Vrije Universiteit Amsterdam

Bachelor Thesis

Exploring Evolutionary Algorithms for Solving the Artificial Ant Task

---

Author: Anesa Ibrahimi
(2693458)
a2.ibrahimi@student.vu.nl

1st supervisor: Prof. dr. Anil Yaman
2nd reader: Prof. dr. Nicolas Cambier

A thesis submitted to the Faculty of Science for the Bachelor Degree of Artificial Intelligence

June 29, 2023

Abstract

Evolutionary algorithms have notably been utilized in various optimization tasks, where bio-inspired natural selection processes are the core of its performance. In this thesis, a deeper inspection of the application of evolutionary algorithms is provided to address the challenges of locomotion and interaction behavior of an artificial ant from the MuJoCo environment. Ultimately, aiming to discover the potential of these algorithms in tackling the optimization problem specified above while also uncovering insights during their utilization and impact in various simulated virtual environments. The algorithms which have been used throughout this research were three main evolution strategies algorithms, namely, CMA-ES, XNES, SNES. These algorithms have been applied during the evolution phase of the ant agent which spawned and resided in 100 different virtual environments consisting of various amount and height of box obstacles. The evaluation and inspection of the impact of these algorithms on the ant agent were aided through quantitative data as well as qualitative data. Moreover, the quantitative data was provided through the fitness-value minimization task which showed the overall dominance of the SNES algorithm in comparison to the remaining algorithms while the qualitative data provided clear visual insights showing the different exploratory behaviours of the algorithms. Even though, all of the algorithms managed to generate a locomotion behaviour suitable for the ant agent to walk over the obstacles and the coordinate plane, further extensive research in the future is necessary to address the hyperparameter sensitivity in the evolutionary & simulation phase of the research as well the employment of the parallelization technique when evolving the ant through the simulated environments.

Contents

1 Introduction

The phenomenon of natural selection processes has been widely studied and documented in various literature, aiming to grasp, replicate, and advance the mechanism of evolution [1]. In the midst of these literatures a significant area of computer science, Evolutionary Computation, started using the concepts of biological evolution and applying them to various computational problems [2]. This collaborative effort among Evolutionary Algorithms (EA) researchers has led to the development of three prominent types of EA: Evolutionary Programming (EP), Genetic Algorithms (GA), and Evolution strategies (ES) [1].

In recent years, evolutionary algorithms have gained significant attention as powerful optimization techniques inspired by natural evolution processes. One particular area of interest is the application of evolutionary strategies in reinforcement learning and optimization tasks. In this thesis, the effectiveness of various evolutionary strategies, including Covariance Matrix Adaptation Evolution Strategy (CMA-ES), Exponential Natural Evolution Strategies (XNES), and Separable Natural Evolution Strategies (SNES), are explored in the context of training an artificial ant in the OpenAI Gym environment. Reinforcement learning enables an agent to learn optimal behaviors through trial-and-error interactions and delayed reward within an environment. However, traditional reinforcement learning approaches often suffer from slow convergence and struggle with high-dimensional and complex tasks. Moreover, the balance between exploration and exploitation in various tasks becomes a challenging aspect to manage [3]. Evolution strategies provide an alternative approach that can address some of these challenges by adopting the principles of natural evolution.

The OpenAI Gym environment provides a collection of reinforcement learning tasks, including simulated environments that simulate real-world scenarios. In this thesis, the main focus is on training an artificial ant, which serves as a challenging and dynamic task. The ant environment requires the agent to learn complex movement patterns and adapt to varying terrain and obstacles. By applying different evolution strategies, the aim is to evaluate their performance and effectiveness in optimizing the ant's movement behavior. The specific evolution strategies include CMA-ES, XNES, and SNES. These strategies have shown promising results in various optimization domains. By comparing their performance on the artificial ant task, insights into their strengths and limitations will be provided. This research contributes to the broader field of evolutionary computation and reinforcement learning by exploring the applicability of evolution strategies in complex and dynamic environments from MuJoCo environment and the impact on the interaction between the ant and environment obstacles.

The remainder of this thesis is organized as follows. Chapter 2 lays the foundational concepts required to understand the subsequent chapters. Chapter 3 provides a comprehensive literature review of related work in the field of evolutionary algorithms, and the specific strategies being explored. Chapter 4 presents the research hypothesis and defines the aim of the study. The methodology and experimental setup used to implement and evaluate the evolutionary strategies on the artificial ant task are included in Chapter 5. Furthermore, it provides in-depth evaluation methods and approaches of the experiments. Chapter 6 discusses the results and analyzes the performance of each strategy. Lastly, Chapter 7 concludes the thesis with limitations, and suggestions for future research while also providing a summary of the findings, and an overall conclusion based on the achieved results of this paper.

2 Background

Evolution strategies (ES), a type of black-box optimization algorithms, originate from the Evolutionary algorithms where the optimizations involve real number vectors [4]. As mentioned earlier in the section above, EA are inspired by natural selection processes. In other words,the characteristics and attributes of a surviving individual that managed to endure various obstacles are then ultimately passed down to the next generation, which consequently allows the population to grow and adapt to the new inherited attributes[4].

2.1 Evolution Strategies (ES)

The most simplistic variant of the ES is called Simple Gaussian ES. Due to its distinctive approach in terms of function information, it is able to be set apart from traditional optimization algorithms since it does not require any explicit gradient information. With this in mind, the application of ES can be used in abstract optimization problems. The following two equations retrieved from [5] show the general approach which ES applies. The optimization target function (F) w.r.t to certain user-defined parameters (θ) gets updated at each time step (t). The summation term then determines the direction of the gradient, by computing the weighted sum of a population (n). This population is comprised of randomly sampled points around current position θ_t.

$$\theta_{t+1} = \theta_t + \frac{\alpha}{n\sigma} \sum_{i=1}^{n} F\left(\theta_t + \sigma\epsilon_i\right) \epsilon_i$$

$$\epsilon_i \sim \mathcal{N}(0, I)$$

This model tracks the mean (α) and standard deviation(σ) values which are then used to generate the offspring population size by sampling the Gaussian distribution (ϵ_i). A top performing sample alongside the optimal function value is selected which is used as a benchmark during next generation estimations [4]. Although this simplistic evolutionary strategy involves a straight-forward process it evidently shows some of the drawbacks of the general behaviour of ES. More specifically, the dependency on the σ value which controls the scale of distribution and step-size [4]. Referring back the equation above, the next generation's σ value is highly correlated and dependent on the previous generation. This implies that it is not able to adjust the exploration space efficiently [4].

2.1.1 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

The Co-Variance Matrix Adaptation evolution strategy also known by its short abbreviation (CMA-ES) has been known to fix the very issue stated above. A distinctive feature of CMA-ES is how it tracks dependencies. The dependencies between samples are tracked with a covariance matrix **between** samples. The value of σ (a.k.a standard deviation) is separated from the covariance matrix, which enables it to change steps faster rather than the full covariance [4]. With this in mind, the evaluation of the various step sizes is done through an evolution path (summing consecutive steps) and comparing it with a randomly selected expected length. In the following equation retrieved from [6], the sampling of search points is demonstrated:

$$\boldsymbol{x}_k^{(g+1)} \sim \boldsymbol{m}^{(g)} + \sigma^{(g)} \mathcal{N}\left(\mathbf{0}, \boldsymbol{C}^{(g)}\right) \quad \text{for } k = 1, \ldots, \lambda$$

The equation above, contains several key components. The most prominent one, being the usage of the Covariance matrix C. Some of the beneficial features of the covariance matrix which have been further examined in [4] are the matrix's orthogonal eigenvectors, real non-negative numbered eigenvalues and its diagonalizable property. Other components of the algorithm, such as (μ) and (σ) value are important in scaling the algorithm's distribution as well as in evaluating the step size and evolution path. The ($m^{(g)}$) value denotes the mean value of search distribution at generation g. The k value in ($\boldsymbol{x}_k^{(g+1)}$) (a.k.a k-th offspring from generation g+1)) denotes the individual/search point while the g value is used to denote the current generation. As previously mentioned, the covariance matrix is a distinctive asset of the algorithm and therefore involves two notable methods used to update itself to increase the probability of reproducing the successful search directions in the previous generations [6]. The first method (Rank-μ-update) aims at achieving fast search through a small population size. Subsequently, information from previous generations is used in which the mean of the estimated covariance matrices from all generations turns into an estimator of steps. This in effect allows a large number of generations to be used , leading in a faster adaptation of the covariance matrix [6]. The second method (Rank-one-update) on the other hand aims at estimating the moving steps through the information passed on by evolution path which is then used to track the sign information [4]. The distinction and explanation of the these two methods clarify their capabilities as well as the general adaptation of the algorithm. With this in mind, the following equation retrieved from [6] depicts the combination of these two methods in the overall adaptation of the Covariance matrix and optimization step.

$$\boldsymbol{C}^{(g+1)} = (1 - c_1 - c_\mu \sum w_j)\boldsymbol{C}^{(g)}$$
$$+ c_1 \underbrace{\boldsymbol{p}_{\mathrm{c}}^{(g+1)}\boldsymbol{p}_{\mathrm{c}}^{(g+1)}}_{\text{rank-one update}} + c_\mu \underbrace{\sum_{i=1}^{\lambda} w_i \boldsymbol{y}_{i:\lambda}^{(g+1)} \left(\boldsymbol{y}_{i:\lambda}^{(g+1)}\right)^\top}_{\text{rank- } \mu \text{ update}}$$

To further understand the notations of the equation above, an explanation of various variables will be provided. In order to adapt the covariance matrix at a given generation, the rank-one update learning rate c_1 and rank-mu learning rate c_μ alongside its corresponding weights w_j are extracted and subtracted from one another. Consequently, it is then added to the value attained from the rank-one update which extracts the evolution path at generation g ($p_{\mathrm{c}}^{(g+1)}$) and added to the result retrieved from the rank - μ update which extracts the weights (w_i) from the vectors (y_i) of the current offspring.

2.2 Natural Evolution Strategies (NES)

Natural Evolution strategies, similar yet noteworthy algorithms of the evolutionary algorithm family, expand and offer a different approach in their optimization methods. A detailed research on the (NES) algorithms have been thoroughly studied by [7] where the distinguishing attributes of the NES algorithms have been claimed to be its usage of natural gradients and the updating method during its evolution process. As opposed to the algorithms above, the optimization process is aimed at iteratively updating a search distribution while looking for the steepest direction within small steps. The usage of natural gradients normalizes the updated step in relation to uncertainty which , according to [7] , assists in preventing premature convergence, oscillations etc. Aside from containing distinctive search methods, its performance and robustness techniques are significant features that enable NES's performance to be set apart. Fitness shaping and adaptation sampling techniques are worthy of accreditation [4]. The fitness shaping technique is employed to use the rank under

monotonous increasing fitness values while the adaptation sampling is used to automatically adapt the settings in an efficient way in order to reduce hyper-parameter design choices (e.g. learning rate) [7].

2.2.1 Exponential Natural Evolution Strategies (XNES)

Although the section above managed to lay several characteristics and benefits of NES, the formation of the exponential NES (XNES) has been in response to some drawbacks of NES algorithm. The work laid in [8] highlights the unreliability of the natural gradient estimations w.r.t high dimensional problems. Thus, the research that has been conducted in [8] introduced the usage of an exponential parametrization of search distribution. During the gradient steps on the covariance matrix, the matrix itself requires a positive valued matrix. However, the natural gradient will not always guarantee or output the positive definite covariance matrix. Through the usage of the exponential mapping, it has been noted by [7] that any update will ensure a valid covariance matrix and thus making the gradient invariant to linear transformations of the search space. This updating method of natural gradient of the XNES algorithm has also shown to be similar to that of the CMA-ES algorithm [8]. Even though XNES and CMA-ES contain different parameters for their mean, scale and shape of distribution it has been noted by [7] that the representation of the covariance matrix of XNES is $(\sigma * C)$ where C is any positive definite symmetric matrix and therefore the representation of the search distribution scale is shared by σ, C values of CMA-ES. The research [7] goes on by highlighting the similarities of these algorithms w.r.t to their invariance properties and updates which in return allow XNES to benefit from CMA-ES's hyperparamter settings. Considering the extensive break-down of the XNES algorithm and shared similarities with CMA-ES that were provided above, significant differences have also been noted by [7]. These differences concern the behavior of the CMA-ES algorithm towards quadratic local optima. It has been observed that CMA-ES descends faster into approximately quadratic local optima. On the other hand, xNES demonstrates better resistance to premature convergence. Therefore, in latter chapters , the comparison of the performances of these algorithms will be provided alongside their implications and take-away.

2.2.2 Separable Natural Evolution Strategies (SNES)

Lastly, this chapter is concluded with one more variant of NES that will also be used throughout the remaining of this paper. This evolution strategy (Separable NES) is responsible of adapting the invariance properties alongside the class of search distribution. This variant, among many, was adopted when the adaptation of full covariance matrix of search distributions was necessary in high dimensional search spaces. The conducted research in [7] reasoned that the required adaptation of the full covariance matrix leads to potential high computational costs and sample inefficiency. Therefore, it has been proposed by [7] to surrender some invariance properties of the search algorithm and restricting the search distribution with only the diagonal covariance matrix (the following equation retrieved from [7] depicts it).

$$p(\mathbf{z} \mid \theta) = \prod_{i=1}^{d} \tilde{p}\left(\mathbf{z}_i \mid \theta_i\right)$$

This essentially can be interpreted as restricting a general class of multi-variate search distributions to separable ones and thus allowing a faster adaptation of its parameters [7]. The formula given above, represents the probability density function (pdf) of the search distribution $p(z|\theta)$ for the SNES algorithm. Here, $\tilde{p}$ is a family of densities on the real numbers, and $\theta = (\theta_1, ..., \theta_d)$ collects the parameters of all these distributions. The pdf is factorized over each dimension i, meaning that the distribution for each variable z_i is independent of the others.

3 Related Work

A recent study that aimed at studying the efficiency of Evolutionary Strategies on continuous control optimization problems has been conducted by [9]. More specifically, the research [9] aimed at analyzing the efficiency of modern neuro-evolutionary strategies for continuous control optimization. The term 'modern' in the context of this study refers to algorithms that compute interrelated dependencies among variations of individuals. Moreover, the evolution strategies that have been used throughout the study [9] included CMA-ES, XNES, and SNES which similarly are the main algorithms being used throughout this research paper. An additional algorithm that has been used in [9] was , namely, OpenAI-ES algorithm. This algorithm involves the usage of an isotropic Gaussian distribution with fixed variance as well as, the fitness of a population to estimate the gradient and update center of distribution through Adam stochastic gradient optimizer. Furthermore, in regards to the analysis of the efficiency of the different reward function of the algorithms in the study, the research used PPO(Proximal Policy Optimization) which uses stochastic actions while operating on a single individual policy.

Among the many experimental setups with various environments, the research focused on 5 main MuJoCo locomotion problems (Swimmer, Hopper, HalfCheetah, Walker2D, Humanoid) which are the most relevant in the context of this paper since it involved examining agent's ability to move as fast as possible in various MuJoCo virtual environments. Additionally, the parameters which Pagliuca et al. [9] used in their research : feed-forward neural network, linear output neurons, hyperbolic tangent function, and Gaussian noise resemble the ones used throughout this paper. Therefore, the reward analysis of the experiments in [9] provide insightful results that have been used to further determine and tailor the algorithms of this paper. The performance of OpenAI-ES algorithm has been noted by [9] to outperform the other participating ES algorithms. More specifically, it's high reward values were dominant in 4 of the MuJoCo environments while the CMA-ES and SNES were two of the algorithms that were approaching the OpenAI behaviour. The high reward values achieved by various agents is said to be enabled due to Open-AI's ability to maximize steps which allow the agent to stay upright. Consequently, in one of the environments (Humanoid), the agent received high reward points by standing still. This implies that the points received for standing upright do not aid in evolving an agent's ability to efficiently walk but rather in optimizing its reward components. Thus, Pagliuca et al. [9], attempted to alter the reward functions which rewarded agents w.r.t speed toward a targeted destination while simultaneously punishing for every agent's joint that reached its limit. As a consequence, [9] noted that their reward function did not work well with agents that do not use all activated joins properly , as is the case with humanoid agent and the ant agent of this paper. Therefore, the 'un-stable' performance between the reward functions implemented in the OpenAI-es algorithm has been considered as a limitation which according to [9] requires further attention and analysis. With this in mind, the other ES strategies of this paper have been chosen to be used and implemented in the current aim of this research which focuses on one MuJoCo environment/agent (Ant) and examining the reward values of the agent w.r.t its locomotion behaviour.

While the extensive description of the work done by Pagliuca et al. [9] above serves as one the most important research in regards to the experiments conducted in this paper, two other studies ([10] and [11]) provide through their work insightful contributions. Through the work of Salimans et al. [10], a novel and efficient alternative to reinforcement learning has been provided. While, the work

of Palmer et al. [11] lays out an approach to escape local optima during the evolution of a neural network controller. In the research conducted by [10], the study of Evolutionary strategies and its application on Black-Box optimization problems has been applied in MuJoCo physics simulator and Atari games. The evolution strategies used in their studies belonged to the NES class where they have highlighted their ability to be scaled by many parallel workers. Thus, [10] noted that ES algorithms require low band-width which implied that each worker only communicates a single scalar when communicating with each other when updating parameters. Salimans et al. [10] novel approach , shared random seeds, is what enabled them to ultimately reduce the bandwidth and communication between workers and their parallelization method. In addition to offering a novel approach to parallelization, [10] emphasized the advantages of ES algorithms through MuJoCo simulation (3D Humanoid) which adopted parallel methods and consequently achieved linear speedup in solving its optimization task.

The work laid in [11] on the other hand focused on a different environment and target. Palmer et al. [11] aimed at escaping local optima while evolving a neural network controller for a simulated bipedal walker. Their work managed to achieve and approach their desired end goal through a technique which involved the construction of linear arrays of sub-populations. In other words, local mating/competition and occasional migration between sub-populations was enabled.

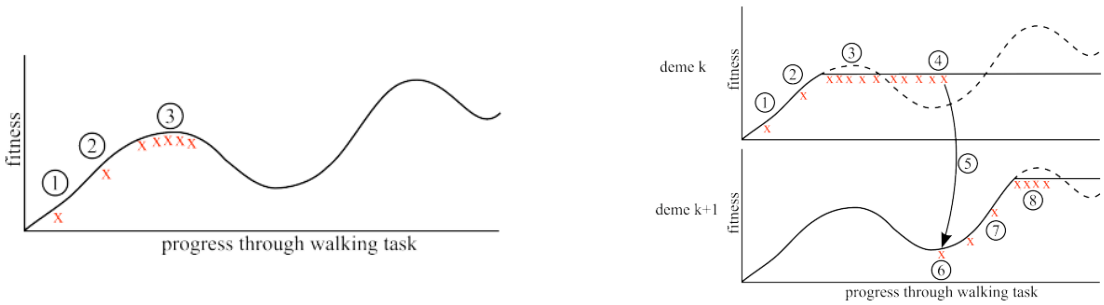


Figure 3.1: One deme vs Multiple demes, (retrieved from : [11])

In figure 3.1 a visual depiction of the main approach of [11] is provided. More specifically, in the figure to the right side, sub-populations migrate in higher sub-populations (deme k+1) where more fitness function is exposed causing the population to move toward higher fitness regions. This is a solution proposed by the research to avoid the scenario of the graph depicted on the left side where a sub-population is trapped in a local optimum. Although, the insights documented by Palmer et al. [11] were shown to be promising, they mentioned a limitation and drawback concerning performance cost. They highlighted that in their multi-sub population method, when keeping sub populations alive in order to potentially mutate and migrate to a more optimal region the performance cost increased linearly per generation. Building upon the insights and findings of the related work mentioned in this chapter, the current research aims to further employ and investigate the usage of ES algorithms in locomotion behaviour and further enhance the optimization process. The following chapters will present an overview of the research hypothesis and approach, which seeks to optimize the performance while mitigating the drawbacks associated with the multi-sub population method and reward functions.

4 Overview

In this chapter, an overview of the research conducted in this thesis is presented. The primary objective of this research is to inspect, implement and optimize the locomotion behaviour of the ant agent of MuJoCo environment through the usage of 3 main ES algorithms. The algorithms in question are : CMA-ES, XNES, and SNES. More specifically, this research aims at finding the most suitable ES algorithm to solve the artificial ant task alongside the most optimal parameter values to enable a robust and dynamic interaction throughout simulations. As mentioned in the preceding chapters, ES algorithms have been widely used in various continuous control and black-box optimization problems. The efficiency of ES algorithms has garnered significant attention, rightfully so, due to the advantages that have been explicitly mentioned and detected in numerous studies such as the ones mentioned in chapter 3 of this paper. The ability of ES algorithms to be easily parallelized, advantageous in long action sequences and delayed rewards, and not calculating gradients are what makes it suitable and adjustable in various problems such as the one involving the optimization of continuous robotic control problems. Therefore, the environment in which the efficiency and limits of ES algorithms will be tested on involve the MuJoCo physics simulator (see [12]). The MuJoCo environment [12] itself involves various agents which include stochastic initial states. From the numerous 3D robots, that MuJoCo facilitates, this paper aims at focusing on *one* of the environments. Namely, the Ant agent (Figure 4.1) which consists of one torso and four legs. The main aim of this environment, is to coordinate the four legs of Ant agent and move in a forward direction by synchronizing its limbs and overall body. The Ant environment, is split up into action space, observation space, rewards, termination state, and starting state [12]. The size of the action space is 8 and accounts for the torques that directly descend and are applied to the hinge joints of the ant. The observation space on the other hand contains positional values of the body parts of the ant as well as its corresponding velocities. The observation space, therefore, contains 28 values which include coordinates,orientation, angles of the body parts. The remaining elements (84) of the observation space account for the contact forces which are used in the center of mass of each link. However, it is important to note that this research disregards contact forces and therefore only focuses on the positional/velocity values of the agent. The reward of Ant agent consist of: Healthy reward, forward reward, control cost, and contact cost. The healthy reward assigns a reward value to the ant agent whenever the ant manages to stay healthy, while the forward reward accounts for the reward given to the ant when it moves forward direction. The control cost on the other hand functions as a penalizing factor that assigns negative rewards for taking actions that are too large while the contact cost assigns negative rewards if external contact force is too large. The starting state of the ant starts at a fix coordinate/position which places the ant in an upright position. Lastly, the termination state involves several scenarios which take into account ant's health and time steps. Ant's state is said to be unhealthy according to [12] when state spaces are no longer finite or the z-coordinate of the torso is not in the closed fixed interval ([0.2,1.0]). Moreover, if 1000 time steps have been reached or exceeded then the episodes duration has been said to be reached and hence reached the terminal state.

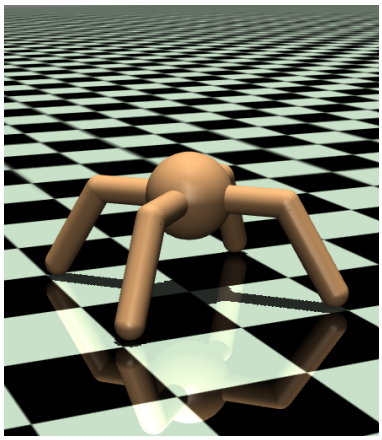


Figure 4.1: Ant Agent of MuJoCo Environment

5 Methodology

The following sections will provide and justify the methodological choices made throughout this research and its experiments. Firstly, the approach of this research will be described and clarified through specific setup configurations. Secondly, the experimental setup of this research will be extensively covered alongside with justifications and specific parameter/design choices. Lastly, the evaluation methods used during the inferring and understanding of results obtained from this research are covered.

5.1 Approach

The approach of this research involves several objectives and methods. Overall, the objective of this research contains several 'variants', implying that it contains a correlational, explanatory and exploratory objective. In other words, the research will aim at discovering and establishing the existence of relationship between Evolutionary algorithms (CMA-ES, XNES, SNES) and locomotion behavior of Ant agent while also clarifying the relationships of these factors and exploring their impacts. Moreover, the study will consist of a combination of qualitative and quantitative analysis. The qualitative analysis consists of interpreting the behaviour of the existing ES algorithms on the Ant agent's movement through visual inspection. This visual inspection is aided and provided through the usage of the Gym package (v0.26.2) (see [13]). The general virtual environment which is a branch of the Gym environment and which will simulate the Ant behaviour during the experimental trials is provided by MuJoCo environment (v 2.2.1)(see [12]). Throughout the research, emphasis was placed on ensuring accurate setup and utilization of the most recent packages due to the potential challenges arising from discrepancies among different packages as stated in their API documentation. The experimentation was conducted on a Windows operating system using the python programming language.

5.2 Experimental Setup

In order to attain the objectives from the section above, the experimental setup was of crucial importance. Aside from the visualization packages needed during the simulation, an important evolutionary algorithm library was used throughout the experimentation, namely, the EvoTorch library (see [[14]]). This library, has assisted in the implementation phase by providing easy-to-use built-in functions and classes that contained scalable and flexible parameter/optimization adjustments. Before proceeding to the implementation and algorithm structure that was used to conduct experiments, it is important to lay-out the global variables that were used throughout the algorithms and their value in the following table.

Global Variables	
environment	'Ant-v4'
maxEval	23000
maxFitness	-1000
Visualize	0
input	27
hidden	20
output	8
bias	1
standard_dev	0.01
whi_no	hid * (inp + 1)
connections	(inp + bias) * hid + (hid + bias) * out
initial_bounds	(-0.00001, 0.00001)

As can be seen in the table above, there are several variables that act as global and constant variables. These values have been chosen according to several factors. The environment variable has been chosen to be the most up to date ant environment which is suitable for the latest Gym and MuJoCo libraries. The *maxEval* and *maxFitness* serve as termination variables. More specifically, they both account for the two states which are checked in the main loop of the algorithm. The *maxEval* value represents the maximum number of evaluations allowed to be conducted in a single run of a single environment in which the Ant agent resides. The value of 23000 represents the total number of evaluations which can be reached by 1000 iterations/generations. This implies that the default value of evaluation computed by the Evotorch algorithm class is set to 23 per generation. The *maxFitness* value on the other hand is set to a negative value of (-1000) which is intentionally negative due to the objective of the algorithms being a minimization problem. Therefore, the algorithm's efficiency is determined and evaluated on the fitness value per generation. While the negative sign of fitness value is justified from the optimization problem of the algorithms , the arbitrary number (1000) is chosen due to ant's agent movement patterns. It has been seen through several trials (see Ch.6), that the ant walked well over the coordinate plane when reaching the fitness value. The visualization variable is by default set to 0 as it was necessary to only infer visually once the best individual was found. The *initial_ bounds* variable stores the initial interval with the values of a new solution, and is set within a small constrained range which will be later on (in ch.6) be documented and reasoned. The input, hidden, and output variables all account and represent the artificial feed-forward neural network architecture that is used throughout the simulation. *Input* represents the amount of input nodes of the neural network which is intentionally set to 27. This value is crucial for the simulation to work, as it represents the observation space of the Ant agent (refer back to Ch.4 for further details). Similarly, *output* variable, represents the output nodes of the neural network and is also limited to the value of 8, due to its representation of the ant's action space. The *hidden* layer of the neural network, is not directly correlated to the ant and therefore was chosen to be an arbitrary value that would not compact the neural network. For a more detailed view of the artificial neural network in a graph format, please refer to the Appendix (Fig 8.1). *Connections* variable accounts for the generation length as well as the solution length. In more detail, it represents the total number of connection in the neural network and also serves as the solution length of the vector generated by the algorithm. Aside from using input, output,

hidden nodes it also uses a *bias* term that is set to 1. Lastly, the *whi_no* variable represents the total number of weights connecting the input layer to the hidden layer.

The extensive description and reasoning of the variables above all come into play in the main algorithm and pipeline used to evaluate and implement the ES algorithms. In the following pseudocode below, the algorithm structure of the large pipeline of the evolution is provided.

Algorithm 1 Large Pipeline Ant Evolution

Require:
Import necessary libraries and modules
Define global variables of Artificial Neueral Network, termination states etc.
procedure EVOLUTION
Class Evolution
Method __*init*__algorithm_name, density, height
Input: algorithm name, density, height
Initialize algorithm, density, height, and xmlname
Method walker(agent)
Input: agent
Output: Resulting reward (-R)
Initialize environment, observation, termination state, agent, input weight, hidden weight, step, Reward
While not done
Render environment if VISUALIZE is 1
Calculate input layer, hidden layer, and output layer
Update obs, reward, terminated, done
Method running_algo()
Initialize problem, searcher, logger, pandas_logger, fit, nEval
While fit $>$ maxFitness and nEval $<$ maxEval
Step the searcher
Update best, fit, nEval if necessary
Save the logger data to files
Output best, fit, number of Evalutations
Initialize density_list, maxheight_list, population, environment, fitness, number of Evaluations, counter
Iterate over density_list
Iterate over maxheight_list
Print combination number and values
Initialize algorithm with SNES/XNES/CMAES, density, and maxheight
Run the evolutionary algorithm and store results
Increment the counter
Save population, fitness, and evaluation counts to files

The pseudocode above is considered to be a large pipeline of the ant's evolution due to the many density and height values being iterated when initializing the algorithm. The density list, consists of different number of boxes (obstacles) that are placed in the virtual environment and generate an array with 10 evenly spaced values between 0 and 900 (e.g. 0,100,200..900). The height list on the other hand consists of different height values which the box obstacles contain and also generate

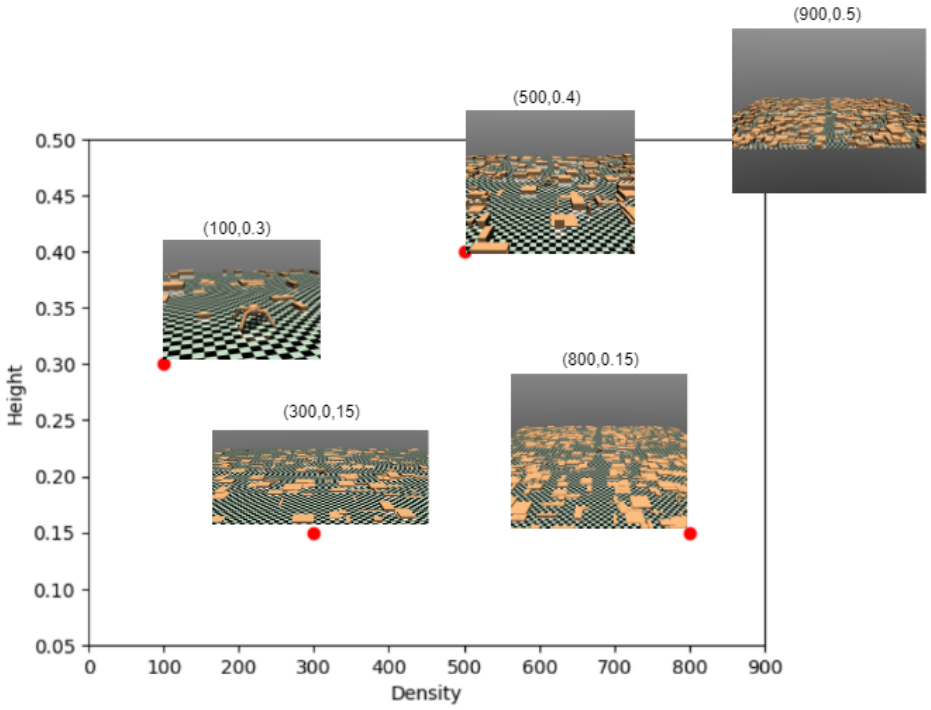


Figure 5.1: Various Simulated Environments of Ant Agent

an array with 10 evenly spaced values between 0.05 and 0.5. Thus, this large pipeline aims at training and evolving ant's movement through various environments, 100 to be precise, where the ant's walking patterns and body positional values are evaluated and used to enable its walking while trying to minimize its fitness value when evolving through different generations. The plot provided in Fig 5.1 depicts several of the virtual environments where the ant will spawn and walk over the terrain.

5.3 Evaluation

As seen in Fig 5.1 and Algorithm 1, the numerous amount of variables and factors impacting the ant's behaviour acquire evaluation phase to be as detailed as possible. In the pseudocode itself (Algorithm 1), the training of the ant is done through the 'walker' method, where an agent is evaluated and simulated through various XML files. These XML files simulate and place the different density and height values of the obtained arrays from the algorithm. In the official MuJoCo XML reference documentation [15], an in-depth explanation of the XML elements/sections is provided. These XML files divide the desired simulated environment in several sections which account for: ant's (or other agent's) initial position, joint & spatial tendon elements within the body, obstacle positions alongside its specific configurations (e.g. size, type). From the many attached sections found present in the XML configurations of the simulated environments, the collision properties (density, friction, margin) of the 'geom' tag (which initiates the box obstacles) has been of great importance during the evolution of the ant. These collision properties alter the state of the boxes in various ways. More specifically , the 'density' attribute determines the mass of the object while the 'friction' property , which contains 3 parameters (sliding, torsional, rolling), controls the friction between the object and other surfaces. Lastly, the 'margin' property holds the distance threshold which contacts/collisions are detected. During the evolution of the ant, a tweaking of these parameters mentioned above has

been crucial in determining the overall interaction and handling between the ant and boxes.

From the 3 default parameters of the 'geom' tag mentioned above, it has been observed through several trials that the 'margin' value helps the ant avoid the boxes more efficiently. The efficiency of avoidance is considered to be the ant's limb movement when passing near an obstacle and its overall walking pattern throughout the coordinate plane. The default values provided in the XML files of the collision properties were : density (5.0), friction (1 0.5 0.5),and margin (0.01). When adjusting the margin value to a value of (1.0), the most apparent effects were seen during the performance of the ant as it seemed to correctly handle the obstacles by walking on top of them or around them. When adjusting the friction values or the density of boxes, through visual inspection, it was evident (further details in Ch.6) that no effect was comparable to the one of the margin value. Therefore, the margin default value has been set to '1.0' in 5 environments of the evolution phase. In more detail, the environments from box quantity (500-900) alongside height (0.4 , 0.5) contained the adjusted margin value due to the environmental setup containing a dense amount of boxes and height which are challenging for the ant to properly interact and walk. The preceding environmental values contained a more sparse amount of boxes which enabled the ant to overcome the obstacles without the adjustments of the XML parameters. As mentioned in section 5.1, this research will also contain quantitative data which has been enabled through the the usage of 'PandasLogger' and 'StdOutLogger' packages. These two packages will store the performance of the algorithm for every run and configuration. In return, numerical data and statistics about each iteration of the algorithm is displayed and stored in a csv file which will then be used to visualize graphs. This in return allowed this research to properly compare the generated fitness values of the algorithms and the best individual per algorithm/configuration.

The following algorithm (Algorithm 2) is responsible for testing and visualizing the performance of the best individual generated from the several ES algorithms and viewing the interaction of the ant in the generated virtual environments enabled by the XML files.

Algorithm 2 ES Visualization

```
Import necessary libraries and modules
Define environment as "Ant-v4"
Define maxEval as 23000
Define maxFitness
Define VISUALIZE as 0
Define input, hidden, output dimensions
Calculate number of connections
procedure TEST_ANT(agent, visu)
    Set seed and load environment
    Initialize variables
    Extract weights from agent array
    Reshape weights
    while not done do
        Render environment if visu = 1
        Prepare input
        Calculate hidden layer activation
        Calculate output layer activation
        Take step in environment
        Update total reward and check termination
    Close environment
    return negative total reward
Load best performing agent from CSV file
Call TEST_ANT function with loaded agent and visu = 1
```

6 Results

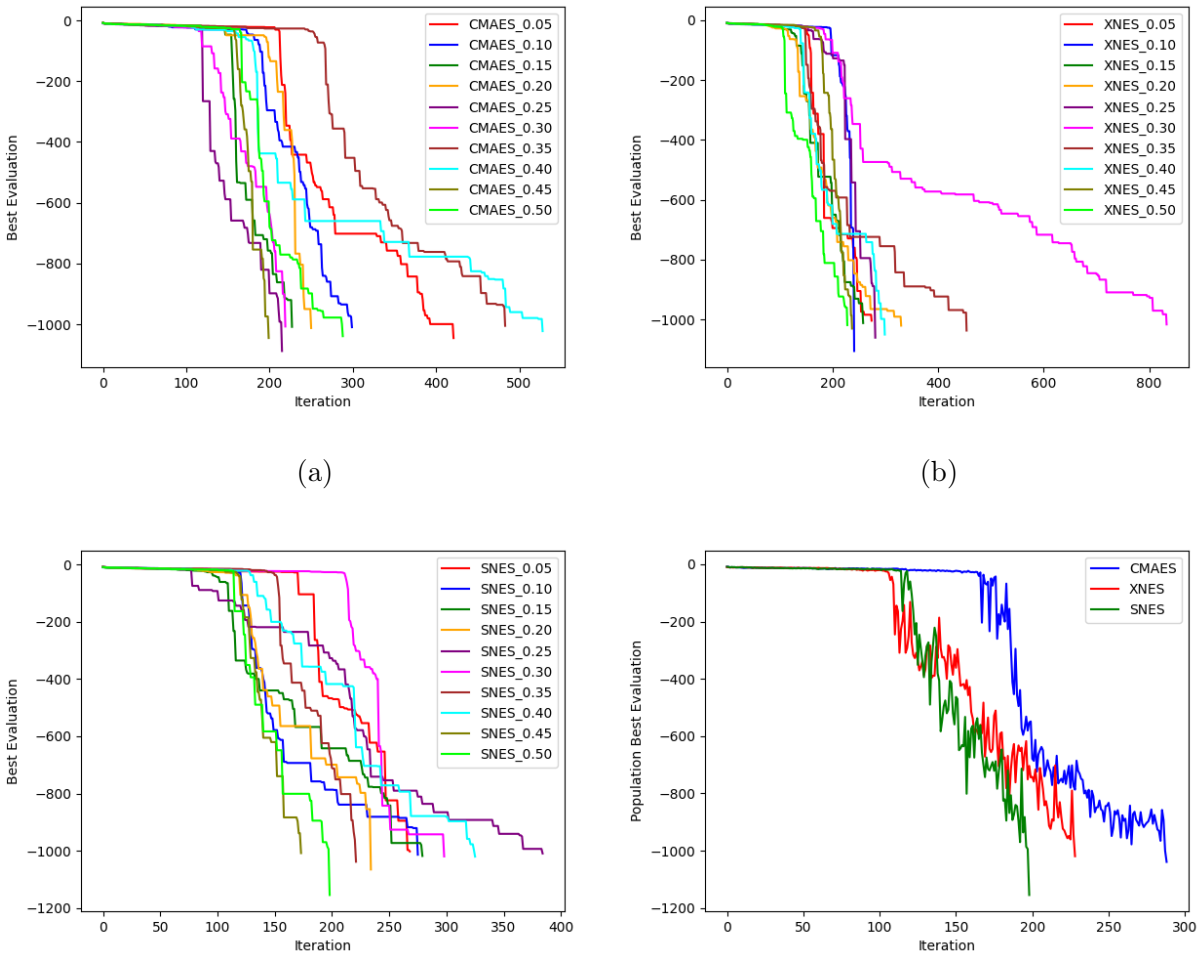
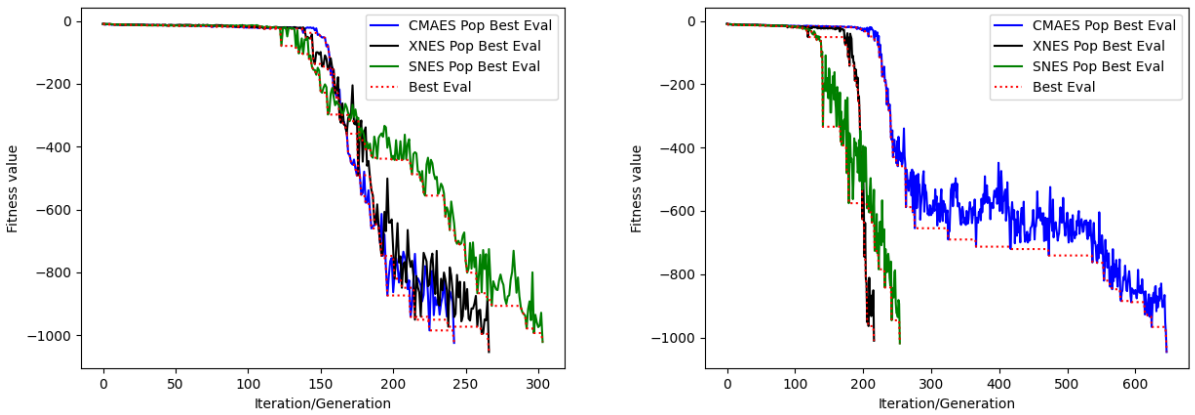


Figure 6.5: Convergence of Fitness Value in Different Heights of 100 Boxes

As mentioned in section 5.2, of the methodology chapter, the large pipeline consisted of 100 environments & configurations. The initial environmental setup consisted of a flat surface (0) boxes and progressed all the way through 900 boxes in an environment. In order to understand and observe the impact of the ES algorithms during the interaction and locomotion behaviour of the ant agent in these various virtual environments, numerical data was extracted from every evolution and displayed graphically. In figure 6.5, attached above, the environment of 100 boxes alongside its 10 different heights are displayed per ES algorithm. In figure 6.5.a, the CMA-ES algorithm's performance is displayed by providing the best evaluation through generations. The best evaluation is considered to be the best fitness value that is found through generations and iterations. Hence, when the best individual's fitness value reached a value lower than or equal to -1000, the algorithm's run is stopped. During the simulation using the CMA-ES algorithm, 7 of the environments with

100 boxes took 200-300 generations to reach the optimal fitness value that would enable the ant to pass through the simulation while 3 of its environments where running for 400-500 iterations. Overall, in the lower settings (lower heights) of the environment a faster convergence to the global optimum is achieved as expected due to the 'easy' environment consisting of sparse and low amount of boxes. In figure 6.5.b, the performance of the XNES algorithm is provided. In contrast to CMA-ES algorithm the convergence to the global optimum is achieved faster and is evidently seen by the fast/sharp drops of the fitness values as generations pass in the majority of the environments. Moreover, the best individual line graphs/values are more dense and closely attached to each other, regardless of the different box heights implying that the ant agent managed to maintain a more rapid evolution progress and traverse through the obstacles while not acquiring that many generations to pass. An outlier, although, is found in one of the environment which contained a medium height of the obstacles which implies that the algorithm got stuck in several local optimums while exploring the search space. Thus, indicating the performance of XNES is less steady compared to CMA-ES one. In the third and final algorithm simulation(SNES), the ant's evolutionary progress is provided in fig.6.5.c in which the overall convergence to the global optimum is achieved throughout all of the environments in a smaller range of generations ($\sim$ 200-400). It is therefore apparent that the SNES's performance outperforms the other two algorithms in terms of achieving the optimal fitness value while also evolving at a faster pace. To further prove the outperformance of SNES, the last picture (fig.6.5.d) shows the best individual per population ,as opposed to the global fitness value, in the highest setting of the environment (100 boxes with 0.5 height). The performance of SNES portrays its ability to evolve and converge to global optimum value much more efficiently as opposed to CMA-ES and XNES. Lastly, a general pattern to be found in all of the plots in 6.5 is the general evolution of the algorithms, considering the fact that as the ant progressed through the different environmental configurations it achieved in the hardest setting (0.5 height value) a global optimal fitness value in lower amount of generations.

While the thorough analysis of the environment with 100 obstacles above provided an insight to the performances and differences between ES algorithms, a more dense environment consisting higher number of boxes is necessary to further explore and evaluate the algorithms capabilities.



(a) (b)

Figure 6.8: Environment with 500 Boxes vs 900 Boxes

Therefore, Figure 6.8 provides the performances of the algorithms in two environments, namely,

500 boxes (a) and 900 boxes (b). These environments include more dense amount of boxes that are stacked above one another, aiming to test the joint limits and body movement of the ant agent while moving throughout the coordinate plane. Since the environment of 100 obstacles/boxes was considered relatively 'easy' to test ant's capabilities, the environment of 500 boxes is set to be the middle-ground while the last environment (900 boxes) is set to be the most adverse environment with randomly stacked and positioned boxes all over the coordinate plane. With this in mind, fig.6.8.a portrays the performance of the ES algorithms in an environment with 500 boxes and a 0.15 height. A contrasting convergence to global optimum is visible, due to SNES's performance taking the most amount of convergence and generations. The population best evaluation found for each of the algorithm is the best value that was found in a specific generation/iteration while the best evaluation denoted in red dot was the global best individual found throughout the simulation. Therefore, it is evident that the performance of the ES algorithms vary in different environmental configurations as can be seen my CMA-ES's dominance in Fig.6.8.a. However, in the plot next to it (Fig.6.8.b) the highest amount of boxes in a simulation displays different results. This plot, simulated an environment consisting the most dense amount of boxes (900) with a height value of 0.15. The convergence of CMA-ES algorithm differs with the one from the adjacent plot showing that it took more than 600 generations to reach an optimum value. The two other algorithms on the other hand remained steady regardless of the comparable environmental changes and managed to converge faster with lower amounts of generations. When visually simulating the best individuals retrieved from the algorithms, SNES and XNES managed to generate a walk over the boxes at a faster pace while sacrificing exploration. CMA-ES, on the contrary, took more steps and exploratory behaviour walking in a diagonal pathway and hence taking more generations to walk over the coordinate plane.

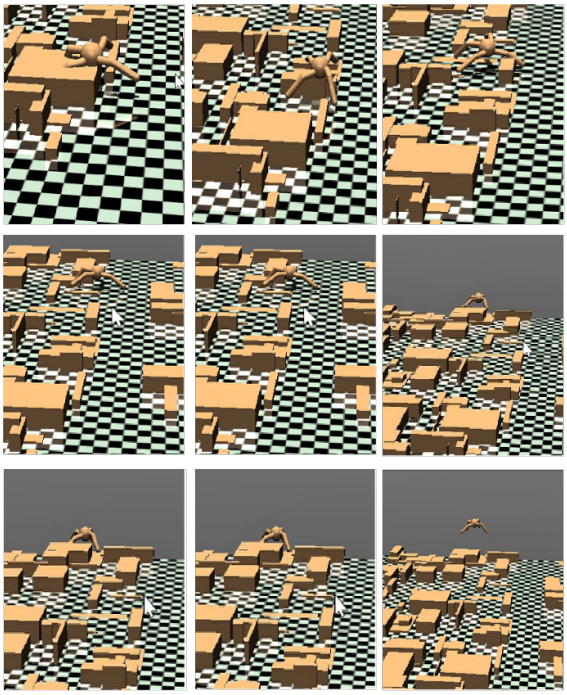


Figure 6.9: An environmental simulation of 700 boxes, 0.4 Height, default margin value

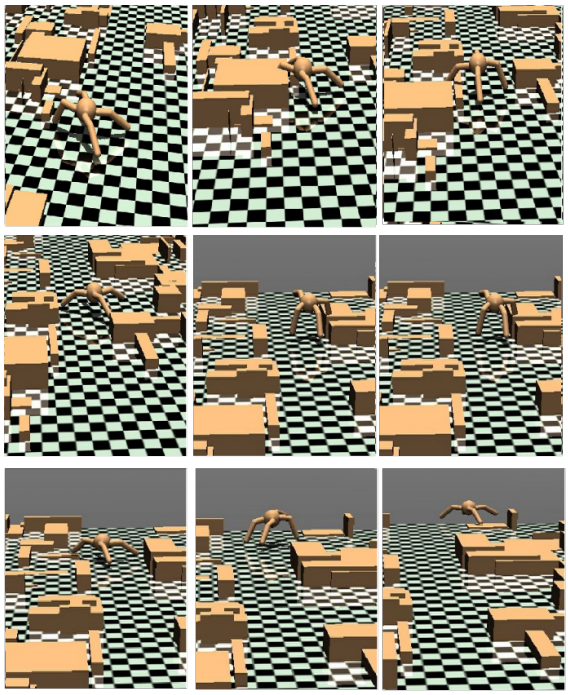


Figure 6.10: An environmental simulation of 700 boxes, 0.4 Height, altered margin value

Whilst the various plots above provided quantitative insights on the differences of performances between ES algorithms, a specific configuration adjustment was performed in the XML files of some of the simulated environments (see Sec.5.3 for further details) which provide further qualitative insight. This configuration adjustment concerned the collision properties which aided the ant when detecting the obstacles it encountered. When adjusting the default margin value from value 0.01 to 0.1, a noticeable difference was seen during the simulation run. In the two grids attached above (Fig. 6.9 & 6.10), a more explicit comparison is provided when simulating an ant agent in an environment consisting 700 boxes with a 0.4 height using the CMA-ES algorithm. As previously mentioned, the general body movement of the ant agent is as equally important as its convergence to a global optimum fitness value. Therefore, when simulating the ant agent with default configurations of the XML files, the collision detection of the agent was flawed as it did not detect the obstacles on time. Consequently, it impacted the algorithm's performance as it forced the algorithm to continue exploring the search space and acquiring more steps in the direction of more densely packed boxes. When adjusting the collision detection factor to a higher value a difference in performance was present. Since the difference at which collisions were detected was raised, the ant was able to use this information to adjust its step more accurately and avoid the boxes accordingly. Fig 6.9, displays through a 3×3 grid the specific ant body movement with a default/low collision value. Starting from the top left corner and moving in the right direction, it is visibly clear that as the ant walks over the boxes, its body merges inside the boxes and takes all of the steps in the direction with the most dense amount of boxes. This behaviour resembles one which cannot detect the boxes and collisions well. In the adjacent grid (Fig. 6.10) a contrasting behaviour of the ant interaction is provided. In the attached screenshots, the ant moves away from the boxes as soon as it detects them and adjusts in direction and position toward spaces that do not contain that many boxes. This in return, enables the ant to avoid obstacles more efficiently and take a shorter path with less collisions.

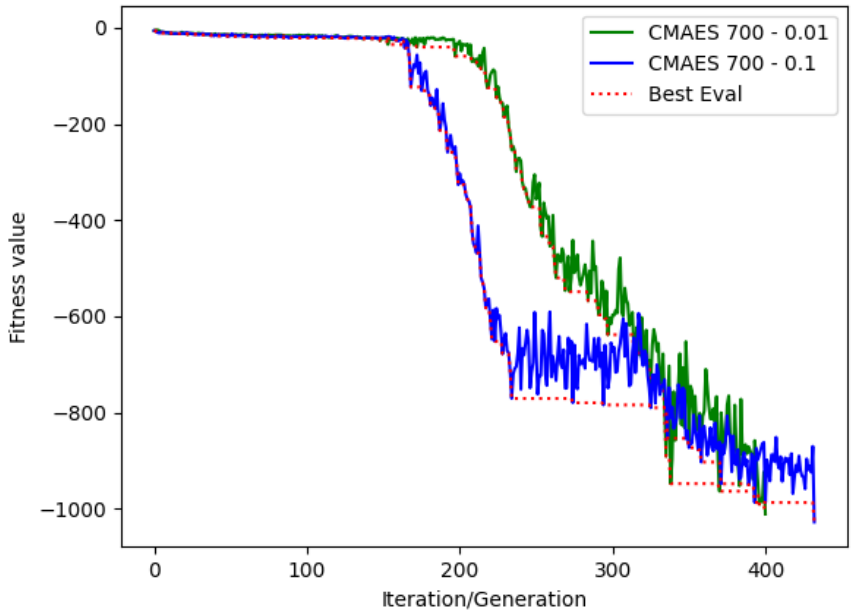
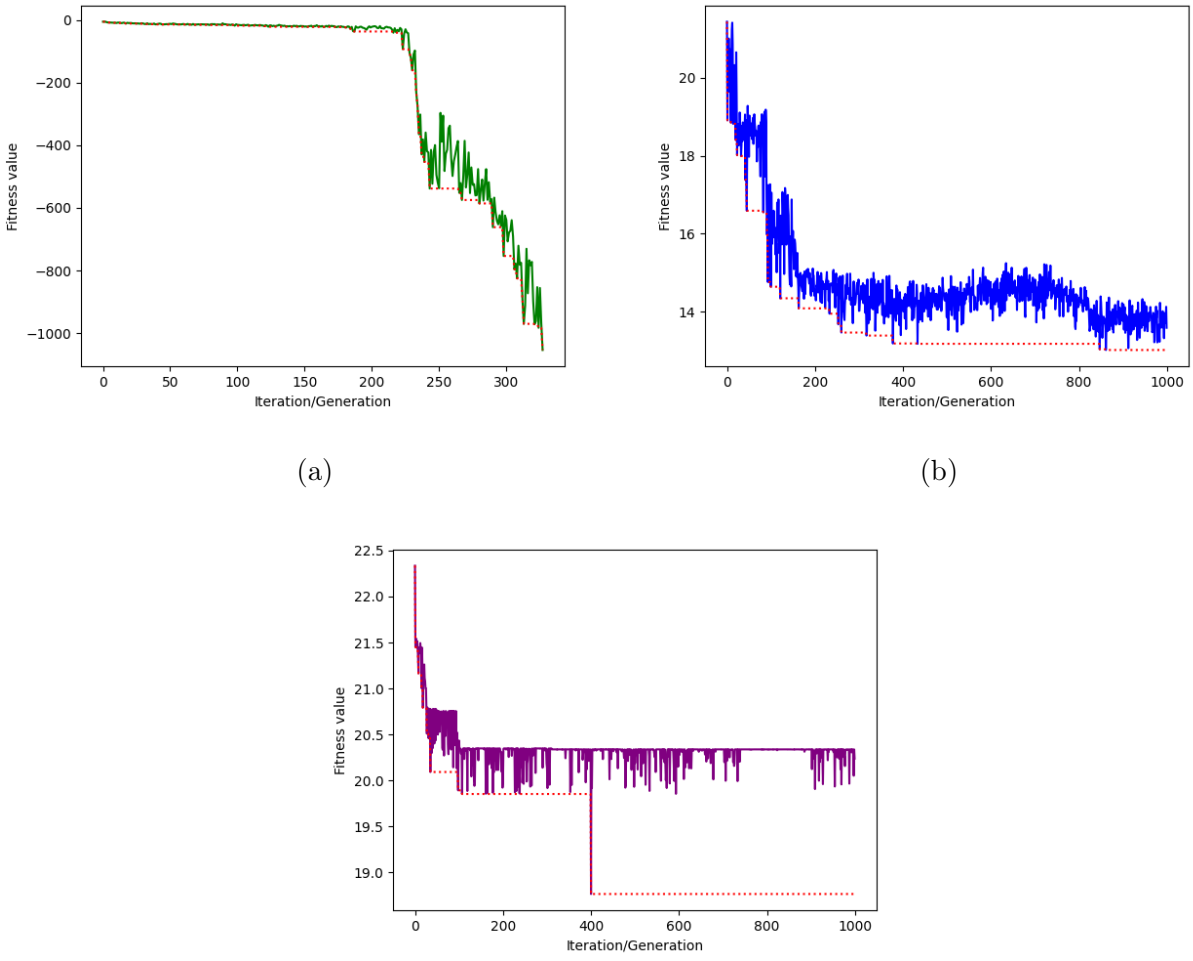


Figure 6.11: CMA-ES Performance with Different margin values

In the plot above (Fig. 6.11) the performance of the the CMA-ES in terms of the population best fitness value and global best fitness values are also displayed to further provide a clearer view on the impact of the adjustment of collision properties. Specifically, when the margin/collision detection property was set to 0.1 the algorithm converged faster to lower fitness values and was able to achieve its minimization goal. The default margin value, contrary to the adjusted value, included more exploration and steps to reach global optimum fitness value. Thus, the qualitative data (Grid) and this quantitative plot (Fig. 6.11) show the impact of the XML environmental settings in regards to the ant behaviour and algorithm's performance. Lastly, one more impacting factor that has been established throughout the simulation are the boundaries that are used to initiate the solution vector for the ant agent. The bounds are set to a considerable low value/range (-0.00001, 0.00001) which has been proven to allow the agent to reach the global minimum (Fig. 6.15.a) while increasing the range to (-1,1) caused the ant to fall in a local optimum (6.15.b). Similarly, fig.6.15.c shows if the range is set between (-5,5). The following graphs have been retrieved from a simulation of XNES algorithm on an environment consisting 500 boxes with 0.4 height.



(a) (b)

(c)

Figure 6.15: Convergence of XNES algorithm with different initial bounds

7 Discussion

Through the various plots provided in Chapter 6 several insights can be extracted from this research. Most importantly, a general key take-away from this research is the impact of parameter choices throughout the simulations. From the construction of the artificial neural network up until the simulated environment generated by XML files, the values which are fed to the ES algorithm and specific evolution methods matter greatly when optimizing the locomotion behaviour of an agent. In regards to the comparison and differences between the ES algorithm performances, all of the algorithms have managed to successfully replicate a steady walking movement for the ant agent. However, each of the algorithms contained specific approaches which were visibly or quantitatively apparent upon their usage. In terms of a more stable algorithm performance during the vast amount of environmental configurations, the performance of CMA-ES stood out the most. This algorithm, took a comparably longer time to converge to a global optimum value. In spite of this, when extracting the best individual and visually simulating it, the ant movement and obstacle handling was the most optimal one. The movement of the torso and limbs were dynamically changing as the number of obstacles and their height progressively increased. As observed in Ch.2.1.1, the usage of a covariance matrix in updating the step size and evolution path can be credited for the performance of CMA-ES mentioned above. Although the performance of CMA-ES algorithm proved to favor a steady evolution path, SNES and XNES were the two contrasting algorithms that replicated more fast-pace, dynamic evolution. When wanting to imitate a costly computational behaviour, the long evolutionary pathway generated from CMA-ES algorithm can increase the time complexity & computational costs of the simulation. Thus, the fast and sharp convergences to a global optimum in lower amounts of generations is supported by SNES and XNES. Even though both of these algorithms are part of the NES family, differences between these algorithms were visually depicted in Ch.6, where SNES algorithm frequently outperformed XNES in terms of convergence to optimal values. This behaviour can again be traced back to the overview of the algorithms provided in Ch.2 , where the SNES algorithm approach has been distinguished through its separation method of search distributions when adapting its parameters.

7.1 Limitations

As indicated in the section above, parameter and design choices throughout this research have had a significant impact on the evolutionary process. Thus, hyperparamter sensitivity is one major drawback that emerges as a by-product of this research. Just by altering the initial bounds responsible for the initial solution vectors, the impact was visibly clear (see fig.6.15) and impacting towards fitness value convergence. Moreover, the default parameters and settings found inside the XML files, which were responsible for the training phase and final simulation of the environment were shown to impact the body movement of the ant agent. Although the adjustment of the collision properties were to a certain extent helpful in replicating a more realistic interaction between the limbs of an ant when passing over the boxes, the boxes themselves did consist a solid state. Implying that when the ant walked over them, the limbs could still sometimes get inside the boxes. This is partially due to the lack of realistic sensory values from the ant agent and the default values found inside the XML files. When rendering the environment, besides having access in the camera view and pausing or playing the simulation, no other additional feature was enabled to the user to make the rendering more interactive. Therefore, the simulated environments with various amount of boxes and height values alongside their random coordinates made the visual inspection more difficult.

7.2 Future Work & Conclusion

Having listed the limitations and insights of this research throughout the former sections and chapters of this paper, this section includes the necessary future research based on the identified gaps and limitations as well as the conclusions that are able to be drawn from the conducted research.

One of the most important area which presents potential avenue for future research is the incorporation of the parallelization method during the implementation of ES algorithms. In a previous section (sec.5.2) a visual plot (fig.5.1) depicting the simulated environments which the ant agent went through has been provided. The plot showed the amount of configurations and evolutionary processes which the ant agent went throughout this research. Hence, showing the amount of computation required by the algorithms and the individual evolutionary processes occurring in various simulations. The research conducted by Greenwood et al.[16], on the other hand, proposes a parallelization technique to improve the computation time when running the ES algorithms. They implement this technique through a multi- processor system. More specifically, they let each processor execute for an arbitrary number of generations and a local subset of individuals. When the global fittest individual has been found, it is then broadcasted to each processor. The main reason why this research [16] proceeded to utilize the parallelization method is to prevent a premature convergence and an increasingly high computation time. A similar research [10] has also advocated in favor of the parallelization technique. This research focused on applying the parallelization method in a similar environment (MuJoCo) through the synchronization of random seeds between workers as well as perturbing subsets of parameters.When applying their novel approach in the MuJoCo environment, reduction in computational time and successful convergence in optimal values were present. Therefore, this technique could potentially be further examined in future studies and compared with the values generated from this paper. Moreover, the further exploration and implications of the XML file parameters should be carefully studied in order to truly grasp and achieve the most 'realistic' simulation and interaction between the agent and obstacles.

In conclusion, this thesis has investigated and analyzed the effectiveness of various evolutionary strategies in optimizing the locomotion behaviour of an artificial ant. In more detail, this thesis used three main ES algorithms (CMA-ES, XNES, SNES) to optimize the ant's movement and interaction in 100 virtual environments consisting obstacles of the box shape and various complementary height values. Throughout the usage of these ES algorithms, a successful walking behaviour has been replicated in all of the environments while also showing the impact of various parameter choices and the differences of performance between the algorithms. While all of the algorithms managed to achieve the main objective of this thesis, further research and improvements are advised to be conducted in future studies. The main fields of consideration in future studies are the utilization of parallelization technique in the evolutionary process, hyperparameter sensitivity and more optimal configuration of the simulated environments.

References

[1] Kenneth De Jong, David Fogel, and Hans-Paul Schwefel. *A history of evolutionary computation*, pages A2.3:1–12. 01 1997.

[2] Melanie Mitchell and Charles E. Taylor. **Evolutionary Computation: An Overview**. *Annual Review of Ecology and Systematics*, **30**(1):593–616, 1999. http://dx.doi.org/10.1146/annurev.ecolsys.30.1.593 doi:10.1146/annurev.ecolsys.30.1.593.

[3] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2018. Accessed: May, 2023. http://incompleteideas.net/book/RLbook2020trimmed.pdf[link].

[4] Lilian Weng. **Evolution Strategies**. *lilianweng.github.io*, 2019. Accessed: June, 2023. https://lilianweng.github.io/posts/2019-09-05-evolution-strategies/[link].

[5] Rohan Tangri. **Evolutionary Strategy: A Theoretical Implementation Guide**. *Towards Data Science*, May 2021. Accessed: June, 2023. https://towardsdatascience.com/evolutionary-strategy-a-theoretical-implementation-guide-9176217e7ed8[link].

[6] Nikolaus Hansen. **The CMA Evolution Strategy: A Tutorial**. *CoRR*, **abs/1604.00772**, 2016. Accessed June, 2023. http://arxiv.org/abs/1604.00772[link].

[7] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, and Jürgen Schmidhuber. **Natural evolution strategies**. *arXiv preprint arXiv:1106.4487*, 2011. Accessed June, 2023. https://arxiv.org/abs/1106.4487[link].

[8] Tobias Glasmachers, Tom Schaul, Sun Yi, Daan Wierstra, and Jürgen Schmidhuber. **Exponential Natural Evolution Strategies**. *Proceedings of the 12th Annual Genetic and Evolutionary Computation Conference, GECCO '10*, 07 2010. http://dx.doi.org/10.1145/1830483.1830557 doi:10.1145/1830483.1830557. Accessed June, 2023.

[9] Paolo Pagliuca, Nicola Milano, and Stefano Nolfi. **Efficacy of Modern Neuro-Evolutionary Strategies for Continuous Control Optimization**. *Frontiers in Robotics and AI*, **7**, 2020. http://dx.doi.org/10.3389/frobt.2020.00098 doi:10.3389/frobt.2020.00098. Accessed April, 2023.

[10] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. **Evolution strategies as a scalable alternative to reinforcement learning**. *arXiv preprint arXiv:1703.03864*, 2017. Accessed May, 2023. https://arxiv.org/abs/1703.03864[link].

[11] Michael Palmer and Daniel Miller. **An evolved neural controller for bipedal walking with dynamic balance**. pages 2119–2124, http://dx.doi.org/10.1145/1570256.1570287 doi:10.1145/1570256.1570287, 07 2009. Accessed June, 2023.

[12] **Gym Library - MuJoCo Ant Environment**. https://www.gymlibrary.dev/environments/mujoco/ant/. Accessed: June, 2023.

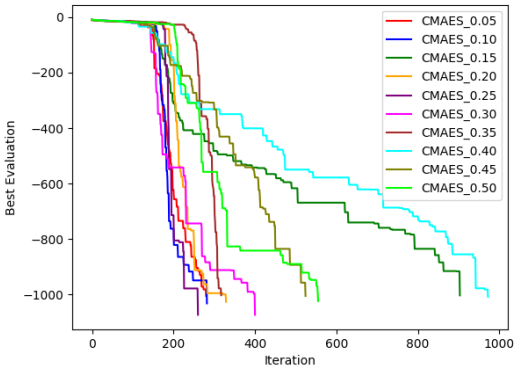
[13] **Gym Documentation**. https://www.gymlibrary.dev. Accessed: June, 2023.

[14] **EvoTorch Documentation**. https://docs.evotorch.ai/v0.4.1/. Accessed: April, 2023.

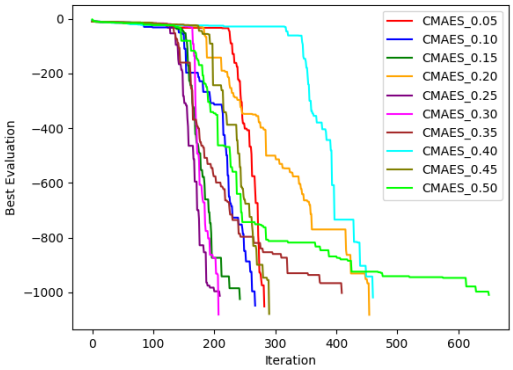
[15] **MuJoCo XML Reference**. https://mujoco.readthedocs.io/en/stable/XMLreference.html#. Accessed: June, 2023.

[16] Garrison Greenwood, Sanjay Ahire, Ajay Gupta, and R. Munnangi. **Parallel Implementations of Evolutionary Strategies**. 06 1996. Accessed: June, 2023.

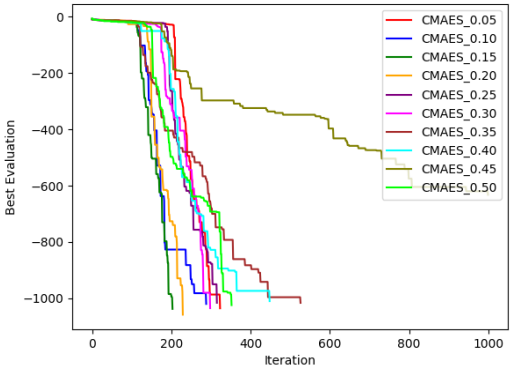
Appendix



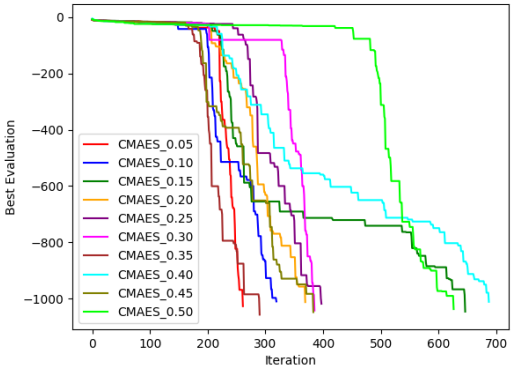
Environment : 200 Boxes



Environment : 500 Boxes

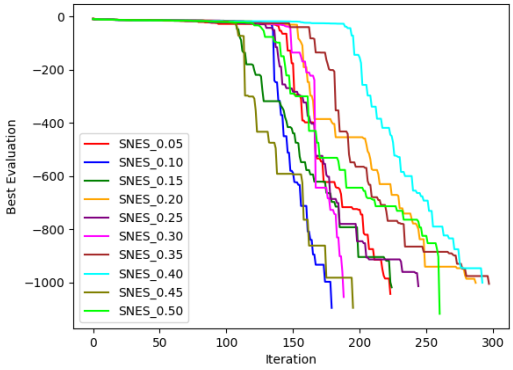


Environment : 600 Boxes

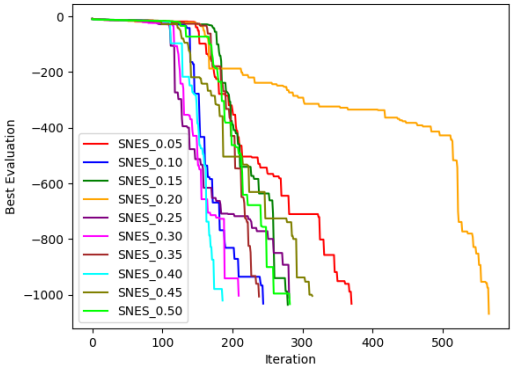


Environment : 900 Boxes

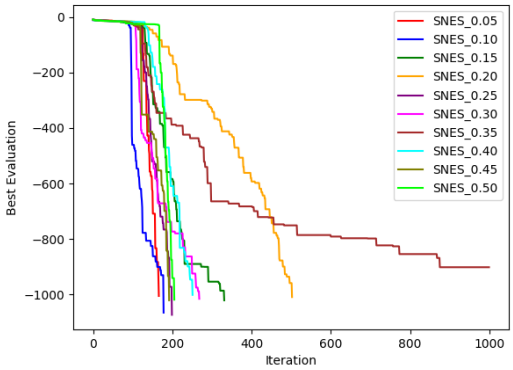
Figure 7.5: Convergence of Fitness Value in Different Heights of various Box Obstacles from CMAES algorithm



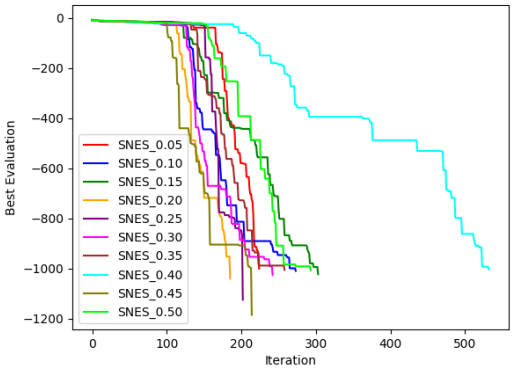
Environment : 200 Boxes



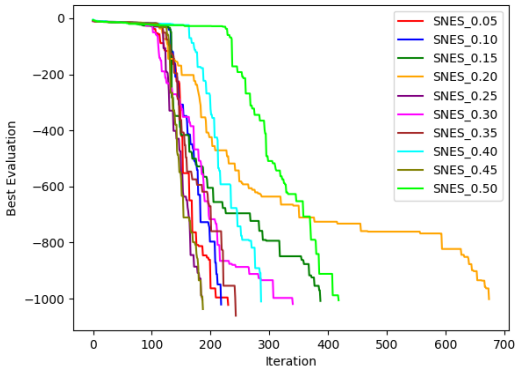
Environment : 300 Boxes



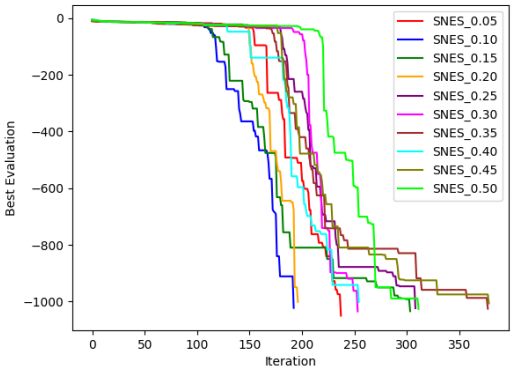
Environment : 400 Boxes



Environment : 500 Boxes

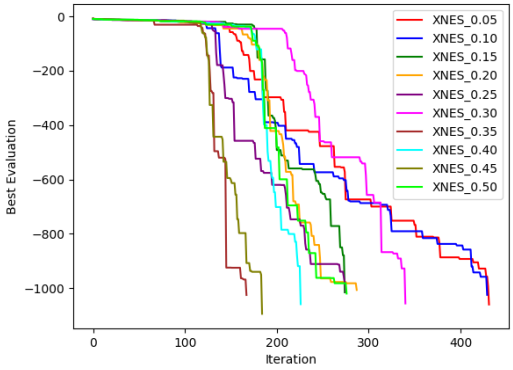


Environment : 600 Boxes

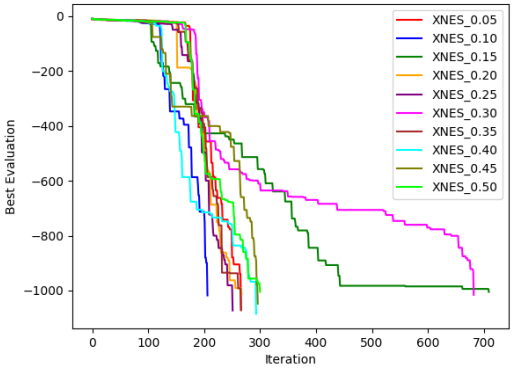


Environment : 700 Boxes

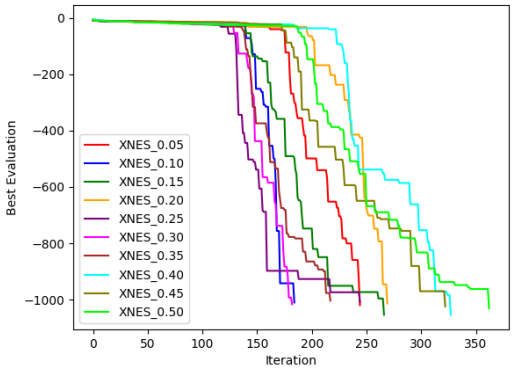
Figure 7.12: Convergence of Fitness Value in Different Heights of various Box Obstacles from SNES Algorithm



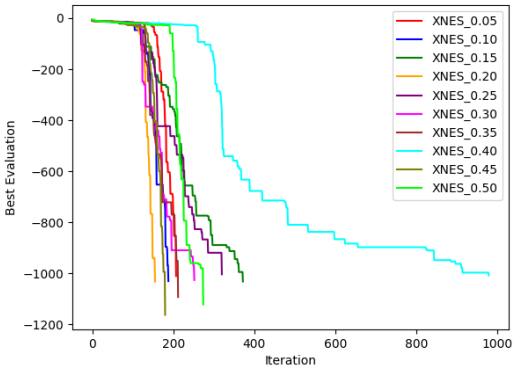
Environment : 200 Boxes



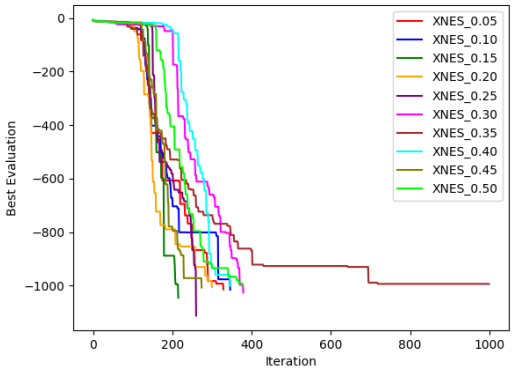
Environment : 300 Boxes



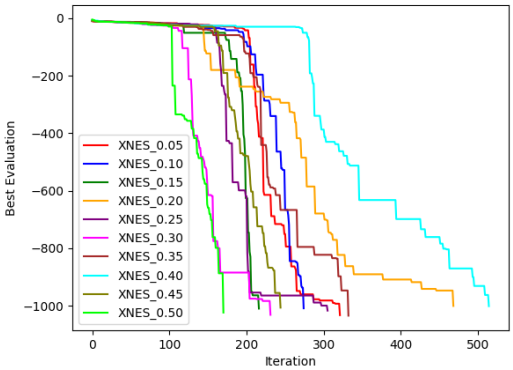
Environment : 500 Boxes



Environment : 600 Boxes



Environment : 700 Boxes



Environment : 900 Boxes

Figure 7.19: Convergence of Fitness Value in Different Heights of various Box Obstacles from XNES Algorithm

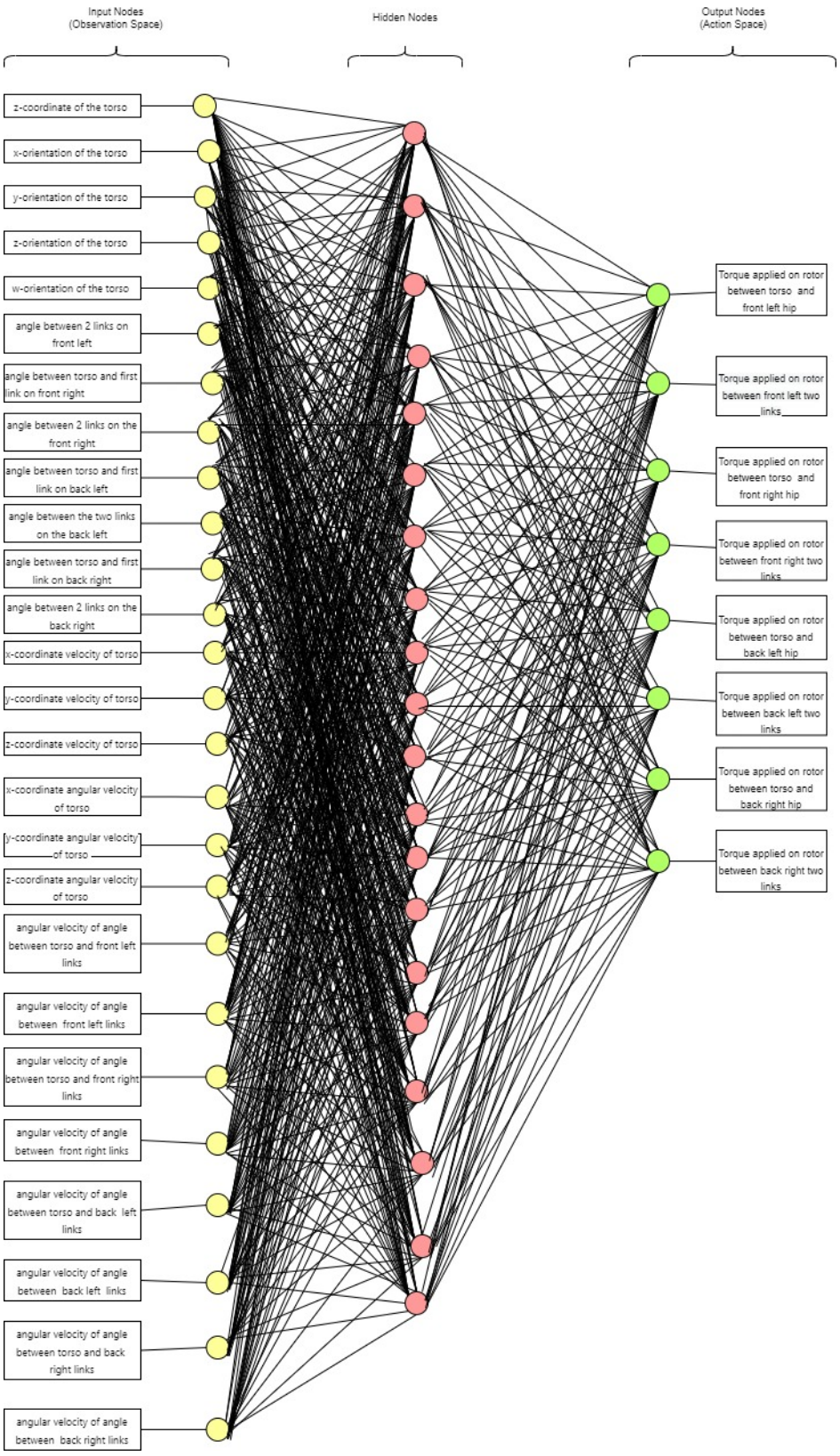


Figure 7.20: Artificial Neural Network Architecture