Chapter 5

Multi-Objective Controller Evolution

[1] The artificial evolution conducted in prior studies have mainly focused on single objectives, for example walking, swimming, light-following, block-pushing or obstacle avoidance (Sims 1994b; Komosinski and Rotaru-Varga 2000; Hornby and Pollack 2001a; Bongard 2002a). Although there have been some studies that appear to have multi-objectivity present in the evolutionary system, such as predator-prey simulations (Cliff and Miller 1996; Nolfi and Floreano 2000; Floreano, Nolfi, and Mondada 2001), body-brain co-evolution (Bongard and Paul 2000; Hornby and Pollack 2001a) and evolution by physical competition (Sims 1994b), they do not explicitly impose the evolutionary search on distinctly different optimization criteria. In other words, these studies do not explicitly qualify the solutions in terms of a Pareto set (explained in next section), which is a focal concept in evolutionary multi-objective optimization (EMO). Consequently, the resulting artificial creatures cannot exhibit clear trade-offs in terms of their different evolutionary goals. Here, we propose a methodology for multi-objective evolution of creature controllers that emphasizes the generation of Pareto optimal sets of solutions, that is the generation of results that explicitly trade-off between two different and conflicting optimization

[1] Some of the material presented in this chapter have been previously published in Teo and Abbass (2002a; 2002b; 2002c).

objectives. More specifically, we will attempt to simultaneously minimize the number of hidden units used in the creature's ANN controller while at the same time maximize the horizontal distance travelled by the creature as guided by its ANN controller. Hence, our proposed approach will produce a Pareto optimal set of controllers that have clear delineations between optimizing network size and locomotion capability through an EMO process.

First, we explain the concept of non-dominance and Pareto optimality. Then, we present an overview of the literature concerning EMO algorithms. This is followed by a review of the PDE family of EMO algorithms, which our proposed Pareto EMO algorithm for generating controllers is based upon. Next, we explain in detail our proposed algorithm called SPANN. This is then followed by a discussion of the experimental setup for evolving locomotion controllers using the proposed Pareto EMO methodology. As the experiments from the previous chapter did not show any discernable differences between the search space difficulties associated with the four different types of ANN architectures proposed in Section 3.3.3, we will continue to experiment with all four ANN types. This will ascertain whether or not significant advantages can be offered by the different ANN architectures under an EMO paradigm. The remainder of the chapter presents a detailed discussion of the results from these experiments.

5.1 Dominance and Pareto Optimality

The optimization problem (hereafter referred to as P1) can be stated as

$$\text{(P1): Minimise } f(x)$$
$$\text{subject to: } \theta(x) = \{x \in R^n \mid G(x) \leq 0\}$$

where x is the set of decision variables, $f(x)$ is the objective function, $G(x)$ is a set of constraints, and $\theta(x)$ is the set of feasible solutions. If the optimization problem is maximization, it is equivalent to a minimization problem by multiplying the objective by (-1). Also, if a constraint is an equation, it can be represented by two inequalities — one is "less than or equal" and the other is "greater than or

equal". A "greater than or equal" inequality can be transformed to a "less than or equal" inequality by multiplying both sides by (-1). In short, any optimization problem can be represented in the previous general form.

Two important types of optimal solutions will be used in this thesis, local and global optimal solutions. Let us define the open ball, that is a neighborhood centered on $\bar{x}$ and defined by the Euclidean distance δ, as follows

$$B_\delta(\bar{x}) = \{x \in R^n \mid ||x - \bar{x}|| < \delta\}$$

Definition 1: Local optimality A point $\bar{x} \in \theta(x)$ is said to be a local minimum of the optimisation problem **iff** $\exists\ \delta > 0$ such that $f(\bar{x}) \leq f(x),\ \forall x \in (B_\delta(\bar{x}) \cap \theta(x))$.

Definition 2: Global optimality A point $\bar{x} \in \theta(x)$ is said to be a global minimum of the optimization problem **iff** $f(\bar{x}) \leq f(x),\ \forall x \in \theta(x)$.

Usually, there is more than a single objective to be optimized in real life applications. In this case, the problem is called a *multi-objective optimization problem* (MOP). The problem P1 can be re-defined as a general multi-objective optimization problem, MOP1, by replacing the objective function $f(x)$ with a vector of objectives $F(x)$ as follows

$$\text{(MOP1): Minimize } F(x)$$
$$\text{subject to: } \theta(x) = \{x \in R^n \mid G(x) \leq 0\}$$

When the objectives are in conflict, the existence of a unique optimal solution is no longer a valid concept. The solution which satisfies the optimality conditions of one objective may be a bad solution for another. Consequently, we need to redefine the concepts of local and global optimality in multi-objective problems. To do this, we define two operators, $\ncong$ and $\prec$ and then assume two vectors, X and Y. $X \ncong Y$ **iff** $\exists\ x_i \in X$ and $y_i \in Y$ such that $x_i \neq y_i$. $X \prec Y$ **iff** $\forall\ x_i \in X$ and $y_i \in Y,\ x_i \leq y_i$, and $X \ncong Y$. $\ncong$ and $\prec$ can be seen as the "not equal to" and "less than" operators over two vectors. We can now define the equivalent concepts of local and global optimality in a MOP.

Definition 3: Local efficient (non-inferior) solution: A vector of objective values $F(\bar{x}),\ \bar{x} \in \theta(x)$ is said to be a local efficient solution of MOP **iff** $\nexists\ x \in (B_\delta(\bar{x}) \cap \theta(x))$ such that $F(x) \prec F(\bar{x})$ for some positive δ.

Definition 4: Global efficient (non-inferior) solution: A vector of objective values $F(\bar{x}),\ \bar{x} \in \theta(x)$ is said to be a local efficient solution of MOP **iff** $\nexists\ x \in \theta(x))$ such that $F(x) \prec F(\bar{x})$.

Definition 5: Pareto solutions: A point $\bar{x} \in \theta(x)$ is said to be a Pareto solution of MOP **iff** $F(\bar{x})$ is a global efficient solution of MOP.

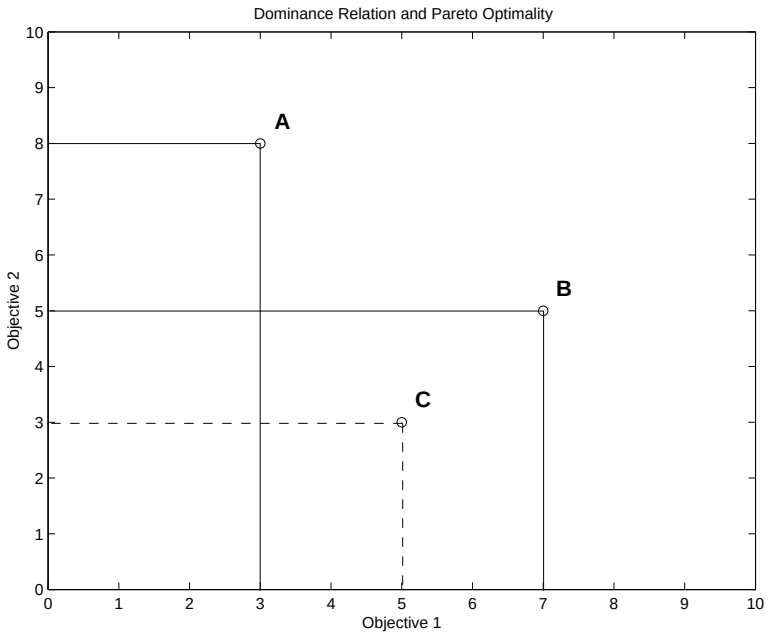

Figure 5.1: Diagram illustrating the concept of dominance and Pareto optimality. X-axis: Objective 1, Y-axis: Objective 2.

The concept of dominance and Pareto optimality is depicted in Figure 5.1. Let us consider the case where there are three solutions A, B, and C and assume that the two objectives 1 and 2 are to be maximized. A is not dominated by any other solution since it has the highest value for objective 2. Similarly, B is not dominated by any other solution since it has the highest value for objective 1. C is not dominated by A since it has a higher value for objective 1. However, C is

dominated by B since it has lower values for both objectives 1 and 2 compared to B. Hence we have the following situation

$$\begin{aligned} \mathsf{dominate}(A) &= \phi \\ \mathsf{dominate}(B) &= \phi \\ \mathsf{dominate}(C) &= \{B\} \end{aligned}$$

where $\mathsf{dominate}(C)$ denotes the set of solutions that dominate C. Therefore, the set of non-dominated or Pareto optimal solutions are given by

$$\mathsf{Pareto\ Set} = \{A, B\}$$

5.2 Evolutionary Multi-Objective Optimization

EMO combines the fields of evolutionary computation with multiple criteria decision-making for solving multi-objective optimization problems (Zitzler 1999; Deb 2001; Coello Coello, Van Veldhuizen, and Lamont 2002). EMO is an established sub-field of optimization and has been utilized for solving both theoretical and practical multi-objective optimization problems for over ten years (Zitzler 2002). A large range of practical applications of EMO to real-life problems across a host of different disciplines can be found in the reference texts by Deb (2001) and Coello Coello, Van Veldhuizen, and Lamont (2002). The literature surveyed on EMO covering general reviews, specific algorithms and related applications in the areas of robotics and artificial life is summarized in Table 5.1.

As explained in the preceding section, unlike in single-objective optimization, a multi-objective optimization problem gives rise to a number of optimal solutions, known as Pareto optimal solutions, of which none can be said to be better than the others with respect to all objectives. EAs are particularly suited for tackling multi-objective optimization problems by virtue of their population-based nature that allows for the generation of multiple solutions of the Pareto set within a single run (Deb 2001; Coello Coello, Van Veldhuizen, and Lamont 2002). Hence, the primary goal in EMO is to find or to approximate the set of Pareto optimal solutions through an evolutionary optimization process.

Type	Description	Reference
General	-	Goldberg (1989)
Reviews		Zitzler (1999)
		Van Veldhuizen and Lamont (2000a)
		Zitzler, Deb, and Thiele (2000)
		Deb (2001)
		Coello Coello, Van Veldhuizen, and Lamont (2002)
		Laumanns, Thiele, Deb, and Zitzler (2002)
		Zitzler (2002)
Algorithms	VEGA	Schaffer (1984)
	MOGA	Fonseca and Fleming (1993)
	NPGA	Horn, Nafpliotis, and Goldberg (1994)
	NSGA	Srinivas and Deb (1994)
	SPEA	Zitzler and Thiele (1999)
	ELSA	Menczer, Degeratu, and Street (2000)
	IEDS	Parmee, Cvetkovic, Watson, and Bonham (2000)
	MOMGA	Van Veldhuizen and Lamont (2000b)
	NSGA-II	Deb, Agrawal, Pratab, and Meyarivan (2000)
	PAES	Knowles and Corne (2000)
	MOMGA-II	Zydallis, Van Veldhuizen, and Lamont (2001)
	PDE	Abbass, Sarker, and Newton (2001); Abbass and Sarker (2002)
	MPANN	Abbass (2001; 2002a)
	SPEA2	Zitzler, Laumanns, and Thiele (2001)
	PCGA	Kumar and Rockett (2002)
	SPDE	Abbass (2002b)
Applications	Robotics & ICS	Gacogne (1997; 1999)
		Tan and Li (1997)
		Coello Coello, Christiansen, and Aguirre (1998)
		Dozier, McCullough, Homaifar, Tunstel, and Moore (1998)
		Pirjanian (1998; 2000)
		Tan, Lee, and Khor (1999)
		Leger (1999)
		Teo and Abbass (2002a)
		Teo, Nguyen, and Abbass (2003)
	A-Life	Oliveira, de Oliveira, and Omar (2000)
		Oliveira, Bortot, and de Oliveira (2002)
		Kim and Hallam (2002)
		Teo and Abbass (2002b; 2002c; 2003)

Table 5.1: Summary of literature survey on EMO reviews, algorithms and related applications in intelligent control systems (ICS), robotics and artificial life (A-Life).

The seminal work on EMO was that of Schaffer (1984) where the Vector Evaluation Genetic Algorithm (VEGA) was introduced for solving machine learning problems. Goldberg (1989) later outlined a 10-point list of how EMO algorithms can be formulated based on the concept of Pareto dominance, out of which a number of the early and well-known EMO algorithms were developed: Multi-Objective Genetic Algorithm (MOGA) (Fonseca and Fleming 1993), Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas and Deb 1994) and Niched Pareto Genetic Algorithm (NPGA) (Horn, Nafpliotis, and Goldberg 1994). These algorithms share two common properties in that solutions were ranked according to their dominance in the population and diversity was maintained using a niching strategy. However, these algorithms did not use any elite-preserving mechanism and as such, could not guarantee convergence to the Pareto optimal solutions. More recent algorithms have since focused on the use of elitism during the EMO process to improve on the convergence properties, such as Strength Pareto Evolutionary Algorithm (SPEA) (Zitzler and Thiele 1999), Pareto Archived Evolution Strategy (PAES) (Knowles and Corne 2000), Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb, Agrawal, Pratab, and Meyarivan 2000), Multi-Objective Messy Genetic Algorithm (MOMGA) (Van Veldhuizen and Lamont 2000b), Strength Pareto Evolutionary Algorithm 2 (SPEA2) (Zitzler, Laumanns, and Thiele 2001) and Multi-Objective Messy Genetic Algorithm II (MOMGA-II) (Zydallis, Van Veldhuizen, and Lamont 2001).

A special issue of the Evolutionary Computation journal was published on EMO algorithms in 2000 (edited by Deb and Horn) where a number of seminal studies on EMO algorithms were presented. Firstly, multi-objective optimization problems were rigorously defined and the theoretical development of EMO algorithms was reviewed by Van Veldhuizen and Lamont (2000a). This article also presented an early attempt at classifying the different types of EMO algorithms and addressed specific issues such as fitness functions, Pareto ranking, niching, fitness sharing, mating restriction and secondary populations. A systematic comparison of a number of EMO algorithms was presented by Zitzler, Deb, and Thiele (2000)

using a set of six test functions specially chosen to elucidate particular problems associated with the EMO process. In this study, it was shown that elitism is particularly important for success in an EMO search. The PAES algorithm was introduced by Knowles and Corne (2000) as a simple (1 + 1) evolutionary strategy algorithm augmented with local search that is able to generate diverse solutions in solving multi-objective optimization problems. In this study, six variants of PAES were compared to variants of NPGA and NSGA over a diverse suite of six test functions. The results showed that PAES consistently performed well over the range of test functions. Parmee, Cvetkovic, Watson, and Bonham (2000) introduced the concept of an Interactive Evolutionary Design System (IEDS) as a methodology which allows EMO to be an interactive rather than a preset process. It was argued that such an interactive process permits the redefinition of the variable and objective space over the evolutionary process that will lead to a more finely tuned design environment. A simple selection method called local selection, which is based on the comparison of an individual's fitness to a fixed threshold rather than to another individual, was introduced by Menczer, Degeratu, and Street (2000) in an EMO algorithm called Evolutionary Local Selection Algorithm (ELSA). It was shown that ELSA naturally maintained genetic diversity by virtue of the local selection process and performed well for three multi-objective optimization problems.

More recently, the Pareto Converging Genetic Algorithm (PCGA) proposed by Kumar and Rockett (2002) eliminates the use of a niching strategy for diversity maintenance and was shown to produce competitive results on three benchmark problems while at the same time reducing computational cost. Laumanns, Thiele, Deb, and Zitzler (2002) also recently researched on the problem of maintaining diversity among the solutions while still being able to converge to the true Pareto optimal solutions in EMO algorithms. The concept of ϵ-dominance was proposed as a method for overcoming these problems and was shown that algorithms using this methodology for archiving solutions will theoretically converge to the actual Pareto-front in the limit while being able to maintain an optimal distribution of solutions along this front.

5.2.1 EMO in Control, Robotics and Artificial Life

EMO has been previously applied to the automated design of intelligent control systems by Tan and Li (1997) and Tan, Lee, and Khor (1999). There have also been studies on using *true* multi-objective optimization methods for the automatic design of artificial creatures. Pirjanian (1998, 2000) used multi-objective optimization to generate action selection modules in a behavior-based robotics experiment. However, this study utilized conventional mathematical optimization methods and did not make use of an evolutionary optimization approach. Evolutionary methods have been used to solve navigational problems with multiple objectives for 2D mobile agents in simulation (Dozier, McCullough, Homaifar, Tunstel, and Moore 1998; Gacogne 1997; Gacogne 1999). Coello Coello, Christiansen, and Aguirre (1998) also used an EMO approach for a robotics design problem but this experiment involved only a non-autonomous subject in the form of an attached robotic manipulator arm. Multi-objective evolutionary optimization has also been used by Leger (1999) although the focus of the EMO approach was for optimizing the physical configurations of modular robotic components rather than for the generation of autonomous robotic controllers. There have been a number of other studies involving the use of some form of EMO for the design of robotic manipulator arms as reviewed by Coello Coello, Van Veldhuizen, and Lamont (2002).

More recently, Oliveira, Bortot, and de Oliveira (2002) used an EMO approach in an artificial life study of 1D cellular automata for the density classification task problem. It was reported that the use of an EMO approach offered significant advantages in terms of defining and evaluating the fitness of evolving populations over a weighted sum approach carried out in a prior experiment (Oliveira, de Oliveira, and Omar 2000). Kim and Hallam (2002) also recently reported the use of EMO for solving the so-called *Woods* problem, which are goal-search problems for agents starting at random initial positions and having to find an end-state goal position. The objective of the study was to quantify the amount of internal memory states required for the finite state machine controllers to solve a given *Woods* task. Pareto-fronts of discrete internal memory states were minimized in a

trade-off against maximizing the fitness of the agents, which were evaluated as the minimum number of steps required to reach the end-state goal position from the initial starting position. However, the artificial creatures were only very simple 2D agents that acted in a discrete grid-world environment with movement allowed only in the four cardinal directions. In this chapter, we will demonstrate the use of EMO for evolving completely autonomous, embodied and situated creatures that act in a 3D world with fully continuous and non-restrictive movements that trade-off between the number of internal nodes required in the neural network controller and locomotion capability achieved. Furthermore, our experiments are aimed at generating legged locomotion in 3 dimensions rather than wheeled or mobile locomotion behaviors that are restricted to 2 dimensions.

5.3 PDE Algorithm

Abbass et al. first introduced the Pareto-frontier Differential Evolution (PDE) algorithm for vector optimization problems (Abbass and Sarker 2002; Abbass, Sarker, and Newton 2001). PDE is a multi-objective adaptation of the original *Differential Evolution* (DE) algorithm introduced by Storn and Price (1995) for optimization problems over continuous domains. The PDE algorithm outperformed the SPEA algorithm (Zitzler and Thiele 1999) on five benchmark problems in this introductory investigation.

PDE combined with local search was later introduced for evolving ANNs in the MPANN algorithm (Abbass 2001). MPANN was found to be highly effective for knowledge discovery in databases. In subsequent work, the MPANN algorithm was empirically shown to possess better generalization in medical diagnosis of breast cancer whilst incurring a much lower computational cost (Abbass 2002a).

In an extension to PDE, a self-adaptive version called Self-adaptive Pareto Differential Evolution (SPDE) algorithm was proposed to allow for self-adaptation of mutation and crossover rates during the optimization process (Abbass 2002b). Both rates for new individuals are inherited from parents during crossover and mutated

in the same process that occurs for decision variables. SPDE was found to be highly competitive against 13 other EMO algorithms on four benchmark test functions and actually outperformed a number of current state-of-the-art algorithms.

As described above, the SPDE algorithm has been found to be a highly effective algorithm for optimization over a continuous domain. Moreover, it has also been tested successfully for the evolution of ANNs. For these reasons, SPDE was chosen as the algorithm for evolving the creature's controllers since the parameters that are being optimized in the evolutionary process are the real-valued weights of the neural network. Furthermore, it provides an added advantage over the original PDE algorithm since it allows for self-adaptation of the crossover and mutation rates. It should be noted that other EMO algorithms may also be used to evolve the creature's controllers. However, since the objective of this work is to investigate the application of an EMO approach for evolving artificial creature controllers and not a comparison between EMO algorithms, the question of which EMO algorithm will work best for this purpose is beyond the scope of this thesis and remains an open question for future work.

5.4 Proposed SPDE-Based Controller Evolution

More recently, the SPDE algorithm has been combined with MPANN for evolving artificial neural networks called the Self-adaptive Pareto Artificial Neural Network (SPANN) algorithm (Abbass 2003). In this thesis, we propose a modified version of SPANN for controller evolution. There are two major differences between this proposed version and the original version of SPANN. Firstly, SPANN uses back-propagation for learning. In the case of locomotion controller evolution, the task to be learned is not clear-cut as in canonical classification problems. As such, the only learning that takes place in the modified version of SPANN occurs only through evolutionary adaptation. A possible future work would be to investigate the possibility of augmenting the current proposed version of SPANN with lifetime learning by somehow introducing a measure of error associated with the locomotion task,

which would then allow back-propagation to be used. Nolfi (1999) has previously shown how learning through back-propagation can be integrated with evolutionary artificial neural networks to predict the next move state for an autonomous agent in a 2D grid world. However, the prediction task required in this case involved only movements in the four cardinal directions by virtue of the grid world. As such it remains an open question how such a methodology can be applied to 3D physically accurate embodied creatures living in virtual worlds with infinitely large numbers of possible directions for locomotion.

Secondly, the repair function originally used in SPANN for evolving the crossover and mutation rates (which truncates the whole number portion leaving only the decimal portion), though useful for the data mining task, was found to cause premature convergence of these rates to the lower boundary of 0 when evolving controllers. Consequently, the evolutionary optimization process would also prematurely stagnate due to the lack of crossover and mutation during reproduction. Hence a new repair function as explained in Section 5.4.1 is proposed in the modified version of SPANN to overcome this problem. For the remainder of the thesis, this proposed version of the SPANN algorithm for controller evolution is referred to whenever the acronym SPANN is used. The pseudocode of this proposed version of the algorithm is given in the next subsection.

5.4.1 The SPANN Algorithm

The pseudocode for the SPANN algorithm is as follows:

1. Create a random initial population of potential solutions. The elements of the weight matrix Ω are assigned random values according to a Gaussian distribution $N(0, 1)$. The elements of the binary vector ρ are assigned the value 1 with probability 0.5 based on a randomly generated number according to a uniform distribution between $[0, 1]$, otherwise 0. The crossover rate δ and mutation rate η are assigned random values according to a uniform distribution between $[0, 1]$.

2. Repeat

   (a) Evaluate the individuals in the population and label those who are non-dominated.

   (b) If the number of non-dominated individuals is less than 3 repeat the following until the number of non-dominated individuals is greater than or equal to 3:

      i. Find a non-dominated solution among those who are not labelled.

      ii. Label the solution as non-dominated.

   (c) Delete all dominated solutions from the population.

   (d) Repeat

      i. Select at random an individual as the main parent α_1, and two individuals, α_2, α_3 as supporting parents.

      ii. Select at random a variable j.

      iii. **Crossover:** With some probability $Uniform(0,1) > \delta^{\alpha_1}$ or if $i = j$, do

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} + N(0,1)(\omega_{ih}^{\alpha_2} - \omega_{ih}^{\alpha_3}) \tag{5.1}$$

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} + N(0,1)(\omega_{ho}^{\alpha_2} - \omega_{ho}^{\alpha_3}) \tag{5.2}$$

$$\rho_h^{child} \leftarrow \begin{cases} 1 & if\ (\rho_h^{\alpha_1} + N(0,1)(\rho_h^{\alpha_2} - \rho_h^{\alpha_3})) \geq 0.5 \\ 0 & otherwise \end{cases} \tag{5.3}$$

$$\delta^{child} \leftarrow \delta^{\alpha_1} + N(0,1)(\delta^{\alpha_2} - \delta^{\alpha_3}) \tag{5.4}$$

$$\eta^{child} \leftarrow \eta^{\alpha_1} + N(0,1)(\eta^{\alpha_2} - \eta^{\alpha_3}) \tag{5.5}$$

      otherwise

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{\alpha_1} \tag{5.6}$$

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{\alpha_1} \tag{5.7}$$

$$\rho_{h}^{child} \leftarrow \rho_{h}^{\alpha_1} \tag{5.8}$$

$$\delta^{child} \leftarrow \delta^{\alpha_1} \tag{5.9}$$

$$\eta^{child} \leftarrow \eta^{\alpha_1} \tag{5.10}$$

where each variable in the main parent is perturbed by adding to it a Gaussian value $N(0,1)$ multiplied by the difference between the two values of this variable in the two supporting parents. At least one variable in Ω must be changed. If δ or η are not in $[0,1]$, repair by adding (if < 0) or subtracting (if > 1) a random number between $[0,1]$ until δ and η are in $[0,1]$.

iv. **Mutation:** With some probability $Uniform(0,1) > \eta^{\alpha_1}$, do

$$\omega_{ih}^{child} \leftarrow \omega_{ih}^{child} + N(0, \eta^{\alpha_1}) \tag{5.11}$$

$$\omega_{ho}^{child} \leftarrow \omega_{ho}^{child} + N(0, \eta^{\alpha_1}) \tag{5.12}$$

$$\rho_{h}^{child} \leftarrow \begin{cases} 1 & if\ \rho_{h}^{child} = 0 \\ 0 & otherwise \end{cases} \tag{5.13}$$

$$\delta^{child} \leftarrow N(0,1) \tag{5.14}$$

$$\eta^{child} \leftarrow N(0,1) \tag{5.15}$$

(e) Until the population size is M

3. Until maximum number of generations is reached.

5.5 Experimental Setup

Four series of experiments were conducted to compare the evolution of controllers using the four different types of ANN architecture. The fitness of each genotype in these experiments was evaluated according to both the f_1 and f_2 objective functions, which measures the locomotion distance achieved and number of hidden units used by the controller respectively as defined in Section 3.4.1. The evolutionary and simulation parameters used were as reported in Section 3.5: 1000 generations, 30 individuals, 500 timesteps and 10 repeated runs. As with the fitness landscape experiments in Chapter 4, the maximum number of hidden units allowed in the ANN was set to 15. Being the objects of the evolutionary optimization process, the locomotion distance and number of hidden units used in the ANN were recorded for every individual generated in every generation.

First, we analyze the optimization results from the evolution of creature controllers for the four types of ANN architectures. Next, we compare the controllers obtained using our SPANN algorithm against those obtained from using the random search, hill-climbing and random walk algorithms. Then, we analyze the evolutionary dynamics at the individual as well as population level of genotypes generated during the evolutionary optimization process to provide a deeper insight into how the evolution of controllers affects the evolution of locomotion capabilities in a physically simulated artificial creature. This is followed by a characterization of the search space difficulty associated with each of the four types of ANN architectures to investigate whether any of the four ANN architectures provide any significant advantages in terms of evolutionary search for controllers with high locomotion fitness. Finally, we analyze the operational dynamics of the overall best controller evolved for locomotion distance using the SPANN algorithm to ascertain what is actually happening in the creature's limbs as they are controlled by the evolved ANN during locomotion as well as the effect of noise on the performance of the evolved ANN controller.

5.6 Results and Discussion

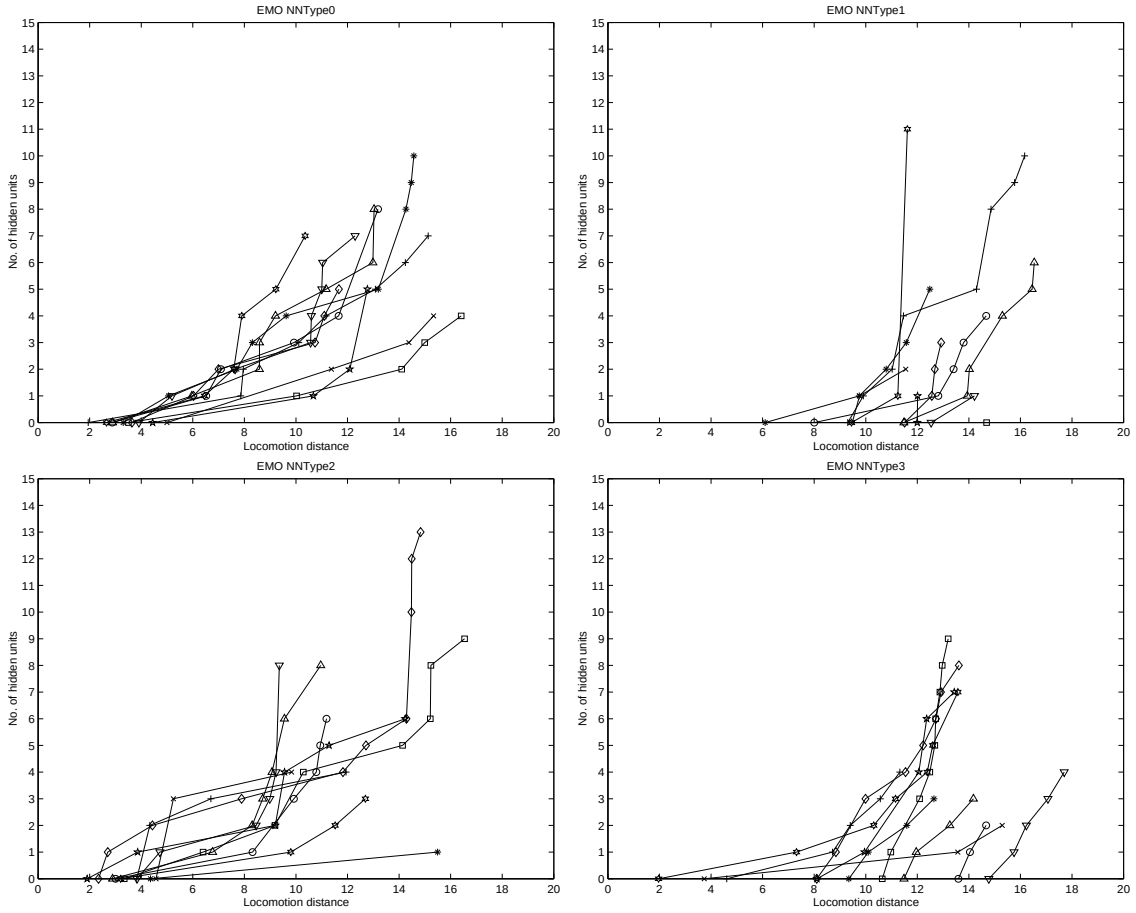

Figure 5.2: Pareto-front of solutions obtained for 10 runs using the SPANN algorithm for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Locomotion distance, Y-axis: No. of hidden units.

The Pareto-fronts achieved at the last generation for each of the 10 runs are plotted in Figure 5.2. It can be noticed that a variety of solutions in terms of controller size and locomotion capability was obtained in the majority of the evolutionary runs. Most of the solutions on the Pareto-frontier comprised of controllers with less than 5 hidden units in the ANN. This is an indication that larger networks did not offer significant advantages in terms of generating better locomotion capabilities compared to smaller networks. There were no obvious differences between

the four different types of ANN architectures. However, having direct input-output connections did have an observable effect on networks with 0 hidden units. The locomotion achieved with NNType1 (Figure 5.2.2) and NNType3 (Figure 5.2.4) architectures ranged between 2 up to almost 15 whereas NNType0 (Figure 5.2.1) and NNType2 (Figure 5.2.3) architectures were clustered between 2 to 4. It should be pointed out that although NNType0 and NNType2 networks with 0 hidden units do not have any sensor-to-motor mappings whatsoever, some small movement is still achieved due to the initial forces generated from the outputs of the these networks since an activation value of zero would still produce an output signal of 0.5 by virtue of the sigmoidal transfer function — however, without any mapping between the input and output layers of the networks, this initial movement is unsustainable due to the lack of synchronization ability (Teo and Abbass 2002a). Hence, NNType1 and NNType3 controllers with direct input-output connections could achieve sufficiently good locomotion capabilities without requiring a hidden layer. The ability of such *pure reactive* agents for solving complex sensory-motor coordination tasks have previously been reported in wheeled robots (Lund and Hallam 1997; Pasemann, Steinmetz, Hulse, and Lara 2001a; Nolfi 2002). These direct connections between the input and output layers also appeared to have generated Pareto optimal networks with smaller sizes in a large majority of the runs. This may be due to the fact that the direct input-output connections are already providing a good mechanism for basic locomotion, thus requiring only a few extra hidden units to further improve on this basic locomotion.

The largest network on the Pareto-front can be found with the NNType2 architecture using up to 13 hidden units. This suggests that the recurrency is somehow making the locomotion task more difficult to learn and thereby adding unnecessary complexity to the task (it must be remembered that the recurrent connections in NNType2 are already themselves adding structural complexity to the ANN architecture). This observation is supported by comparing against the Pareto optimal controllers obtained using the simple feed-forward-only NNType0 architecture, a number of which utilized fewer hidden units than the NNType2 Pareto optimal

controllers but were still able to achieve highly similar locomotion capabilities.

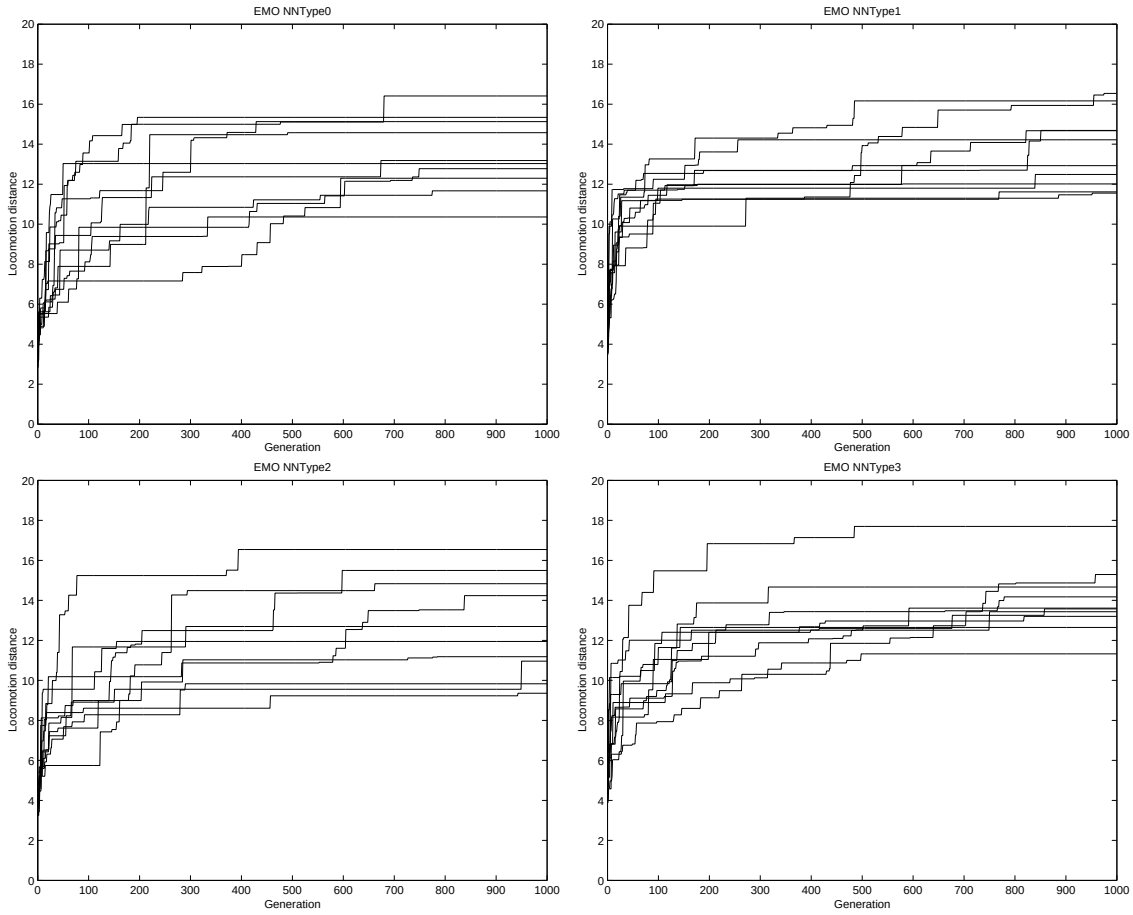

Figure 5.3: Best locomotion distance of Pareto solutions obtained over 1000 generations for 10 runs using the SPANN algorithm for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Generation, Y-axis: Locomotion distance.

The evolution of the Pareto solution for best locomotion distance using the SPANN algorithm for the 10 runs over 1000 generations is depicted in Figure 5.3. No marked differences between the four ANN architectures could be seen. Most of the improvement occurred early during the evolutionary process where in most runs, the best solution exceeded a locomotion capability of beyond 10 units by the 150th generation. Compared to the best solutions obtained using random search, hill-climbing and random walk, the EMO algorithm provided a smoother progress

over time in discovering fitter solutions. This may be an indication that the landscape is explored better using EMO. Among the four different ANN architectures, NNType1 appeared to have the least amount of variation in terms of the best solution obtained over the 10 different runs (Figure 5.3.2). On the other hand, NNType2 showed much larger differences between the 10 runs (Figure 5.3.3). This might be an indication that the additional direct input-output connections exhibited a less rugged landscape compared to additional recurrent-only connections in the ANN when evolving controllers using the EMO algorithm. In the former case, most of the runs were able to proceed along more similar paths due to the presence of a smoother landscape, thereby leading to less variation among solutions.

NNType	Overall Best Locomotion Distance	Average Best Locomotion Distance ± Standard Deviation	t-statistic (against NNType0)
0	16.4104	13.4755 ± 1.8613	-
1	16.5375	13.6866 ± 1.8318	0.30
2	16.5418	12.7079 ± 2.4674	(0.87)
3	17.6994	13.9626 ± 1.7033	0.53

Table 5.2: Comparison of best locomotion distance for Pareto solutions found using the SPANN algorithm over 10 independent runs.

The overall best Pareto solution in terms of locomotion fitness together with the mean best locomotion fitness and standard deviations obtained using the SPANN algorithm are given in Table 5.2. The overall and average best solutions were obtained with the NNType3 architecture although the differences between architectures were small. A t-test at both $\alpha = 0.05$ and $\alpha = 0.01$ significance levels showed no significant differences between the four ANN architectures in terms of the best solutions obtained over 10 independent runs. The lowest average best locomotion fitness was given by NNType2, which also showed a much larger deviation among the best solutions found compared to the other architectures. This supports the earlier observation that the evolutionary path encountered when using additional recurrent-only connections in NNType2 was more rugged, leading to greater variation among solutions.

Table 5.3 lists the global Pareto optimal solutions found by SPANN over

NNType	No. of Hidden Units	Locomotion Distance
0	0	4.9981
	1	10.6792
	2	14.1010
	3	14.9951
	4	16.4104
1	0	14.6870
	4	15.2998
	5	16.4548
	6	16.5375
2	0	4.5740
	1	15.4944
	9	16.5418
3	0	14.7730
	1	15.7506
	2	16.2295
	3	17.0663
	4	17.6994

Table 5.3: Comparison of number of hidden units used and locomotion distance for global Pareto optimal controllers obtained using the SPANN algorithm over 10 independent runs.

the 10 runs for each type of ANN architecture. The solutions with the highest locomotion fitness used between 4 and 9 hidden units in the ANN controller. For NNType0 and NNType3, the maximum number of hidden units required to generate the best locomotion was only 4 hidden units whereas NNType1 required 6 hidden units. NNType2 required the most number of hidden units (9). It is interesting to note that by allowing direct input-output connections in NNType1 and NNType3, controllers which did not use the hidden layer at all (0 hidden units) could generate a sufficiently good locomotion ability, moving the creature up to a distance of 14.7 and 14.8 units respectively. This is empirical proof that perceptron-like controllers, which rely only on input-to-output connections without any internal nodes, are sufficient for generating simple locomotion in a four-legged artificial creature. As previously pointed out, this phenomenon has been previously observed to occur in wheeled robots as well (Lund and Hallam 1997; Pasemann, Steinmetz, Hulse, and Lara 2001a; Nolfi 2002). As such, robots that are only required to perform simple

tasks can be evolved as purely reactive agents, which would dramatically simplify the process of synthesizing successful robot controllers. As previously explained in the first paragraph of this section, for NNType0 and NNType2 controllers with 0 hidden units, there is still an act of force on the creature produced by the zero-activation output from these networks that permit the small initial movements.

5.6.1 SPANN vs. Random Search, Hill-Climbing and Random Walk

NNType	Algorithm	Average Best Locomotion Distance ± Standard Deviation	t-statistic (against SPANN)	No. of Hidden Units
0	SPANN	13.4755 ± 1.8613	-	6.5 ± 2.0
	Random Search	7.5406 ± 0.6733	(9.96)	10.0 ± 2.3
	Hill-Climbing	5.5969 ± 0.9714	(11.39)	7.0 ± 2.3
	Random Walk	9.7725 ± 1.2009	(4.63)	7.4 ± 2.1
1	SPANN	13.6866 ± 1.8318	-	4.1 ± 3.5
	Random Search	10.0931 ± 0.7348	(6.06)	7.4 ± 4.0
	Hill-Climbing	8.4652 ± 2.6246	(5.45)	7.4 ± 2.0
	Random Walk	10.4776 ± 0.9616	(4.84)	7.4 ± 2.2
2	SPANN	12.7079 ± 2.4674	-	6.2 ± 3.5
	Random Search	7.3609 ± 0.7490	(7.02)	10.9 ± 2.5
	Hill-Climbing	7.6568 ± 2.9365	(3.49)	6.4 ± 2.5
	Random Walk	9.5602 ± 1.3372	(3.59)	8.2 ± 2.0
3	SPANN	13.9626 ± 1.7033	-	4.9 ± 2.6
	Random Search	10.2878 ± 1.2747	(5.59)	9.2 ± 4.2
	Hill-Climbing	6.9057 ± 1.5719	(11.46)	6.8 ± 2.2
	Random Walk	10.0333 ± 1.2535	(5.78)	7.0 ± 1.4

Table 5.4: Comparison of best locomotion distance for Pareto/best solutions obtained over 10 independent runs using the SPANN, random search, hill-climbing and random walk algorithms over 10 independent runs.

Table 5.4 provides a comparison of the best results for locomotion distance obtained using the SPANN, random search, hill-climbing and random walk algorithms. For all four ANN architectures, the SPANN algorithm produced controllers that had much higher locomotion capabilities than controllers obtained using random search, hill-climbing and random walk. To confirm that the results obtained

using the proposed SPANN algorithm were significantly superior, a t-test was conducted. At both significance levels of $\alpha = 0.05$ and $\alpha = 0.01$, the best solutions obtained using SPANN over 10 independent runs were all significantly higher in terms of locomotion fitness than random search, hill-climbing and random walk across all four ANN architectures.

The last column of Table 5.4 shows the average number of hidden units used in the ANN of the best controller evolved for locomotion distance obtained over the 10 different runs. It is clear that with the additional optimization objective of minimizing the number of hidden units used in the ANN for the SPANN algorithm, the number of hidden units required by the best controllers obtained from evolution for all four different types of ANN architectures were lower than those obtained using random search, hill-climbing and random walk, which were just optimizing along the single objective of locomotion distance. However, what is interesting is the fact that a significantly higher locomotion distance was achieved with a smaller controller size when the EMO algorithm was used. This may be explained by the strong evolutionary pressures on the survival of controllers within the population during reproduction and selection, thereby playing a significant role in forcing controllers to become increasingly efficient at locomotion as well as requiring fewer active hidden units in the ANN controller. Moreover, the inclusion of a second objective to the optimization process may have provided an extra-dimensional bypass in which the SPANN algorithm was able to reach a fitter solution space compared to random search, hill-climbing and random walk. This phenomenon has been previously encountered during the evolution of walking behavior in a simulated biped robot when additional morphological parameters for size and mass distribution of body segments were added to the original chromosome of the artificial creature (Bongard and Paul 2001). The EMO methodology may have naturally created the extra-dimensional bypass through the optimization of multiple objectives. The extra-dimensional bypass may have also arisen from the usage of ρ, which allows hidden units to continuously evolve even though it may be inactive during certain periods of the evolutionary process.

5.6.2 Evolutionary Dynamics

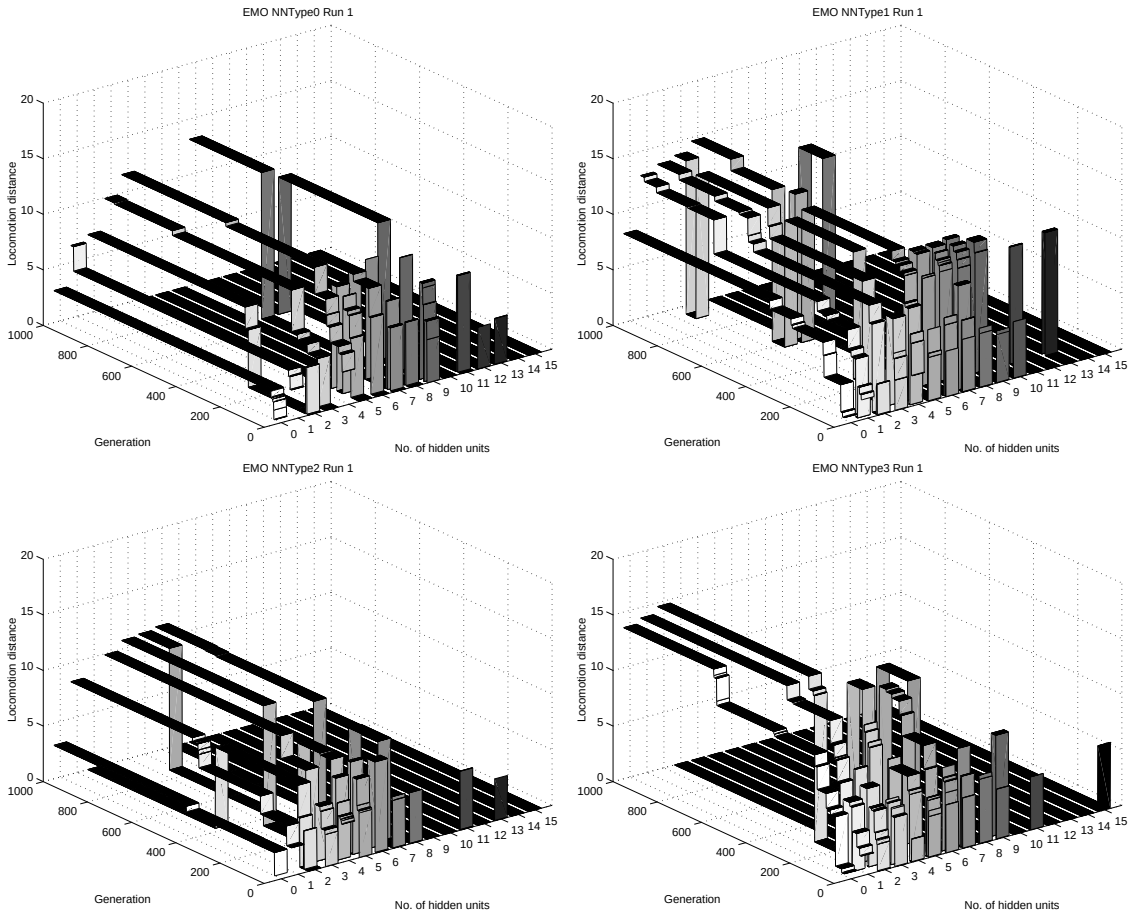

Figure 5.4: Non-dominated solutions generated by the SPANN algorithm over 1000 generations for runs using the first seed for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Generation, Y-Axis: No. of hidden units, Z-axis: Locomotion distance. Additional graphs can be found in the accompanying CD-ROM.

Figure 5.4 illustrates the evolution of non-dominated solutions over 1000 generations from the first run using the four different ANN architectures. Three main results can be concluded from the analysis of these graphs, which are representative of the dynamics of controller evolution from the other 9 runs conducted for each of the architecture types respectively. Firstly, a large variety of solutions in terms of locomotion capability and controller size is maintained throughout the evolutionary

process. The evolution starts off with an even spread of solutions across the range of permissible hidden layer sizes. A high level of evolutionary activity can then be seen to occur before the 250–300th generation. During this evolutionary era, solutions with different controller sizes and locomotion capabilities were actively competing for survival as non-dominated solutions that will be carried over to the next generation as parents. After the initial flurry of evolutionary activity, the optimization process begins to stabilize after the 400–500th generation. Even at the end of 1000 generations, between 3 to 6 different Pareto controllers in terms of the number of hidden units used in the ANN were still present in the optimal set of solutions out of a possible 16 different configurations for active hidden units.

Secondly, the graphs show that some genotypes with a certain hidden layer architecture disappears from the non-dominated set of solutions and then reappears, for example controllers with 8 hidden units in NNType0 (Figure 5.4.1) and controllers with 5 hidden units in NNType2 (Figure 5.4.3). This phenomenon is indicative of the evolutionary search process moving through the fitness landscape by experimenting with different hidden layers sizes and eventually re-discovering a network configuration previously used but now with added locomotion capabilities.

Lastly, we see from the evolutionary dynamics of controller evolution that it is generally very hard for larger controllers with more hidden units to survive due to the strong evolutionary pressure of trading-off the ANN performance and its complexity. This observation is attributed to the fact that a larger controller does not easily lead to locomotion abilities that can't be achieved with a smaller controller in this particular problem. As a result, larger controllers find it hard to compete with smaller controllers in trying to maximize the horizontal distance travelled by the quadruped.

In summary, the multi-objective approach maintains genetic diversity by allowing individuals with different controllers and capabilities to be retained within the genetic pool. This shows that the EMO approach is highly advantageous since it allows for simultaneous maintenance of genetic diversity as well as survival of individuals that exhibit particular advantages over the rest of the population. The

maintenance of genetic diversity using an EMO approach has recently been verified by Abbass and Deb (2003).

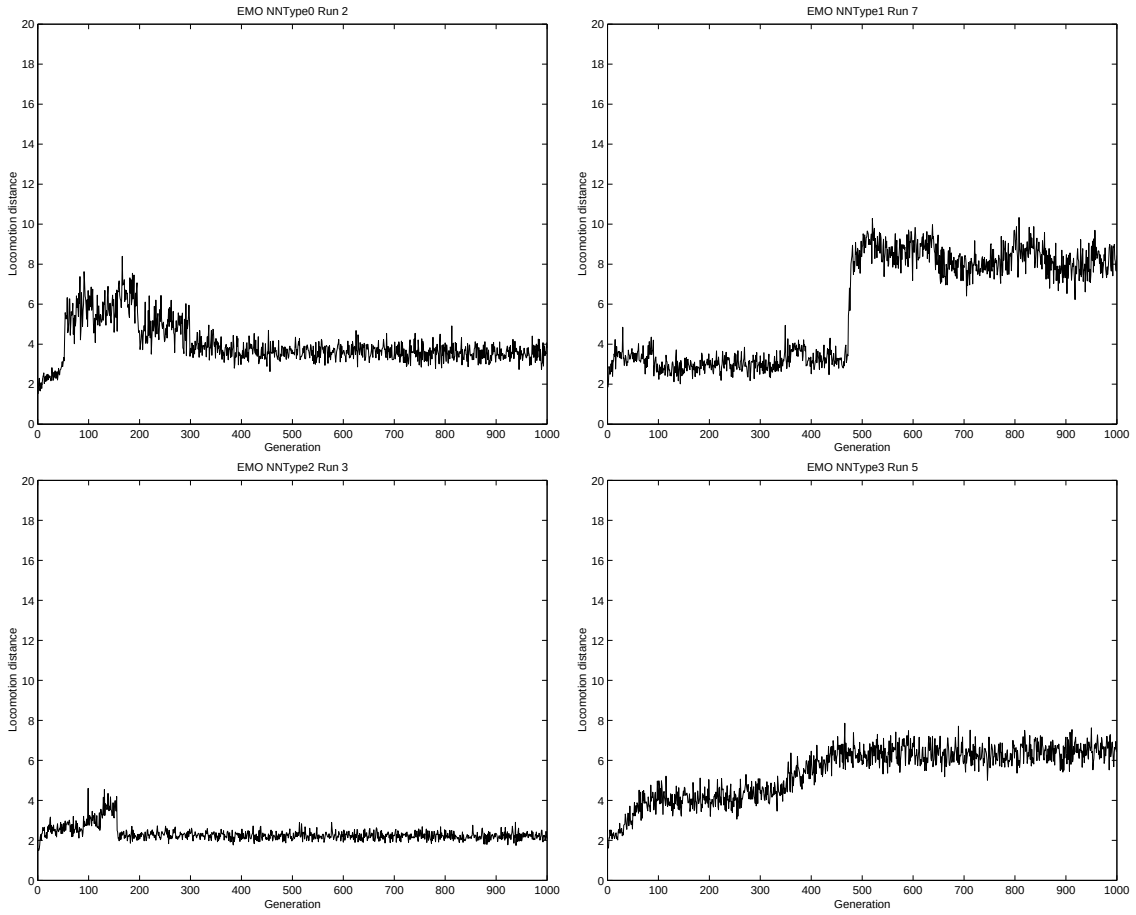

Figure 5.5: Mean locomotion distance of population over 1000 generations (selected seeds only) using the SPANN algorithm for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Generation, Y-Axis: Locomotion distance. Additional graphs can be found in the accompanying CD-ROM.

Next, we analyze the mean of the population for locomotion distance as it evolved over 1000 generations, which is depicted in Figure 5.5. There were generally four trends in the movement of the population mean during the evolutionary optimization process present across the four types of ANN architectures. The most commonly occurring trends in the population means were that of an early increase

followed by a decrease and then stabilizing into a constant range, which is depicted in Figure 5.5.1, and that of a slow and small increase over time, which is depicted in Figure 5.5.4. However, there were also instances where there was a sudden evolutionary jump into a much higher fitness space which was then maintained throughout the evolutionary process, as shown in Figure 5.5.2. This can be explained by reaching an evolutionary peak and maintaining the search process within this high fitness subspace, perhaps assisted by the presence of a large basin of attraction associated with this subspace. This occurrence may also be indicative that the majority of new solutions being generated are located near or close to a newly-found superior solution, which results in the increase in population mean since new solutions being generated have fitness values close to this superior individual. Finally, there were also occurrences of the opposite phenomenon, that of a sudden decrease in the population mean, indicative of reaching an evolutionary peak which is surrounded by low fitness subspaces, as illustrated in Figure 5.5.3. This again suggests the presence of a large basin of attraction, this time associated with a low fitness subspace. Overall, these results support the earlier characterization of the random walk fitness landscape using informational measures (Section 4.5.3.1) which showed that there were a variety of shapes present on the evolutionary landscape and that depending on the initialization and trajectory of the search process, smooth or highly rugged landscape regions may be encountered.

NNType	Average Locomotion Distance ± Standard Deviation
0	3.9848 ± 0.7972
1	5.5943 ± 1.9875
2	3.4466 ± 1.2297
3	5.1650 ± 1.7368

Table 5.5: Mean and standard deviation of all individuals' f_1 fitness generated within all generations and over all 10 runs for SPANN. The average is calculated over 300,000 individual f_1 fitness recorded across the entire evolutionary optimization process.

Table 5.5 lists the average locomotion fitness achieved by all individuals generated throughout the 1000 generations over 10 runs. On average, the fitness

space occupied by the evolving population was highest in NNType1 followed closely by NNType3. Correspondingly, a much lower fitness region of the search space was occupied by the evolving populations of NNType0 and NNType2. This indicates that the NNType1 and NNType3 architectures, which have the direct input-output connections, permitted reproduction of offspring that were generally fitter than those produced by NNType0 and NNType2. The inclusion of direct input-output connections introduces some sort of a lower bound on the performance of controllers having the NNType1 and NNType3 architectures since these direct input-output connections are not subjected to evolutionary pressures and hence are always present in any controller of these types. On the other hand, controllers using NNType0 and NNType2 architectures have no such lower bound since controllers of these types that use no hidden units will not have any mapping between the input and output layers whatsoever. Nonetheless, it should be noted that although the fitness regions occupied by the evolving populations of NNType1 and NNType3 were higher than that of NNType0 and NNType2, no significant advantages were evident in terms of leading the search towards a more optimal final solution since a t-test showed that there were no significant differences between the best locomotion distances of the Pareto solutions found using the four different types of ANN architecture (see Table 5.2).

5.6.3 Search Space Characterization

The distribution of genotypes generated during the EMO search process is plotted in Figure 5.6 in terms of locomotion distance and number of hidden units used in the ANN. Except for NNType1, the distribution of solutions obtained across the solution space was less clustered for all other architectures using the EMO algorithm compared to random search, hill-climbing and random walk. This suggests that the EMO search process was able to sample more uniformly across both objective spaces in spite of having the added optimization criterion of minimizing the number of hidden units used in the ANN. In NNType1, a number of spikes in the frequency distribution could be seen (Figure 5.6.2). A separate graph is

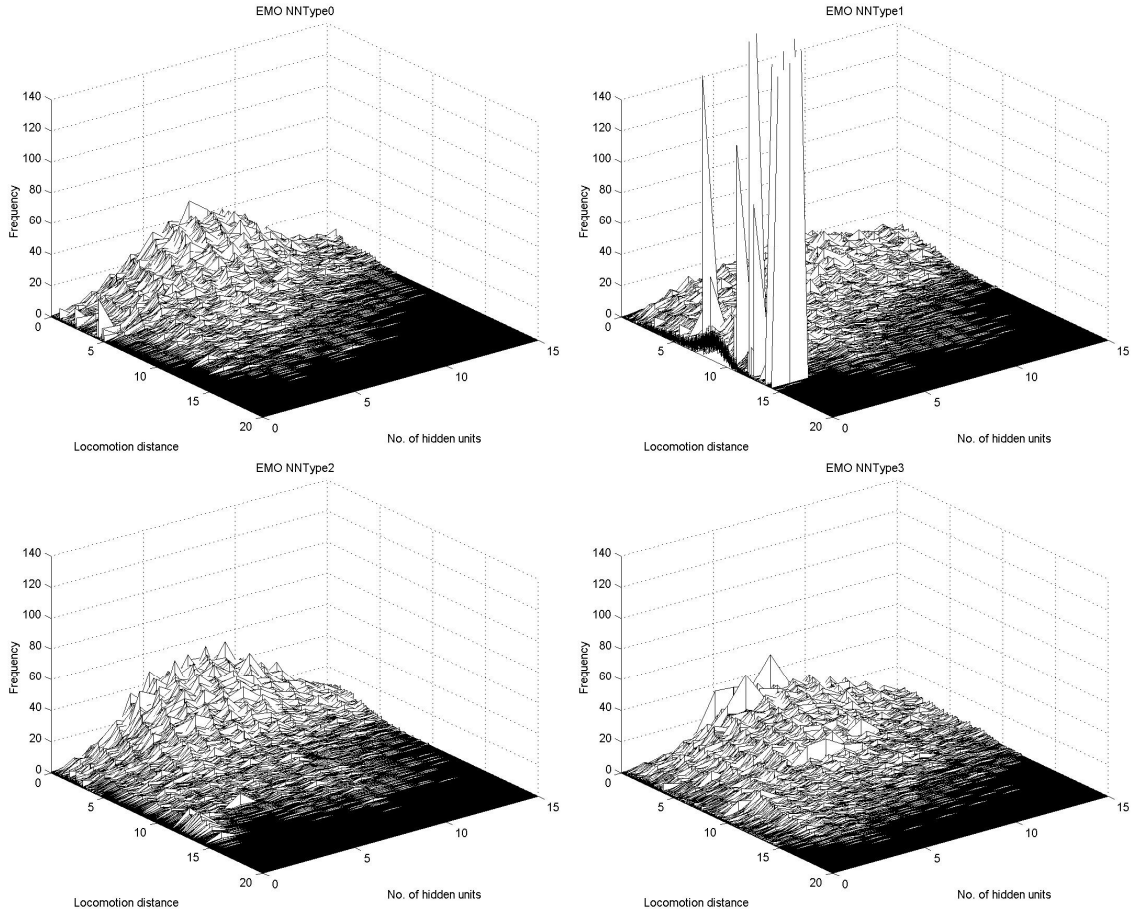

Figure 5.6: Frequency distribution of solutions using the SPANN algorithm for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Locomotion distance, Y-axis: No. of hidden units, Z-axis: Frequency.

plotted for this architecture in Figure 5.7. This occurrence is discussed further in the next paragraph where the concentration of solutions in terms of the two separate objectives is more evident in the accompanying contour graphs. Furthermore, for all four architectures, a significantly larger proportion of individuals were generated in the fitter regions of the locomotion objective space compared to random search, hill-climbing and random walk.

The contour graphs in Figure 5.8 illustrate the distribution of solutions across the two objectives of minimizing the hidden layer and maximizing locomotion

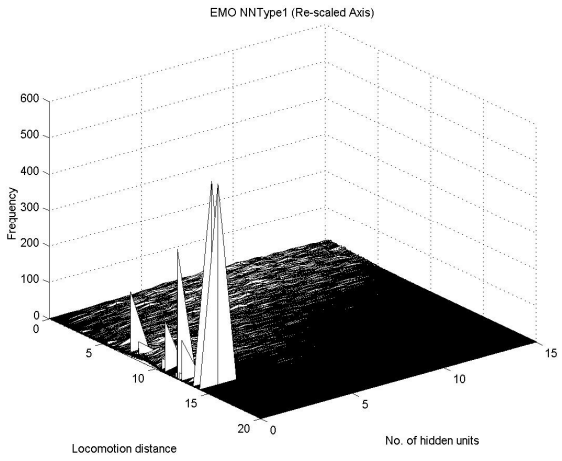

Figure 5.7: Frequency distribution of solutions obtained using the SPANN algorithm for ANN architecture NNType1 with re-scaled frequency axis. X-axis: Locomotion distance, Y-axis: No. of hidden units, Z-axis: Frequency.

distance. Compared to hill-climbing and random walk, the effect of having an extra objective in the form of minimizing the number of hidden units used in the ANN can be seen from the slight shift in distribution of solutions towards the lower halves of the graphs, which is very prominent in NNType3 (Figure 5.8.4) and most obvious in NNType1 (Figure 5.8.2). As was discussed previously, the presence of direct input-output connections allowed the search process to find effective locomotion controllers using a very small number of hidden units or none at all. For NNType1, the spikes highlighted in the previous paragraph can be seen in the high fitness regions of both objective spaces using only 0 or 1 hidden units and achieving between 9 and 15 units of locomotion distance.

The probability density function of solutions obtained using the SPANN algorithm is illustrated in Figure 5.9 for all four ANN architectures. The graphs clearly show that the probability of encountering fitter solutions in terms of locomotion distance was much higher than in all previous search algorithms. The probability density curves were similar to a large extent across all four architectures in terms of the locomotion fitness. From the cumulative curve, it can be seen that for NNType0 (Figure 5.9.1) and NNType2 (Figure 5.9.3), the probability

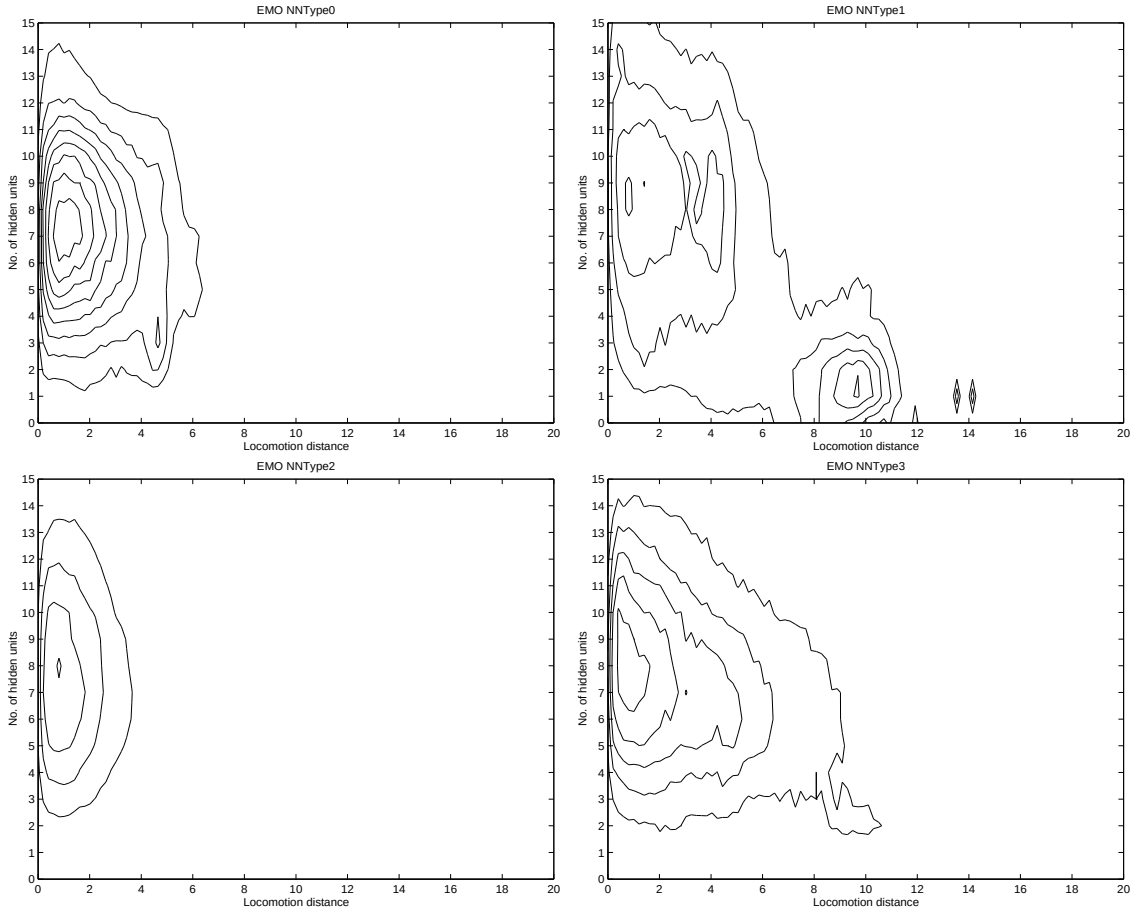

Figure 5.8: Contour graphs of frequency distribution of solutions obtained using the SPANN algorithm for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Locomotion distance, Y-axis: No. of hidden units.

of generating controllers approached 0 beyond a locomotion capability of 12 units whereas for NNType1 (Figure 5.9.2) and NNType3 (Figure 5.9.4), the probability only approached 0 beyond a locomotion capability of 14 units. This is another weak indication that the direct input-output connections provided an easier path to reach fitter controllers in terms of locomotion capability while the recurrent connections did not appear to provide any significant advantages over the simple feed-forward ANN architecture that had no extra connections between layers.

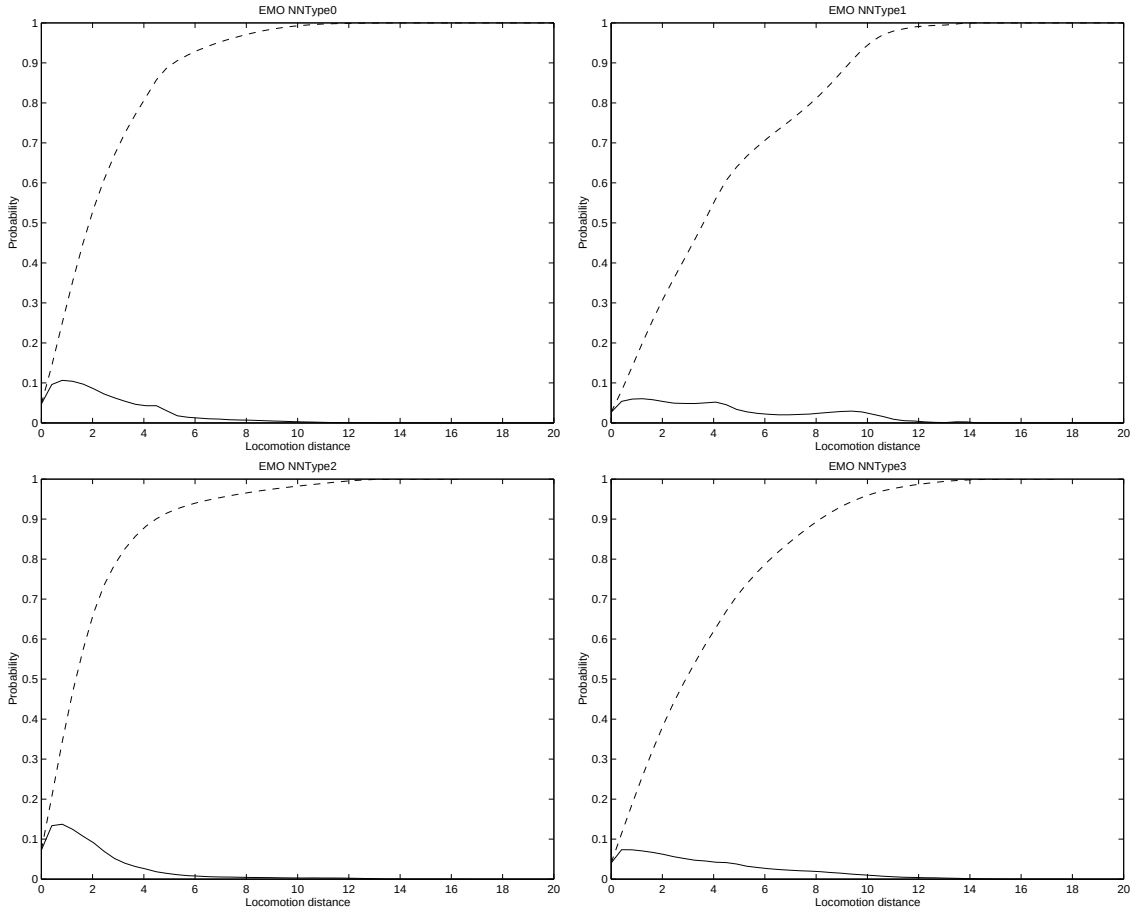

Figure 5.9: Density (solid) and cumulative (dashed) probability distribution of solutions obtained using the SPANN algorithm for ANN architecture 1. NNType0 (top left), 2. NNType1 (top right), 3. NNType2 (bottom left), 4. NNType3 (bottom right). X-axis: Locomotion distance, Y-axis: Probability.

5.7 Operational Dynamics

5.7.1 Behavior Inside and Outside Evolutionary Window

A top-down view of the artificial creature's path as controlled using the overall best ANN evolved for locomotion distance is plotted in Figure 5.10.1 for the actual period between 1–500th timestep where the fitness of the controller is being evaluated. A second graph of the artificial creature's path for the period between the 501–1000th timestep is plotted in Figure 5.10.2 to observe the locomotion be-

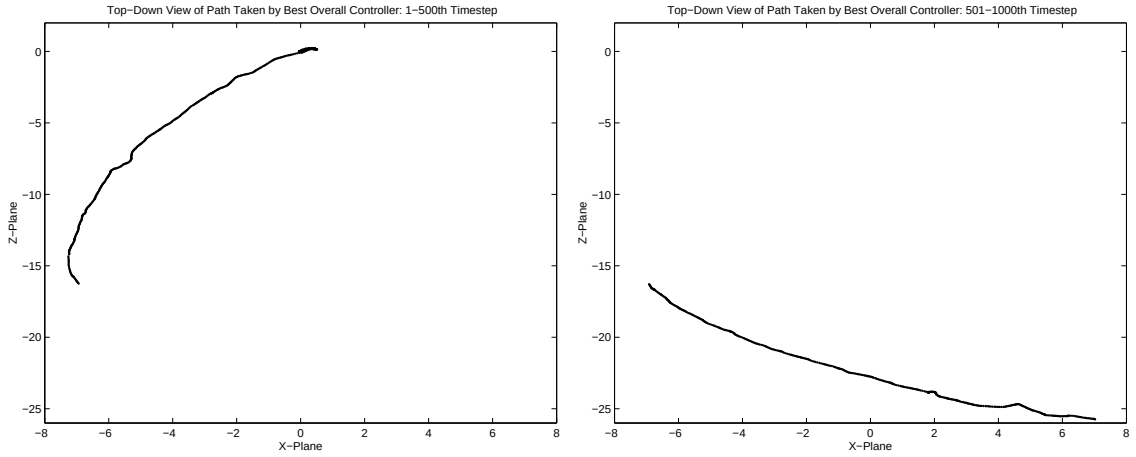

Figure 5.10: Path taken by artificial creature as controlled by overall best ANN evolved for locomotion distance using the SPANN algorithm — NNType3: 4 Hidden Units. 1. 1–500th timestep (left), 2. 501–1000th timestep (right) X-axis: X-Plane, Y-axis: Z-Plane (in Vortex, the Z-Plane is the Y-axis).

havior beyond the actual fitness evaluation window used during evolution. This controller has the NNType3 architecture and uses 4 hidden units. Although the path taken was not exactly a straight line, it did however maximize the horizontal distance moved fairly well. It begins from the origin at coordinates (0,0) and after 500 timesteps ends approximately at coordinates $(-6.9,-16.5)$. This emergent behavior is interesting in that although the initial setup has the creature's forwards orientation as being in the positive coordinate areas of the X-Plane and Z-Plane, the evolved locomotion behavior was in fact a backwards oriented walk if the initial positioning of the creature is taken as the reference frame. Visualization of the other global Pareto solutions revealed similar orientations for the evolved locomotion behaviors (interested readers can view video clips of these evolved behaviors in the accompanying CD-ROM). This can be explained by the fact that all the global Pareto solutions using the NNType3 architecture were actually obtained from one run using a particular seed. Across the global Pareto solutions obtained with other architectures using SPANN as well as other algorithms, the majority of the evolved controllers produced movement in the forwards direction rather than this backwards

movement. The design of the creature's limbs and in particular how the joint constraints were set up allowed for both forwards and backwards oriented walks to be evolved. Note that the initial movement seen in the other direction from coordinates (0,0) to approximately (0.5,0) occurred during the standing up phase of the creature's locomotion. Once the creature stood up, it then started the backwards oriented walk to achieve the maximal horizontal distance moved.

Towards the end of the walk, the path could be seen to start curving back towards the X-Plane. The continuation of this peculiar behavior beyond 500 timesteps as controlled by this evolved ANN can be seen in the plot of the path in Figure 5.10.2. Nonetheless, the creature was still able to walk in a fairly straight line thereby achieving a reasonably maximal locomotion distance during this next 500 timesteps. If the path of the creature is considered over the entire 1-1000th timestep, what this analysis shows is that the operational dynamics of the evolved behavior during the period which the controller was actually evolved to perform can be quite different to the operational dynamics when used beyond its evolutionary design period. This phenomenon relates back to what was highlighted by Ronald and Sipper (2001) in that the use of biologically-inspired solutions in engineering problems may be problematic because unexpected and sometimes unwanted results or behaviors might arise (discussed in the last paragraph of Section 2.3).

5.7.2 Limb Dynamics

The outputs generated from the operation of the overall best controller evolved for locomotion distance to Actuators y_1–y_8 are plotted in Figure 5.11. In all except one of the outputs, sine-like wave signals were generated by the evolved ANN to the motors in the respective limb actuators. This is consistent with the evolved walking behavior which is cyclic in nature.

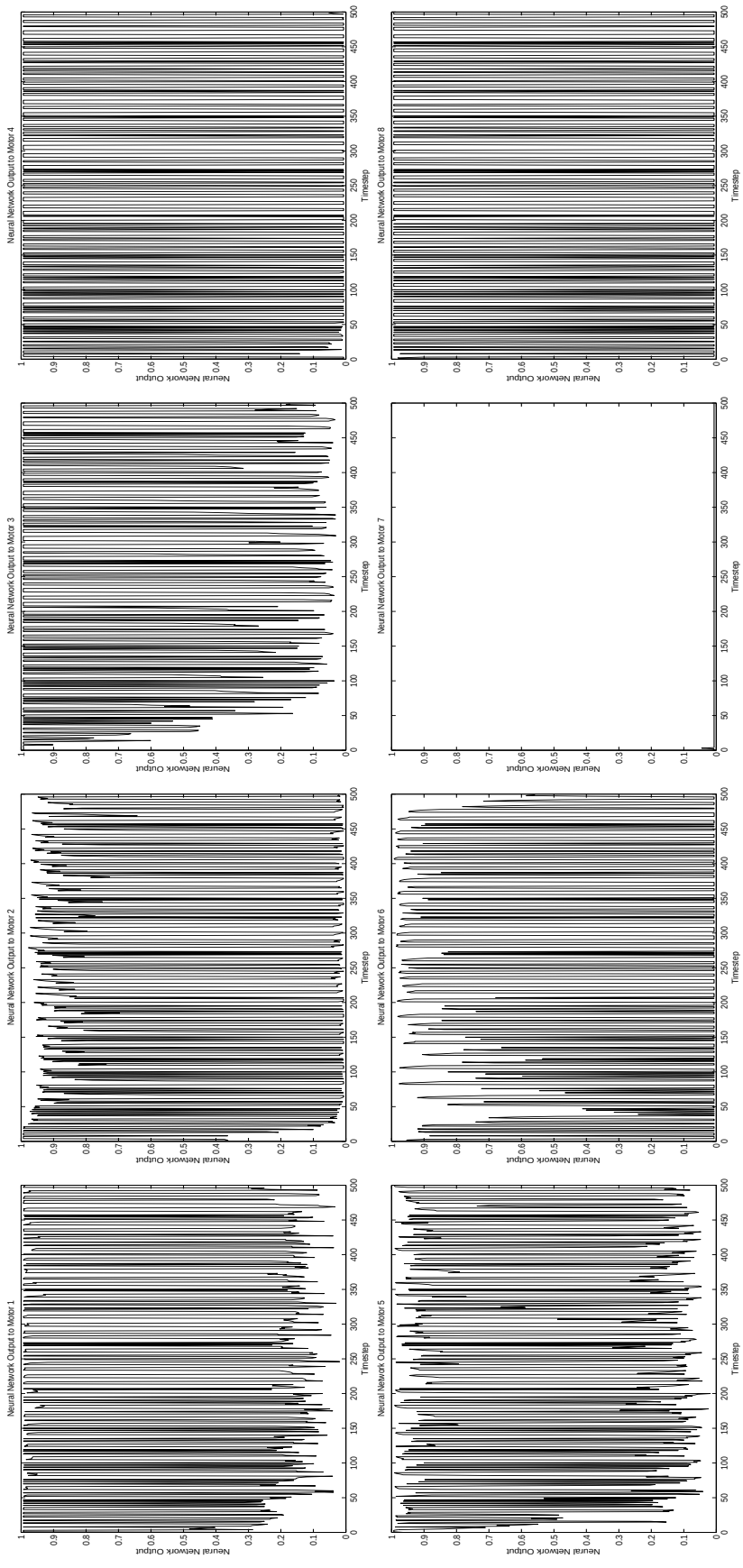

Figure 5.11: The outputs from the overall best evolved ANN controller to Actuator 1. y_1 (top extreme left), 2. y_2 (top middle left), 3. y_3 (top middle right), 4. y_4 (top extreme right), 5. y_5 (bottom extreme left), 6. y_6 (bottom middle left), 7. y_7 (bottom middle right), 8. y_8 (bottom extreme right). X-axis: Timestep, Y-axis: Neural network output.

It is also interesting to note that the signals generated were quite distinctive over time as near maximal outputs of either 0 and 1 at the peaks and troughs of the cycle were being generated by the ANN. This behavior is close to the optimal control strategy known as "Bang-Bang Controls", which take their values on the extreme points (in our case, 0 and 1) (Macki and Strauss 1982). In terms of the single output which generated a totally flat signal of practically 0 magnitude over time (Figure 5.11.7), this indicated the presence of a passively-controlled lower limb, which obtained its swinging motion from the movement of the attached upper limb. A visual inspection of the creature in simulation confirmed that this limb did in fact exhibit some dynamical behavior during locomotion. This suggests that the evolutionary search found a simpler control solution through the use of a passive dynamic limb (McGeer 1990).

Limbs	Correlation Coefficient
Upper and Lower Back Left	0.9410
Upper and Lower Front Left	−0.9648
Upper and Lower Back Right	0.0376
Upper and Lower Front Right	−0.9998

Table 5.6: Correlation coefficients for neural network outputs between the upper and lower limbs of each leg.

Another interesting dynamical behavior that emerged from the outputs of the ANN is that the component limbs in all of the legs learnt to coordinate and synchronize their movements within each leg, with the exception of the leg containing the passive dynamic limb. This is evidenced by the very high correlation between the upper and lower limbs of the back left, front left and front right legs as shown in Table 5.6. For a legged gait with good locomotion capabilities, the constituent components in each leg would be expected to function as a cohesive unit in order for each leg to generate useful movements for locomotion. The outputs to the individual limb actuators for the back left leg have evolved to be almost entirely in-phase, as evidenced by the very high positive correlation coefficient (0.9419). On the other hand, the outputs to the individual limb actuators for the front left and front right legs have evolved to be almost entirely out-of-phase, as evidenced by the very high

negative correlation coefficients (-0.9648 & -0.9998). This shows that the neural network controller has learnt to coordinate and synchronize between the limbs of each leg, thereby allowing for successful locomotion to occur.

5.7.3 Effects of Noise

In this section, we investigate the effects of noise on the performance of the overall best evolved controller for locomotion distance obtained from SPANN. Seven different levels of noise were applied to the joint angle sensors, touch sensors and outputs to the actuators individually as well as in combination for all three elements. Random noise levels ranging from 1% to 50% of the individual ranges of values for these sensors and actuators (see Section 3.2) were applied.

Noise Level	Locomotion Distance with Noise in Joint Angle Sensors	Locomotion Distance with Noise in Touch Sensors	Locomotion Distance with Noise in Actuators	Locomotion Distance with Noise in All Sensors and Actuators
1%	13.4402 ± 1.7281	14.3599 ± 0.9073	13.8640 ± 0.9970	13.8147 ± 1.2219
5%	13.2121 ± 2.0271	12.5348 ± 1.0275	13.2310 ± 0.7213	11.2828 ± 0.8163
10%	11.8406 ± 1.2960	10.6214 ± 1.3914	12.9669 ± 1.2229	9.5492 ± 0.5552
20%	8.7969 ± 0.8017	6.3701 ± 1.9298	10.8933 ± 2.5271	4.8763 ± 0.4763
30%	6.8828 ± 0.4429	3.5598 ± 1.2999	8.8441 ± 1.9260	2.6185 ± 1.0707
40%	6.6890 ± 0.6028	1.6049 ± 1.1109	6.7339 ± 1.7209	1.2486 ± 0.6111
50%	6.7256 ± 0.9367	1.3050 ± 1.0950	3.7132 ± 1.9092	1.2153 ± 0.4210

Table 5.7: Comparison of average locomotion distance achieved over 10 runs by overall best controller evolved for locomotion distance using the SPANN algorithm with varying noise levels in the sensors and actuators.

Table 5.7 lists the average and standard deviation of locomotion distances achieved by the overall best locomotion controller from SPANN with varying levels of noise applied to the sensors and actuators of the artificial creature. The performance of the controller degraded monotonically in all cases as the level of noise was increased from 1% to 50%, except for 50% noise in the joint angle sensor. At the lowest level of noise of 1%, the least significantly affected component was the touch sensor which still achieved on average 81.1% of the original locomotion distance.

However, as the noise level was increased, the touch sensors seemed to be most affected by the presence of noise. In all cases where noise ranging from 5% to 50% was applied to the individual components, the lowest average locomotion distance was obtained when there was the presence of noise in the touch sensors. When noise was introduced to all sensors and actuators of the artificial creature in combination, the performance was lower than in all cases where noise was applied to each individual component except for 1% noise in the joint angle sensors. A visual inspection of the artificial creature in simulation with 10% random noise added to the sensors and actuators both individually and in combination revealed that the major characteristics of the locomotion behavior was still present, such as the backwards oriented walk and general movement of the limbs (interested readers can view video clips of a sample of these behaviors in the accompanying CD-ROM). Therefore, the evolved controller was still able to perform reasonably well with low levels of noise present in the sensors and actuators of the artificial creature.

5.8 Advantages of Pareto EMO

From these experiments, it can be seen that the most significant advantage in using a Pareto EMO approach for studying the evolution of artificial creatures as proposed in our SPANN algorithm is that a variety of controllers can be generated in a single evolutionary run without requiring any further modification of parameters by the user. This means that an entire set of controllers with varying network sizes and locomotion capabilities can be generated at once, allowing for comparisons between creatures with different abilities and controllers to be made after just a single run is conducted for each type of creature. As such, the Pareto EMO approach provides a flexible and convenient platform for conducting investigations into the evolution of artificial creatures. This represents a significant advantage over single-objective evolutionary systems that need to be re-run multiple times in order to test the effect of other factors such as number of hidden units on the locomotion capability of the artificial creatures (Bongard and Pfeifer 2002). Such a setup would require a

significantly larger number of evolutionary runs before a suitable set of controllers with different network characteristics and locomotion capabilities can be obtained in order to conduct comparisons between different creature designs. An alternative method would be to re-formulate the problem by taking a weighted sum of the two objectives into a single objective. However, there are a number of drawbacks in using a weighted sum methodology, which we discuss in detail later in Section 6.4.2. In short, the Pareto EMO paradigm allows the user the option to conveniently choose from a variety of controllers with varying architectural complexities and locomotion competencies to suit the eventual simulation environment, constraints and purposes.

A further advantage of using a Pareto multi-objective approach for artificial evolution is that genetic diversity is maintained naturally during the course of the evolutionary process (Abbass and Deb 2003). A common problem with evolutionary optimization algorithms is premature convergence due to loss of genetic diversity and this phenomenon has been observed to cause problems in the artificial evolution of virtual creatures as well (Komosinski and Rotaru-Varga 2001). In a simple artificial life ecosystem, mutualism (Pachepsky, Taylor, and Jones 2002) was proposed as a method for promoting genetic diversity and was shown to improve evolvability as well as population stability in the artificial evolutionary system. Here, we propose an evolutionary multi-objective algorithm that promotes reproductive diversity by allowing the evolutionary process to optimize along two separate and distinct goals of minimizing network size while maximizing locomotion ability. This type of evolutionary optimization algorithm therefore fits well into the scheme of creating artificial creatures. In this case, evolutionary creativity should not be stifled by the optimization process but should instead be encouraged if interesting and diverse creatures are to emerge from the process.

The use of EMO also opens up the possibility of creating extra-dimensional bypasses through the search space that provide an easier path for the optimization process to reach fitter regions of the solution space. This phenomenon has been previously encountered in conventional single-objective evolutionary optimization systems through the addition of extra genotypic parameters into the artificial evo-

lutionary system (Bongard and Paul 2001). However, such artificial methods of increasing the search space is rather subjective and may require significant experimentation in order to find the right parameters that can create the extra-dimensional bypass. In our proposed methodology, this is achieved in a natural manner through the use of multiple optimization objectives. Here, we do not make the claim that such extra-dimensional bypasses will be present under all EMO runs but simply that the use of multiple objectives in an artificial evolutionary system allows for the possibility of such bypasses to emerge through the inherently more complex interactions between genes under multiple evolutionary pressures. The presence of high epistatis is evidenced by the highly rugged fitness landscape areas present in certain sub-regions of the search space as discussed in Section 4.5.3.1.

5.9 Chapter Summary

An investigation into the use of multi-objective evolutionary optimization for automatic synthesis of ANN controllers that are proficient at generating locomotion capabilities in a physically simulated quadruped yielded the following results:

- An EMO algorithm called SPANN was implemented for the multi-objective evolution of artificial creature controllers. ANN controllers based on four different types of underlying architecture were successfully evolved for maximum horizontal locomotion capability and minimum usage of number of hidden units in the ANN.

- An EMO algorithm provides significant advantages over conventional single-objective optimization algorithms by: (1) reducing the number of runs required to test different design factors associated with the synthesis of artificial creatures, (2) preserving genetic diversity and, (3) offering extra-dimensional bypasses for the search process to reach fitter solution spaces.

- Additional recurrent connections do not provide any significant advantages over conventional feed-forward neural network architectures for evolving loco-

motion capability in a four-legged artificial creature.

- Pure reactive agents not requiring hidden layer transformations in the ANN controller produced sufficiently good locomotion capabilities. The use of direct input-output connections in a perceptron-like controller was sufficient for generating a basic locomotion ability in a four-legged artificial creature. The use of a hidden layer was not required to synthesize a controller with a comparatively good locomotion capability.

- An operational dynamics analysis revealed that the ANN controller learnt to coordinate and synchronize between the upper and lower limbs in three of the simulated quadruped's legs. The presence of a passively-controlled dynamic limb was observed in the remaining leg. The ANN controller still performed well when a reasonable level of noise was present in the sensors and actuators.

The design, implementation and use of an EMO algorithm called SPANN for the evolution of artificial creature controllers has been presented in this chapter. ANN controllers were successfully evolved for minimum hidden layer size and maximum horizontal locomotion distance using four different types of ANN architecture. In the next chapter, we will compare the SPANN algorithm against more conventional methods of evolutionary optimization to verify that this approach is actually beneficial for evolving artificial creature controllers.