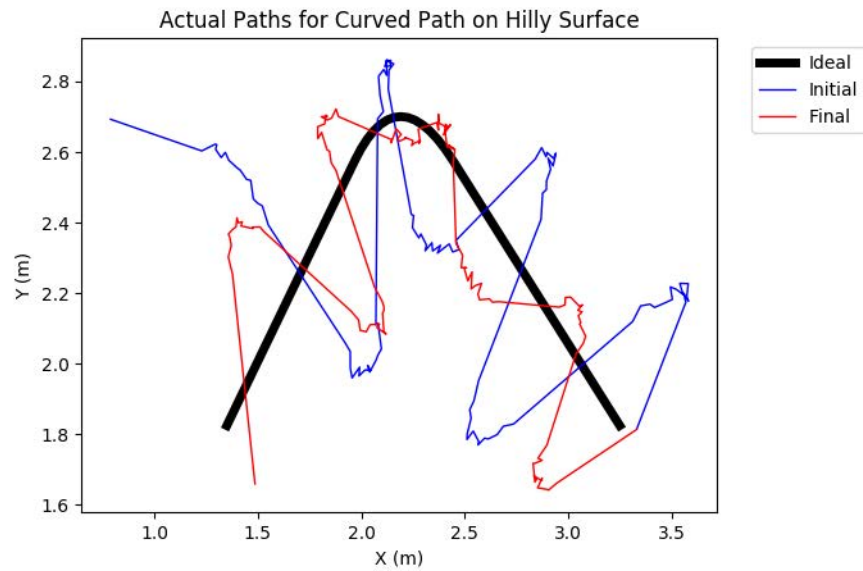


Figure 5.19: The ideal path (black) comparing to the initial path (blue) and the final path optimized with actor-critic (red). The robot is attempting to crawl on a hilly surface and a curved path.

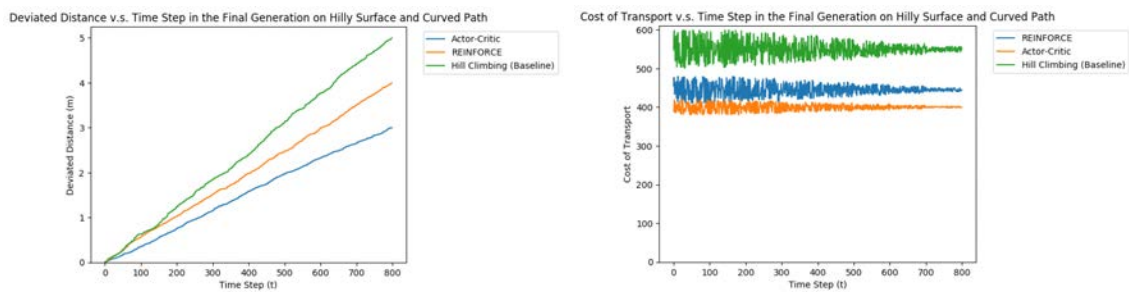


Figure 5.20: Metrics 3.1 (left) and 3.2 (right) for the last generation, on a hilly surface and a curved path.

## Trials with five random starting points

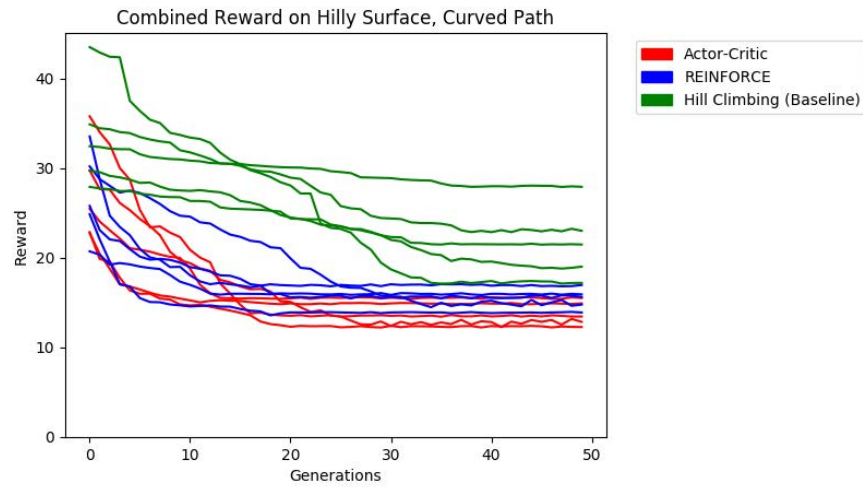


Figure 5.21: Learning curves for hill-climbing (baseline), REINFORCE, and actor-critic on a hilly surface, for the robot walking on a curved path over 50 generations. Initialized with five randomly generated configurations for each optimization method.

	REINFORCE	Actor-Critic	Hill-Climbing (Baseline)
Height (m)	0.152 - 0.2729	0.141 - 0.1773	0.142 - 0.2861
Width (m)	0.132 - 0.298	0.1235 - 0.2391	0.2009 - 0.401
Mass (kg)	0.062 - 0.078	0.055 - 0.081	0.086 - 0.173
Spring Constant (N/m)	0.052 - 0.083	0.053 - 0.098	0.073 - 0.1386
Primitive Shapes - Sphere	7	8	4
Primitive Shapes - Cube	0	0	3
Primitive Shapes - Cylinder	3	2	3
Number of Joints	1	1	1 - 2

Table 5.12: The ranges of the final parameters for each optimization method with the robot crawling on a hilly surface and a curved path. Each trial starts at a randomly generated configuration and runs for 50 generations. For the primitive shapes, the number in each row represents how often the shape is chosen as the final shape.

# Chapter 6

## Analysis

The results exhibit that the proposed algorithm is in the correct direction. As shown by the learning curves, REINFORCE makes some significant improvements from the original leg shape design, and actor-critic enhances the performances further, especially in the later generations. The actor-critic version has the best performance as expected given the noise and high variance in vanilla REINFORCE.

It is also spellbinding to observe a somewhat linear relationship between the cost of transport and the deviated distance. This information further strengthens the argument that the algorithm is useful, given that it can enhance the performances in the two metrics together.

The final parameters in Figures 5.2, 5.8, 5.13, and 5.18 are reasonable. On a flat surface, the robot does not need huge tips for crawling, so the algorithm favors the lighter and shorter legs. The spring constants for the flat surface are lower as well, indicating that the springs do not have to be as stiff to move around smoothly. On the uneven surface, the robot needs to have bigger and heavier legs to get over the slope and travel farther along the path. The spring constants demonstrate that the joints need to be stiffer and more durable to move across the bumpy surface. Spheres are chosen as the primitive leg shapes in both optimization methods. It is an encouraging sign because intuitively, circular shapes, such as wheels on cars, have the smoothest movement.

An intriguing investigation is on the reason for spheres to be preferred over cylinders using

the two deep reinforcement learning algorithms. Spheres are expected to outperform cubes. However, cylinders, whose surfaces are also curvy, are expected to do just as well as spheres. Through further exploration of the results, cylindrical legs tend to have higher deviated distances and costs of transport than spheres. A justifiable explanation is that, since the robots behave in a slight hopping motion, the edges of the cylinders occasionally touch the surface and affect the robot’s locomotion. With the spherical legs, since there are no edges, the robot can balance itself effortlessly even if it ends up hopping. It is also noteworthy that cylindrical legs finish with denser masses than spheres’, which handicap the cost of transport for cylindrical legs.

Another interesting observation is that the final number of joints per leg is one in both algorithms. The number of joints is explored because some insects have more than one joint in their legs [Cruse et al., 2009]. This outcome resembles the evolution of human legs; having a single rotational joint, like a knee for each leg, is the most beneficial for walking across surfaces. A plausible conviction is that for insects such as grasshoppers, they have more joints in their legs because their fundamental movement is to jump.

Hill-climbing, acting as the baseline, has worse performances than those of the proposed algorithm in all scenarios. Some possible causes include the facts that the number of iterations is not enough for hill-climbing to find the optima, and that the algorithm is trapped in local optima. Hill-climbing is also more unstable and converges slower. On the other hand, it ends up with the 1-joint design too, even though the shapes of the tips are different from REINFORCE and actor-critic in some experiments. This observation further suggests that the 1-joint design is beneficial for the locomotion in this study.

The preliminary results for using Equation 3.3 in place of Equation 3.1 are promising, as they produce similar results as the rest of the experiments in the project. However, due to the time constraints, Equation 3.3 is only applied to the scene with a flat surface and a straight path. It is still not entirely verified if the possibility of Equation 3.1 growing unboundedly would not affect the objective function.

Moreover, the algorithm trained with straight paths is run with curved paths for validation

reasons. With the curved paths, the proposed algorithm's performances are slightly more unstable than with the straight paths. The convergences are also slower. However, these discrepancies are still within the expectations, given that the control for the curved paths is not optimized. Furthermore, under the same control and environment, the proposed algorithm still beat the standard shapes and the hill-climbing method in the two chosen metrics. Another captivating remark is that the proposed algorithm can crawl on the paths in a more fine-grained manner. For example, from the results in Figure 5.14, the robot's final path is more delicate than the initial path. This observation further suggests that the proposed method is effective.

Finally, to test for the sensitivity to the starting points, five randomly generated starting points are run with the proposed algorithm. The results suggest that for the flat scenes, despite starting at different configurations, the algorithm can arrive at similar results, and is more robust to starting configurations than hill-climbing. For the uneven surfaces, the results are more unstable. The proposed algorithm has some divergence among the five trials, but it is less significant than hill-climbing's. This observation is possibly due to the fact that the existing motor control is suboptimal for the bumpy surfaces, again.



# Chapter 7

## Conclusion

In this study, a novel evolutionary robotics algorithm is proposed for simulated robots to adapt their body parts to crawl on specific terrains. The new method retains the evolutionary favor as it is still run for a certain number of generations. It also blends in REINFORCE and actor-critic methods from deep reinforcement learning, hoping to acquire a more precise and faster convergence. Based on the experimental results, with the setup described in this paper, the proposed algorithm outperforms hill-climbing, in both speed and accuracy.

# Chapter 8

## Future Work

A continuation of this project is to verify if Equation 3.1 growing unlimitedly affects the objective function. Because of the shortage of time, Equation 3.3 is only analyzed in the flat scene on a straight path with actor-critic. Applying Equation 3.3 to every experiment is the next advancement.

A potential extension is the optimization of the weights between the two metrics. A common practice is to tune for favorable weights for each objective function. The current weight pair  $(0.025, 1)$  is an acceptable starting point, but more sophisticated weights can further improve the performances. It is also likely that each terrain requires its specific weight factors, given that the ratios between deviated distance and cost of transport are not uniform across all terrains.

Currently, there are only limited selections of shapes for the tips of the legs. They are all the built-in primitive shapes from V-REP. If the algorithm can support customized shapes without sacrificing too much of the efficiency (for example, using Bezier splines [Collins et al., 2018]), the additional flexibility can further expand the search space and increase the possibility of identifying more optimal shapes.

Another extension is the simultaneous adaption of designs and control. For this report, an elementary vision-based controller is used. There is already research on learning the control and body parts in an alternating fashion [Schaff et al., 2019]. Given that each leg shape and

terrain combination is likely to have its particular optimal control, it would be intriguing to apply the same principle to this study.

Finally, the most significant extension is experimenting with actual robots. Due to the circumstances of COVID-19, the original plan of validating the algorithm via live robotic experiments has been disrupted. It would have been fascinating to observe the differences between simulations and reality and to discover ways to mitigate the divergence.

# Bibliography

- [1] Arias-Montano, A., Coello, C. A. C., and Mezura-Montes, E. (2012). Multiobjective evolutionary algorithms in aeronautical and aerospace engineering. *IEEE Transactions on Evolutionary Computation*, 16(5):662–694.
- [2] Collins, J., Geles, W., Howard, D., and Maire, F. (2018). Towards the targeted environment-specific evolution of robot components. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 61–68.
- [3] Corucci, F., Cheney, N., Giorgio-Serchi, F., Bongard, J., and Laschi, C. (2018). Evolving soft locomotion in aquatic and terrestrial environments: effects of material properties and environmental transitions. *Soft robotics*, 5(4):475–495.
- [4] Cruse, H., Dürr, V., Schilling, M., and Schmitz, J. (2009). Principles of insect locomotion. In *Spatial temporal patterns for action-oriented perception in roving robots*, pages 43–96. Springer.
- [5] Duan, Y., Chen, X., Houthoofd, R., Schulman, J., and Abbeel, P. (2016). Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338.
- [6] Ha, D. (2019). Reinforcement learning for improving agent design. *Artificial life*, 25(4):352–365.
- [7] Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014.

- [8] Nolfi, S., Floreano, D., and Floreano, D. D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT press.
- [9] Schaff, C., Yunis, D., Chakrabarti, A., and Walter, M. R. (2019). Jointly learning to construct and control agents using deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9798–9805. IEEE.
- [10] Sims, K. (1994a). Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372.
- [11] Sims, K. (1994b). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22.
- [12] Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V. (2018). Sim-to-real: Learning agile locomotion for quadruped robots. *arXiv preprint arXiv:1804.10332*.
- [13] Tricoli, V., Lamas, L., Carnevale, R., and Ugrinowitsch, C. (2005). Short-term effects on lower-body functional power development: weightlifting vs. vertical jump training programs. *The Journal of Strength & Conditioning Research*, 19(2):433–437.
- [14] Wang, L., Yang, Y., Correa, G., Karydis, K., and Fearing, R. S. (2019). Openroach: A durable open-source hexapedal platform with onboard robot operating system (ros). In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9466–9472. IEEE.
- [15] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.