Chapter 3

The Virtual World

This chapter outlines the virtual world used for conducting the experiments to be presented in this thesis. It is divided into five main sections. First, we briefly discuss the physics engine used to simulate the creature and its environment. Next, we explain in detail the setup of the creature's morphology. Then we describe the basic workings of an artificial neural network (ANN) that act as the creature's locomotion controller, as well as evolutionary methods of learning for ANNs. The next section then explains the genotype used to represent the artificial creature's locomotion controller including an outline of the process which converts the genotype into the operational ANN used to control the movement of the artificial creature. Finally we discuss the choice of our common evolutionary, simulation parameters and statistical test of significance used for the experiments conducted in this thesis.

3.1 Physics-Based Simulation

The accurate modelling of the simulation environment plays a crucial part in producing artificial creatures that move and behave realistically in 3D (Taylor and Massey 2001). A dynamic rather than kinematic approach is paramount in allowing for effective artificial evolution to occur. Physical properties such as forces, torques, inertia, friction, restitution and damping need to be incorporated into the artificial evolutionary system. To this end, the Vortex physics engine (CM Labs 2002) was

employed to generate the physically realistic artificial creature and its simulation environment.

By virtue of conducting our artificial evolution within a physically accurate virtual world, rich dynamical interactions are able to occur between the simulated creature and its environment. This in turn enables complex walking behaviors to emerge as the creature evolves the use of its sensors to control the actuators in its limbs through dynamical interactions with the environment. Furthermore, the accurate modelling of physical forces results in creature locomotion that is both realistic and biologically interesting.

3.1.1 Vortex Physics Engine

The Vortex physics engine is a commercial-off-the-shelf simulation toolkit developed by Critical Mass Laboratories (CM Labs 2002). It consists of a set of libraries that comprise of C^{++} routines for robust rigid-body dynamics, collision detection, contact creation, and collision response. Vortex is currently being used by NASA's Autonomy and Robotics Group to develop autonomous rovers for possible deployment in future Mars exploration programs (CM Labs Press Release 2003). The advantages of Vortex include allowing the simulation of natural behavior of objects in the physical world and offering fast yet accurate computation of the dynamical forces that act on the simulated objects. Using the supplied libraries, real-time interactive 3D simulations can be created in which objects, particularly jointed objects moving under constraints, exhibit natural movement under various environmental conditions. However, as Vortex is a constraint-based simulation, it naturally suffers from increasingly higher computational requirements as the number of objects being simulated in the world increases. As such, the design of the artificial creature and its world are kept relatively simple in order to maintain a reasonable run time, especially when conducting the evolutionary experiments.

3.2 Creature Morphology

The artificial creature is a basic quadruped with 4 short legs. Each leg consists of an upper limb connected to a lower limb via a hinge (one degree of freedom) joint and is in turn connected to the torso via another hinge joint. Therefore, there are 8 degrees of freedom overall. The mass of the torso is 1g and each of the limbs is 0.5g. The torso has dimensions of $4 \times 2 \times 1$cm and each of the limbs has dimensions of $1 \times 1 \times 1$cm. In terms of its morphological dimensions, the creature can be visualized as some type of insect. A biological equivalent in terms of size and weight can be found in beetles, where their body lengths range from 0.25mm to 16cm and body mass from 0.4mg to 30g (Grzimek 1984). Research into evolving tiny creatures is being given more attention lately, especially in the field of nanotechnology. A screen dump of the actual creature within the simulation world is shown in Figure 3.1 and a geometric representation of the creature is given in Figure 3.2.

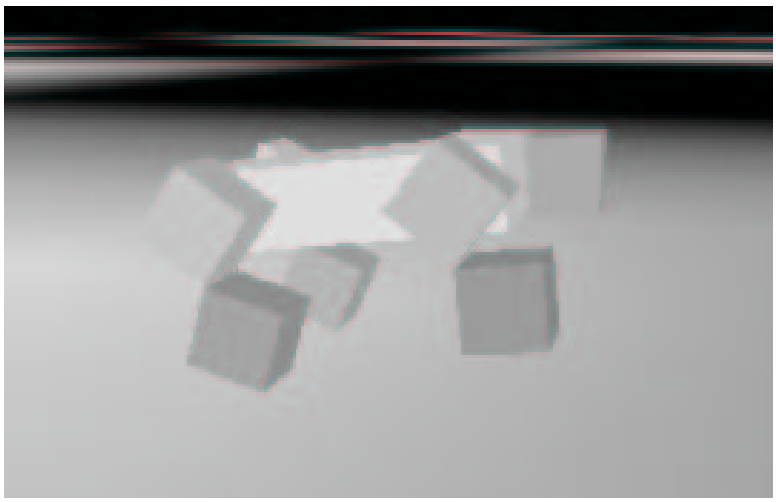

Figure 3.1: A screen dump of the simulated quadruped in the simulation world.

The hinge joints are allowed to rotate between 0 to 1.57 radians. Each of the hinge joints is actuated by a motor that generates a torque producing rotation of the connected body parts about that hinge joint. The creature's overall central nervous system is illustrated in Figure 3.3 where the three letter abbreviations identify each of the 8 different limbs. The first letter denotes (**U**)pper or (**L**)ower, the second denotes to (**F**)ront or (**B**)ack, and the third denotes (**R**)ight or (**L**)eft.

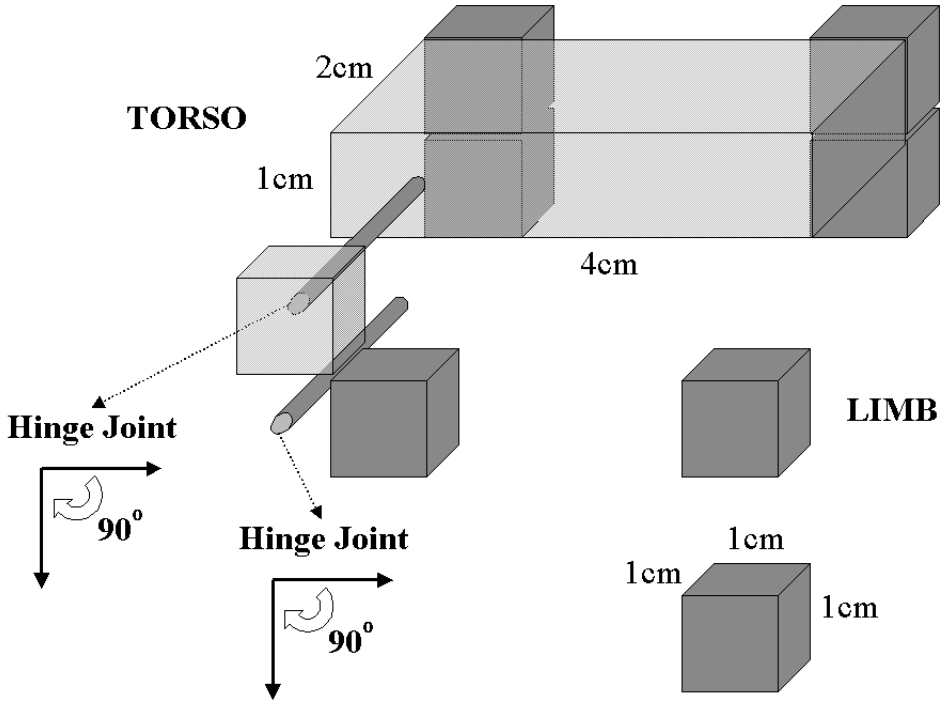

Figure 3.2: A geometric representation of the simulated quadruped.

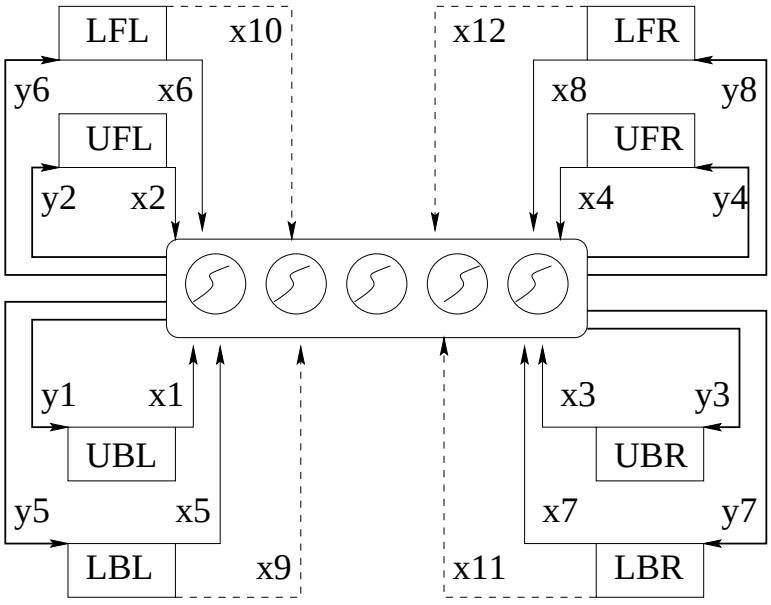

Figure 3.3: A diagrammatic representation of the simulated quadruped's central nervous system.

Correspondingly, the artificial creature has 12 sensors and 8 actuators. The 12 sensors consist of 8 joint angle sensors $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ corresponding

to each of the hinge joints and 4 touch sensors $(x_9, x_{10}, x_{11}, x_{12})$ corresponding to each of the 4 lower limbs of each leg. The joint angle sensors (x_1–x_8) return continuous values in radians whereas the touch sensors (x_9–x_{12}) return discrete values, 0 if no contact with the ground and 1 if contact is made. The choice of inputs was decided upon after a review of the literature where joint angle and touch sensors were the most commonly used types of input to the controller (eg. Sims 1994a; Komosinski 2000; Paul and Bongard 2001; Bongard and Pfeifer 2002). The 8 actuators $(y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8)$ represent the motors that control each of the 8 articulated joints of the creature. These motors are controlled via outputs generated from the ANN controller which is then used to set the desired velocity of rotation of the connected body parts about that joint. A summary of the sensors is given in Table 3.1 followed by the actuators in Table 3.2.

Sensor Number	**Detects Angle Between**	**Value Returned**
x_1	Torso & upper back left limb	$[0, 1.57]$
x_2	Torso & upper front left limb	$[0, 1.57]$
x_3	Torso & upper back right limb	$[0, 1.57]$
x_4	Torso & upper front right limb	$[0, 1.57]$
x_5	Upper & lower back left limbs	$[0, 1.57]$
x_6	Upper & lower front left limbs	$[0, 1.57]$
x_7	Upper & lower back right limbs	$[0, 1.57]$
x_8	Upper & lower front right limbs	$[0, 1.57]$
Sensor Number	**Detects Contact Between**	**Value Returned**
x_9	Lower back left limb & ground	$\{0, 1\}$
x_{10}	Lower front left limb & ground	$\{0, 1\}$
x_{11}	Lower back right limb & ground	$\{0, 1\}$
x_{12}	Lower front right limb & ground	$\{0, 1\}$

Table 3.1: Description of the simulated quadruped's 12 sensors that provide inputs to the ANN controller.

3.3 Creature Controller

3.3.1 Artificial Neural Networks

An ANN may be described as a directed graph: $G(N, A, \psi)$, where N is a set of nodes, A denotes the connections between the nodes, and ψ represents the

Actuator Number	Controls Joint Between	Velocity Range
y_1	Torso & upper back left limb	$[-5, +5]$
y_2	Torso & upper front left limb	$[-5, +5]$
y_3	Torso & upper back right limb	$[-5, +5]$
y_4	Torso & upper front right limb	$[-5, +5]$
y_5	Upper & lower back left limbs	$[-5, +5]$
y_6	Upper & lower front left limbs	$[-5, +5]$
y_7	Upper & lower back right limbs	$[-5, +5]$
y_8	Upper & lower front right limbs	$[-5, +5]$

Table 3.2: Description of the simulated quadruped's 8 actuators that receive outputs from the ANN controller.

learning rule which enables the strengths of inter-neuron connections to be automatically adjusted. A node receives its inputs from an external source or from other nodes in the network. The node undertakes some processing on its inputs and sends the result as its output. The processing function of a node is called the activation function. The activation, a, is calculated as a weighted sum of the inputs to the node in addition to a constant value called the bias.

The following notations will be used to describe a single hidden layer feed-forward ANN:

- I and H are the number of input and hidden units respectively.
- $\mathbf{X}^p \in \mathbf{X} = (x_1^p, x_2^p, \ldots, x_I^p), p = 1, \ldots, P$, is the pth pattern in the input feature space $\mathbf{X}$ of dimension I, and P is the total number of patterns.
- $\mathbf{Y}_o^p \in \mathbf{Y_o}$ is the corresponding scalar of pattern $\mathbf{X}^p$ in the output target space $\mathbf{Y_o}$.
- w_{ih} and w_{ho}, are the weights connecting input unit i, $i = 1, \ldots, I$, to hidden unit h, $h = 1 \ldots H$, and hidden unit h to the output unit o.
- $\Theta_h(\mathbf{X}^p) = \sigma(a_h)$; $a_h = \sum_{i=0}^{I} w_{ih} x_i^p$, $h = 1, \ldots, H$, is the h^{th} hidden unit's output corresponding to the input pattern $\mathbf{X}^p$, where a_h is the activation of hidden unit h, and $\sigma(.)$ is the activation function that is taken in this paper to be the logistic function $\sigma(z) = \frac{1}{1+e^{-Dz}}$, with D the function's sharpness or steepness and is taken to be 1.

- $\hat{Y}_o^p = \sigma(a_o)$; $a_o = \sum_{h=0}^{H} w_{ho}\Theta_h(\mathbf{X}^p)$ is the network output and a_o is the activation of output unit o corresponding to the input pattern $\mathbf{X}^p$.

The following pseudocode describes the functioning of a single hidden layer feed-forward ANN in operation.

1. Until termination conditions are satisfied, do
   (a) for each input pattern, $(\mathbf{X}^p, Y_o^p)$, apply the following steps
      i. Inject the input pattern $\mathbf{X}^p$ into the network.
      ii. Calculate the output, $\Theta_h(\mathbf{X}^p)$, for each hidden unit h.
      iii. Calculate the output, $\hat{Y_o}^p$, for each output unit o.

3.3.2 Evolutionary Artificial Neural Networks

Traditionally ANNs are trained using learning algorithms such as *back-propagation* (BP) (Rumelhart, Hinton, and Williams 1986) to determine the connection weights between nodes. However such methods are gradient-based techniques which usually suffer from the inability to escape from local minima when attempting to optimize the connection weights. To overcome this problem, evolutionary approaches have been proposed as an alternative method for optimizing the connection weights. ANNs evolved via this method is thus referred to as evolutionary ANNs (EANNs). In the literature, research into EANNs usually involves one of three approaches: (1) evolving the weights of the network (Fogel, Fogel, and Porto 1990; Belew, McInerney, and Schraudolph 1992), (2) evolving the architecture (Miller, Todd, and Hegde 1989; Kitano 1990), or (3) evolving both simultaneously (Koza and Rice 1991; Angeline, Saunders, and Pollack 1994). For a thorough review of EANNs, refer to the comprehensive survey conducted by Yao (1999).

Our objective is to evolve ANNs that can perform successfully as locomotion controllers for artificial creatures. Here we will attempt to optimize both the connection weights and number of hidden nodes through evolutionary methods. Other architectural aspects of the ANN such as types of connections between layers,

types of transfer functions and number of input/output units have been kept fixed and are not subjected to evolutionary optimization.

3.3.3 Controller Architectures

The choice of ANN architectures used for controller evolution is normally made arbitrarily when evolving both simulated (Reeve 1999; Bongard and Paul 2001; Pasemann, Steinmetz, Hulse, and Lara 2001a; Paul and Bongard 2001; Reil and Massey 2001; Bongard 2002a; Bongard and Pfeifer 2002; Reil and Husbands 2002) as well as physical artificial creatures (Husbands, Harvey, Jakobi, Thompson, and Cliff 1997; Lund and Hallam 1997; Floreano and Urzelai 1998; Floreano and Mondada 1998; Nolfi and Floreano 1999; Floreano and Urzelai 2000; Nolfi and Floreano 2002; Nolfi 2002). Usually some form of recurrency is used in the ANN, either partially (Bongard and Paul 2001; Pasemann, Steinmetz, Hulse, and Lara 2001a; Paul and Bongard 2001; Bongard and Pfeifer 2002; Bongard 2002a) or fully (Floreano and Urzelai 1998; Floreano and Mondada 1998; Nolfi and Floreano 1999; Reeve 1999; Floreano and Urzelai 2000; Reil and Massey 2001; Nolfi and Floreano 2002; Nolfi 2002; Reil and Husbands 2002). On the other hand, simple direct connections between sensor inputs and motor outputs have also proven to be sufficient for evolving robots controllers with simple behaviors that can accomplish the set task (Lund and Hallam 1997; Pasemann, Steinmetz, Hulse, and Lara 2001a; Nolfi 2002). As such, it remains unclear from the body of literature what types of ANN architecture should be used to evolve controllers for artificial creatures.

We will now introduce four different types of ANN architecture for controller evolution where in the next chapter, we will attempt to provide some characterization of the search space associated with each type of controller architecture. The first type of ANN is a simple feed-forward ANN and is denoted NNType0 (Fig. 3.4.1). The second type of ANN is a feed-forward ANN augmented with direct connections from input to output units and is denoted NNType1 (Fig. 3.4.2). The third type of ANN is a feed-forward network with recurrency on the hidden units (Elman-type ANN (Elman 1990)) and is denoted NNType2 (Fig. 3.4.3). The last

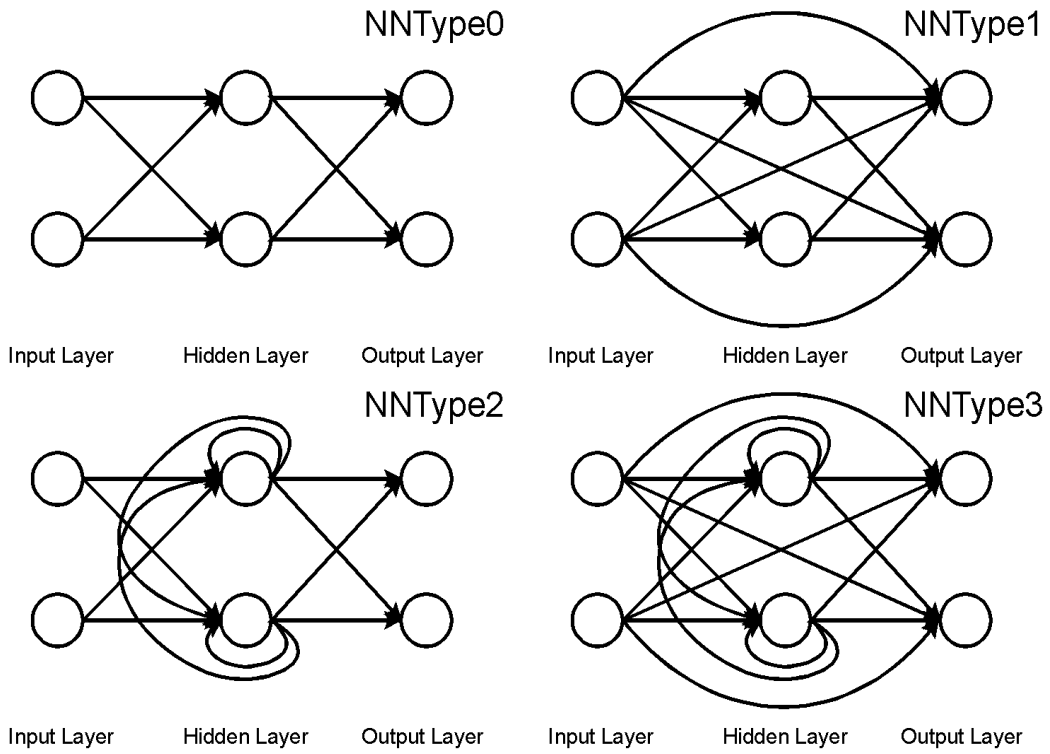

Figure 3.4: Diagrammatic representation of 4 types of ANN architecture: 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right).

type of ANN is a feed-forward network augmented with both direct connections from input to output units as well as recurrency on the hidden units and is denoted NNType3 (Fig. 3.4.4).

Recurrent connections were included to allow the creature's controller to learn state-dependent dynamics of the system. Direct input-output connections were also included in the controller's architecture to allow for direct sensor-motor mappings to evolve that do not require hidden layer transformations. For all four ANN architectures, all units in the preceding layer are fully-connected to all units in the following layer. A bias term is also incorporated into the calculations of the activations of the hidden as well as output units.

All four ANN architectures have a variable hidden layer in terms of its number of active hidden units. This is an integral feature of the ANNs that is essen-

tial for the purposes of this investigation as explained in Section 1.2. Experiments and analyses found later in this thesis will further clarify this point.

3.4 Genotype Representation

Our chromosome is a class that contains one matrix Ω and one vector ρ. The matrix Ω is of dimension $(I + H) \times (H + O)$. Each element $\omega_{ij} \in \Omega$, is the weight connecting unit i with unit j, where $i = 0, \ldots, (I - 1)$ is the input unit i, $i = I, \ldots, (I + H - 1)$ is the hidden unit $(i - I)$, $j = 0, \ldots, (H - 1)$ is the hidden unit j, and $j = H, \ldots, (H + O - 1)$ is the output unit $(j - H)$.

The vector ρ is of dimension H, where $\rho_h \in \rho$ is a binary value used to indicate if hidden unit h exists in the network or not. As such, it works as a switch to turn a hidden unit on or off. The sum, $\sum_{h=0}^{H} \rho_h$, represents the actual number of hidden units in a network, where H is the maximum number of hidden units. The use of ρ allows a hidden node to evolve even if it is not active during certain periods of the evolutionary optimization process.

The chromosome has two additional components when the crossover and mutation rates are also subjected to evolutionary optimization and self-adapted in the algorithms. These additional elements are the crossover rate δ and the mutation rate η. The addition of these last two elements to the genotype representation allows simultaneous training of the weights in the network and selecting a subset of hidden units as well as allowing for the self-adaptation of crossover and mutation rates during optimization.

A direct encoding method was chosen to represent these variables in the genotype as an easy-to-implement and simple-to-understand encoding scheme. Other more complex direct as well as indirect encoding schemes such as those involving developmental mechanisms may prove to be useful and represents possible future work extending from this investigation. A summary of the variables used in the chromosome to represent the artificial creature's genotype is listed in Table 3.3. The mapping of the chromosome into the ANN is depicted in Figure 3.5.

Variable	Representing	Value Type	Value Range
Ω	ANN Connection Weights	Real	$]-\infty,+\infty[$
ρ	Active Hidden Units	Discrete	$\{0,1\}$
δ	Crossover Rate *	Real	$[0,1]$
η	Mutation Rate *	Real	$[0,1]$

Table 3.3: Description of the variables used in the chromosome to represent the artificial creature's genotype. * denotes elements present only in algorithms that use self-adaptation of crossover and mutation rates.

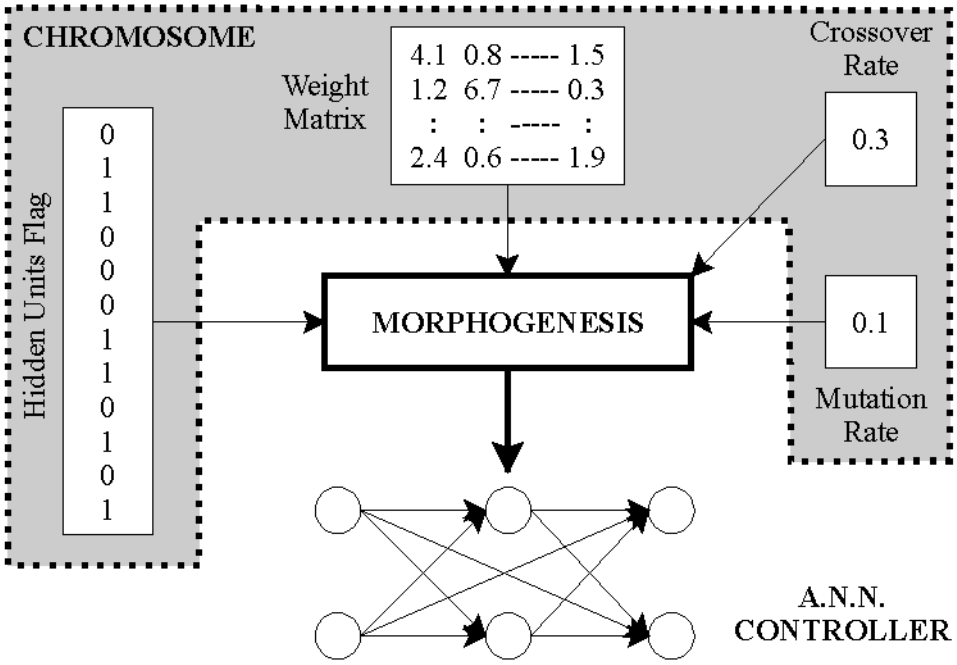

Figure 3.5: A diagram illustrating the mapping from a chromosome to an ANN controller.

3.4.1 Fitness Functions

The fitness f_1 of each locomotion controller represented in the genotype g is defined to be simply

$$f_1 = \Uparrow \ d(g) \tag{3.1}$$

where d refers to the horizontal Euclidean distance achieved by the creature as controlled by the ANN at the end of the evaluation period of 500 timesteps. In other

words, d measures the planar locomotion distance travelled by the artificial creature over the designated period. In experiments involving multi-objective evolutionary optimization of the locomotion controller, a second fitness f_2 is defined to be

$$f_2 = \Downarrow \sum_{h=0}^{H} \rho_h \tag{3.2}$$

where the number of active hidden units in used in the ANN controller is counted by summing the binary vector ρ_h, which is part of the genotype g. Therefore, the first objective is to maximize the horizontal locomotion achieved by the artificial creature and where a second objective is involved, to minimize the use of nodes in the hidden layer of the ANN controller, which will in turn determine the VC-dimension of the controller as explained in Section 1.3. Unless stated explicitly, the fitness of a controller always refers to the first fitness function f_1 which measures the locomotion distance.

3.5 Evolutionary and Simulation Parameters

The relevant literature on artificial evolution of physically simulated creatures was reviewed to ascertain suitable parameter values for use in the evolutionary runs to be conducted in the experiments. Table 3.4 summarizes the parameters used for these types of experiments.

Authors	No. of Generations	Population Size	No. of Timesteps	No. of Repeats
Lipson and Pollack (2000)	300–600	200	12	N/A
Bongard and Paul (2000)	300	300	20000	10
Hornby and Pollack (2001a)	100–500	100	N/A	10
Taylor and Massey (2001)	50–100	∼300	200–1000	N/A
Otsu, Ishiguro, Fujii, Aoki, and Eggenberger (2001)	500	100	N/A	5
Komosinski and Rotaru-Varga (2001)	120–800	200	50–500	10
Bongard (2002a)	50	200	500	10
This Thesis	1000	30	500	10

Table 3.4: A comparison of the evolutionary and simulation parameters used for evolving artificial creatures in simulation.

As can be seen, the parameters used in the literature varied significantly. The evolutionary parameters used in this thesis were chosen after both reviewing the literature and conducting a number of preliminary experiments. This was to ensure that the evolutionary runs balanced both time requirements and quality of solutions obtained. These values are given in the last line of Table 3.4. It may be noticed that the number of generations we have chosen is above the normal range used in the literature. This is to ensure that the evolutionary optimization has been given enough time to converge to a reasonably optimal solution. Secondly, the use of a small population size is also noted with our proposed multi-objective evolutionary optimization algorithm. Small population sizes have been previously shown to be advantageous when evolving ANNs using GAs (Foster, McCullagh, and Whitford 1999). Furthermore, preliminary experiments have shown that a large population size was not essential for evolving successful locomotion controllers (Teo and Abbass 2002a; Teo and Abbass 2002b; Teo and Abbass 2002c). One reason is that the objective of minimizing the number of hidden units which is discrete in nature imposes an upper bound on the possible number of non-dominated solutions that can exist in the population (see Section 5.1 for an explanation of non-dominance and Pareto optimality). This upper bound is simply the maximum number of hidden units allowed + 1 (see second paragraph of Section 4.4 for further explanations). A population size of 30 is approximately double the bound on the number of non-dominated solutions that can exist in any population using this approach (Abbass 2002a). Since our approach is an elitist Pareto approach, we only preserve the non-dominated set in each population. Therefore, we do not need to introduce additional methods for reducing the number of Pareto solutions when they exceed a certain threshold as in the case of the PDE algorithm (Abbass, Sarker, and Newton 2001) where a neighborhood function was used, and NSGA-II (Deb, Agrawal, Pratab, and Meyarivan 2000) where a niching strategy was used.

3.5.1 Statistical Testing

In order to test for the presence of significant statistical differences between two sets of results, the paired two-sample t-test (Runyon, Haber, Pittenger, and Coleman 1996) is used throughout all experimental setups in this thesis. Since the main objective of this work is to investigate the automatic generation of locomotion controllers, all tests of significance are always conducted in relation to the f_1 objective which evaluates locomotion fitness. The t-statistic is calculated using the direct difference method (Runyon, Haber, Pittenger, and Coleman 1996) which is given by the equation

$$t = \frac{\bar{D}}{\sqrt{\frac{\sum D^2 - \frac{(\sum D)^2}{N}}{N(N-1)}}} \tag{3.3}$$

where $\bar{D} = \bar{X}_1 - \bar{X}_2$ and $\bar{X}_1$, $\bar{X}_2$ are the sample means of the two groups and D is the difference between the corresponding pairs of random variables. The 10 different initial populations are fixed for all experiments by using the same set of 10 seeds corresponding to each of the 10 individual runs. N is the sample size of a single group and the degree of freedom is approximated by taking $N - 1$. For all statistical testing of results, a two-tailed test at both significance levels of $\alpha = 0.05$ (95% confidence interval, t-value = 2.262) and $\alpha = 0.01$ (99% confidence interval, t-value = 3.250) are conducted. Bracketed t-values indicate that the sample mean of the group being tested, X_2, is lower compared to the sample mean group of results tested against, X_1.

In experiments involving comparisons between different ANN architecture types, the tests for statistical significance are always made against the NNType0 architecture. As explained earlier in Section 3.3.3, the NNType0 architecture is a simple feed-forward artificial neural network with connections only between the input-hidden and hidden-output layers. It does not have any other additional direct nor recurrent connections that are present in the other types of controller architectures. Thus, the NNType0 controllers have the most minimal architectures among

the different types of controller architectures assuming that the number of hidden units is fixed. Hence, to test whether more complex types of ANN architectures with additional connections provided significantly different locomotion capabilities, t-tests are always carried out against the NNType0 architecture.

3.6 Chapter Summary

The design of our physically-based artificial evolutionary system for evolving the locomotion controllers of a virtually embodied creature was described. The choice of parameters to be used in the experimental sections of this thesis were also discussed and justified. In the next chapter, we will attempt to characterize the fitness landscapes of four ANN architectures to investigate which type of network will be most suitable for the artificial evolution of controllers.