Proceedings of the 2007 IEEE Symposium on Artificial Life (CI-ALife 2007)

Epistasis in Multi-Objective Evolutionary Recurrent Neuro-Controllers

Mario Ventresca and Beatrice Ombuki-Berman

***Abstract*— This paper presents an information-theoretic analysis of the epistatic effects present in evolving recurrent neural networks. That is, how do the gene-gene interactions change as the evolutionary process progresses from an initially random state to the final generation and does this reveal anything about the problem difficulty. Also, to what extent does the environment influence epistasis. Our investigation concentrates on multi-objective evolution, where the major task to be performed is broken into sub-tasks which are then used as our objectives. Our results show that the behavior of epistasis during the evolutionary process is strongly dependant on the environment. The experimental results are presented for the path following robot application using continuous-time and spiking neuro-controllers.**

***Index Terms*— Epistasis, spiking, continuous-time, recurrent neural network, multi-objective, evolutionary algorithm.**

I. Introduction

VARIOUS neural network models have been successfully utilized as controllers for autonomous robots. However, due to their connectivity, recurrent neural network weights are inherently dependant on each other to achieve the desired output. In a neuro-evolutionary scenario, network weights and other free parameters are randomly initialized and become specialized as the generations pass. One goal of this work is to investigate how the problem difficulty changes with respect to a change in network weights over the generations. Additionally, we examine how the environment can influence these epistatic interactions by fixing the state of the environment over all experiments (to focus on its effect). This paper provides some insight into these questions from the standpoint of epistasis.

With respect to evolutionary algorithms, epistasis refers to the influence of interacting genes on the overall fitness of the genotype [1]. As such, epistasis plays an important role on the search difficulty of a problem. Generally, higher degrees of epistasis imply a harder search. In many evolutionary robotics applications neural networks are encoded into chromosomes so that they may be evolved. Therefore, the weight dependence as the network evolves will be visible as epistatic interactions within the chromosomes. So, we may apply epistatic metrics to evaluate the difficulty of searching for neuro-controllers that can accomplish the given task.

A popular choice of neuro-controller is based on continuous-time [2] or spiking recurrent models [3]. These models are capable of modelling temporal effects and thus do not exhibit purely reactive behaviors. That is, they also take recent actions into account before acting on the current input signal. We will concentrate on these models in this paper.

Recently, multi-objective techniques such as Pareto ranking [4] have been utilized in the evolutionary robotics and embodied cognition communities. For example, [5] and [6] utilize this technique to simultaneously evolve the neuro-controller and morphology of multi-legged robots. The goal is to minimize the controller size while maximizing the robots locomotion distance. This paper differs in that we maintain a fixed neural architecture, and instead investigate multiple sub-objectives of a larger goal. For example, in order for a robot to quickly traverse through a maze it must satisfy the sub-objectives of forward motion, relatively fast speed and avoidance of obstacles.

The remainder of this paper is as follows: Section II briefly describes the Pareto ranking strategy, Section III will discuss the information-theoretic measures of epistasis. Section IV introduces continuous-time and spiking recurrent neural networks. Our experimental setup is outlined in Section V and the results are presented in Section VI. Our conclusions and directions for future research are given in Section VII.

II. Multi-Objective Evolution

A multi-objective problem is one where the solution is composed of two or more objectives or goals which may interact in a constructive or destructive manner. While various approaches exist to incorporate multi-objectivity into evolutionary algorithms, we will utilize the Pareto ranking [4] strategy.

The underlying idea behind the Pareto ranking fitness evaluation strategy is to preserve the independence of the individual objectives. This is accomplished by stratifying the current population of solutions into ranks whereby lower ranks store more desirable solutions. In order for a solution to occupy a lower rank it must be clearly superior (i.e., non-dominated) to the others in all objectives of the problem. Solutions that occupy the same rank are considered indistinguishable from each other. A definition of the notion of dominance as given by [7] for any two solutions G_1 and G_2 is:

1) The solution G_1 is no worse than G_2 in all objectives, mathematically stated as $f_j(G_1) \not\rhd f_j(G_2)\ \forall$ objectives $j = 1, 2, ...M$, where f_j represents the fitness of the j^{th} objective.
2) The solution G_1 is strictly better than G_2 in at least one objective, mathematically, $\exists j \in 1, 2.., M$ such that $f_j(G_1) \lhd f_j(G_2)$.

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

M. Ventresca, Department of Systems Design Engineering, University of Waterloo, Waterloo, ONT, Canada (email: mventres@pami.uwaterloo.ca)

Dr. B. Ombuki-Berman, Computer Science Department, Brock University, St. Catharines, ONT, Canada (email: bombuki@brocku.ca)

If these conditions are satisfied, then it is said that S_1 dominates S_2 and S_1 must occupy a lower rank. Conversely, if these conditions are not satisfied then S_1 does not dominate S_2. Those solutions in the lowest rank are said to be non-dominated.

III. Information-Theoretic Epistasis Metrics

In order to perform our analysis we utilize the information-theoretic measures described below (as proposed by Seo *et al* [8]). These metrics quantify the significance of a single gene, epistasis between pairs of genes and for the whole problem. A more detailed description of the required computations are presented in Appendix A.

A. Gene Significance

This measure aims to quantify the contribution of each of the n genes of S_i for $i = 1, ..., n$ to the fitness of the solution $f(S)$. This is interpreted as the amount of information $I(S_i; f(S))$ that the i^{th} gene of a solution reveals about the fitness and is calculated by

$$\xi_i = \frac{I(S_i; f(S))}{H(f(S))}. \tag{1}$$

The denominator $H(f(S))$ calculates the entropy over the marginal distribution of possible fitness values of $f(S)$ and scales ξ_i to a range of $[0, 1]$. A gene significance value close to 0 implies that the gene contributes very little to the fitness. Conversely, a value close to 1 indicates that the fitness is highly influenced by the gene.

If a single gene S_k has a value near 1 and all others are all close to 0, then the other genes may be not needed. This is because the fitness is very dependant on the value of S_k. It may be possible to use this measure to shorten the solution representation, although not done so here.

B. Gene Epistasis

While the gene significance is concerned with an individual gene's contribution to the overall fitness, the gene epistasis measure concentrates on pairs of genes. This is interpreted as the amount of information $I(S_i, S_j; f(S))$ that any two genes S_i and S_j reveal about the fitness of S, where

$$\begin{aligned} I(S_i, S_j; f(S)) &= \sum_{i=1}^{n} I(S_i; f(S)|S_1 ... S_{i-1}) \\ &= I(S_1; f(S)) + I(S_2; f(S)|S_1), \end{aligned}$$

where this value is calculated for all $i \neq j$. Then, the amount of epistasis between the two genes is calculated by

$$\varepsilon_{ij} = \begin{cases} 1 - \frac{I(S_i;f(S)) + I(S_j;f(S))}{I(S_i,S_j;f(S))}, & \text{if } I(S_i, S_j; f(S)) \neq 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

This value exists on the range $[-1, 1]$. When $\varepsilon_{ij} > 0$ it is said that the genes S_i and S_j exhibit a constructive epistatic relationship, meaning they interact constructively with each other. On the other hand if $\varepsilon < 0$ the genes interact in a destructive manner. When the gene epistasis is 0 the two genes are mutually independent.

C. Problem Epistasis

As a measure of the epistasis of the problem itself, the sum of the absolute value of all the gene epistasis values is used. The problem epistasis η is then calculated by equation 3.

$$\eta = \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j<i} |\varepsilon_{ij}|. \tag{3}$$

This value is bounded below by 0 and 1 from below and above, respectively. Values of η closer to 1 indicate a highly epistatic problem.

IV. Neural Models

Our experiments will concentrate on two recurrent neural network models, namely continuous-time [9] and spiking [3]. The main difference between the two models lies in the method in which data is encoded and signals are transmitted between neurons. Spiking networks use a spike-encoding, whereas continuous-time models use a rate-encoding scheme which approximates a spike-encoding [3]. Nevertheless, both models have been successfully utilized in evolutionary systems for adaptive behavior tasks [2] and [10].

A. Continuous-time Model

Continuous-time recurrent neural networks (CTRNN) are composed of dynamical neurons, whereby the rate at which their activation changes is dependant on a time constant $\tau > 0$. Furthermore, each neuron acts as a leaky integrator, such that input increases the neuron's action potential which slowly degrades over time. In this manner, a neuron is influenced by its previous state(s). The state of a neuron is determined by both its previous activation and its current output, which is dependant on the type of activation function. The purpose of the time constant τ_i is to determine the rate of change of the neuron's state. This process is summarized as

$$\frac{d\nu_i}{dt} = \frac{1}{\tau_i} \left(-\nu_i + \sum_{j=1}^{N} w_{ij} \sigma(\nu_j + \theta_j) + I_i \right) \tag{4}$$

where ν_i represents the activation potential for the i^{th} neuron. The activation function $\sigma(\cdot)$ takes each j^{th} incoming signal and its bias θ_j as input. Additionally, real-valued external input I_i can be provided to the neuron as well. For this work we utilize the common logistic function (5) as each neuron's activation.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

It has been shown by Funahashi and Nakamura [11] that CTRNNs are universal approximators of any real-valued function.

B. Spiking Model

Spiking recurrent neural networks (SRNN) attempt to utilize a more biologically realistic data encoding and signal transmission approach [3]. It has been argued that these types of networks are better suited for applications where the timing of signals is important [12]. It has been mathematically shown by Maass [13] that networks of spiking neurons have considerably more computing processing power than similarly sized non-spiking networks. However, this does not imply that non-spiking models are obsolete.

Since their inception many variations on spiking neuron models have been proposed but we have focus on the Spike Response Model [14]. According to this model, a neuron is solely represented by its membrane potential ν_i at time t. When the membrane potential reaches the neurons threshold level θ, the neuron will fire, represented as t_i^f (where f refers to the fact the neuron fired). The set of all n firing times of the i^{th} neuron is defined as its spike train, represented as

$$F_i = \{t_i^f : 1 \leq f \leq n\} = \{t | \nu_i(t) = \theta\}. \tag{6}$$

The actual spike is a function of the synaptic delay Δ, synaptic time constant τ_s, and the membrane time constant τ_m. Here we model each spike according to equation (7), where adjusting the respective constants causes the shape of the function to change (where $s = t - t^f$).

$$\varepsilon(s) = \begin{cases} e^{\frac{s-\Delta}{\tau_m}} \cdot (1 - e^{\frac{s-\Delta}{\tau_s}}), & \text{if s} \geq \Delta \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

A plot of this function is shown in Figure 1 for $\Delta = 1$, $\tau_s = 10$ and $\tau_m = 4$.

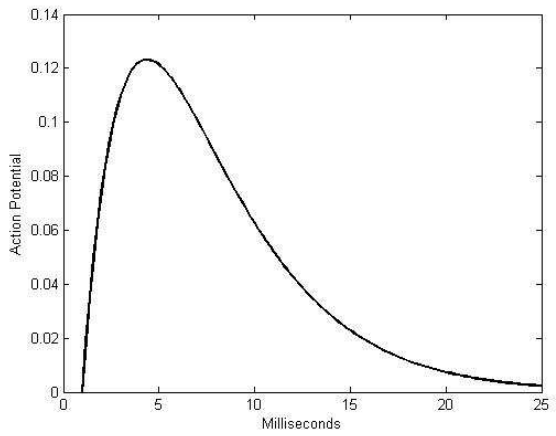

Fig. 1: The behavior of an action potential.

After emitting a spike, the neuron enters the absolute refractory period, which lasts for Ω time units. During this time it is impossible for the neuron to evoke a spike. After this period, the neuron enters a state of relative refractoriness during which it becomes increasingly likely that it can fire. A common method to model this behavior is presented in equation (8).

$$\eta(s) = \begin{cases} -\theta e^{\frac{-s}{\tau_m}}, & \text{if s} > \Omega \\ -\infty, & \text{otherwise} \end{cases} \tag{8}$$

where $s = (t - t^f)$ represents the difference between the current and spike times, respectively, while τ_m is a membrane time constant. Figure 2 shows a plot of the refractory periods.

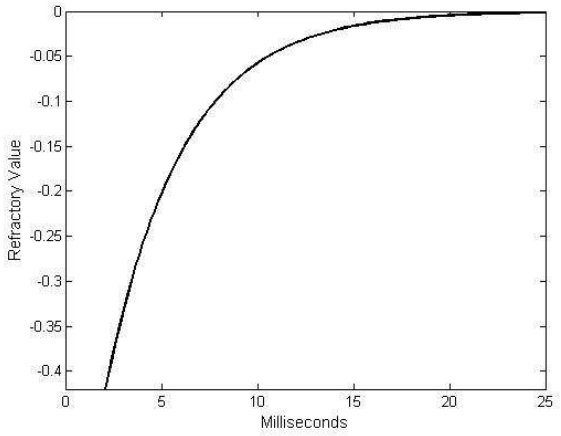

Fig. 2: The absolute and relative refractory periods.

By combining equations (7) and (8) we can describe the dynamics of the i^{th} neuron having several incoming connections. Each of the incoming signals is given a weight $w_j \in \Re$. Thus, the membrane potential can be described as

$$\nu_i(t) = \sum_j w_j^t \sum_{f \in F_i} \varepsilon_j(s_j) + \sum_{f \in F_i} \eta_i(s_i). \tag{9}$$

So, if $\nu_i \geq \theta$ the neuron will evoke a spike and the neuron will enter the refractory stages. In this paper, we allow each k^{th} synapse to have its own time constant, τ_{s_k} as well as its own delay Δ_k. Similarly, each n^{th} neuron can have its own membrane time constant τ_{m_n}, although all neurons have a fixed firing threshold $\theta = 0.7$ (empirically decided).

V. Experimental Setup

In order to run any experiments we must address the genetic representation and search operator for the neural network models. Additionally, the application being utilized is described as is the method for gathering the probabilities used for calculating epistasis.

A. Genetic Encoding

The continuous-time recurrent neural networks were directly encoded into a set of three matrices representing connection weights $w_{ij} \in [-5, 5]$ between neurons i and j, the time constants $\tau_i \in [1, 50]$ and biases $\theta \in [-1, 1]$. Similarly, spiking networks were encoded into three matrices which represent the connection weights $w_{ij} \in [-5, 5]$, axonal delay $\Delta \in [1, 5]$ and the synaptic and membrane potentials τ_s, $\tau_m \in [1, 10]$ such that for each i^{th} neuron, $\tau_{m_i} > \tau_{s_{ij}} \forall j$. So, each matrix entry represents a gene of the encoding. The above respective intervals have been empirically determined.

B. Evolutionary Operator

In order to adapt the CTRNN and SRNN representations we utilize a mutation-only evolutionary algorithm. The operator we implemented adds a small random value to each variable in the network with a small probability that is inversely proportional to the total number of non-zero genes the respective variable, V. For example, V can be the matrix of weights in a SRNN, where $|V|$ is the total number of non-zero weights. This operator is summarized in equation (10), where $random$ is a uniform random number on $[0, 1]$, $v_i \in V$ and j is the current generation.

$$v_i^j = \begin{cases} v_i^{j-1} + \mathcal{U}(-0.5, 0.5), & \text{if random} < 1/|V| \\ v_i^{j-1}, & \text{otherwise} \end{cases} \quad (10)$$

This operator is applied to each of the matrices for each network at every generation of the algorithm. On average only 1 value per matrix will have the mutation operator applied to it, which is a result of the inverse proportionality.

C. Path-finding Robot

The path-finding robot is a common evolutionary-robotics benchmark application [15]. Given a map such as that in Figure 15, the goal is to design a robot capable of quickly navigating around it while maintaining a safe distance from the walls. The starting point of the robot is indicated by the gray circle. We have utilized the Wright State University Khepera Simulator [16], which is a Java-based program, as our simulating environment.

The robot receives input from 8 infrared sensors distributed around it, which sense the distance to objects. Each sensor reading is a value between 0 and 1, where smaller values indicate that an object is close. Movement is accomplished by the rotation of its two wheels. The wheel speeds are integer values between 0 and 10 units, although they are scaled during fitness evaluation to make calculations simpler.

Input to the CTRNN neuro-controller will simply be the 8 raw sensor readings. The output neuron activation potential will then be the rotation speed of the left and right wheels, respectively. However, the SRNN cannot use the raw sensor readings. Instead, these readings must be transformed into a series of spikes. We have allowed each of the input neurons to emit a spike for every tenth of sensor input greater than a value of 0.4. Due to the non-negativity of spike times, the SRNN-controlled robot cannot move its wheels backwards.

In order to accomplish the task we use the objectives given below, as described in [15] (although, in that work these objectives are combined into a single value). For each of the objectives O_i the wheel rotation speed range is mapped to [-0.5, 0.5] and each $O_i \in [0, 1]$. First, to encourage fast movement through the maze, the absolute value of the left (L) and right (R) wheel speeds are summed together, $O_1 = |L| + |R|$. Higher values of this objective are desirable.

Forward motion is abetted by transforming the wheel speeds to the range [0,1] by adding 0.5 to each speed and then taking the absolute value of their difference. This value will be denoted as Δv. We then subtract the square root of Δv from 1 in order to transform it to a maximization problem. So, this objective is computed by $O_2 = (1 - \sqrt{\Delta v})$.

The final objective is to ensure that the robot does not hit any walls. This is accomplished by calculating $O_3 = (1 - j)$, where j is the smallest sensor reading, which indicates the distance to the closest object.

D. Probability Model

In order to calculate the three epistasis metrics described in Section III the probability model must be defined. Since connection weights for both neural models are real-valued and the epistasis metrics expect discrete variables we must perform a discretization of the weights and output. This is accomplished by dividing the respective variable ranges into 10 equal intervals. For example, the weight interval [-5,5] is divided into {[-5,-4),[-4,-3)...,[4,5]}. Therefore, each weight variable takes on one of these ten possible states. Similarly, each of the three objectives of a solution is divided into 10 equally sized partitions, {[0,.1),...,[.9,1]}.

The probabilities themselves are based on the current population of solutions. At each generation we estimate the $p(S_i, f(S))$, $p(S_i, S_j)$ and $p(S_i, S_j, f(S))$ joint distributions and the $p(S_i)$ and $p(f(S))$ marginal distributions from the current population of individuals. Then, we can compute the epistasis measures as outlined in Section III.

VI. Experimental Results

We present the experimental results as averages over 10 runs of 30 controllers per neuro-controller and each run lasted for 40 epochs. The selection strategy used was a 2-way tournament selection with a selection pressure of 0.75, as described in [17]. For each run the Pareto optimal set contained 3 elements on average.

During each fitness evaluation the Khepera robot was given 600 steps. The fitness is then evaluated as an average per step, with respect to each objective value. Almost every experiment was successful at evolving a neuro-controller capable of quickly traversing fully through the maze starting from its fixed starting position (1 unsuccessful CTRNN, 0 SNN). Each of the objectives outlined in Section V will be referred to as: Fast (refers to wheel speeds), Forward (forward movement), Avoid (obstacle avoidance).

It should be emphasized that the results given below represent the population as a whole. That is, we are not concerned with the best individual or its efficacy at solving the given task. Rather, we investigate how the *problem difficulty* changes over the entire run, and also the environmental influence for the entire population. Furthermore, the path following problem has been specifically chosen for its simplicity and also we disregard generalization since it would become cumbersome to analyze the environmental effects.

A. CTRNN Results

This section will describe the experimental results for the evolution of a CTRNN. Figure 3 shows the evolution of each of the three objectives, where convergence occurs after about

15 generations. Since each objective shows an increase, it implies that they are very dependant on one another to achieve a neuro-controller with high fitness. The obstacle avoidance and forward motion objectives show a strong similarity in behavior, that is, the robot is very good at accomplishing these objectives.

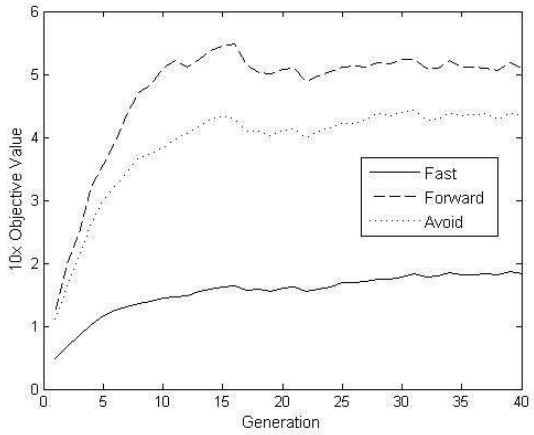

Fig. 3: Average fitness of the population for each objective.

The average amount of gene significance with respect to each objective over the evolutionary run is presented in Figure 4. During the initial generations we observe a relatively large value which indicates that the fitness of the neuro-controllers are very dependant on a subset of genes. Since the initial weights are randomly initialized, this corresponds to the entire population of controllers having a similar fitness. As new behaviors (i.e. ability to turn left) are evolved this value will decrease since the responsibility control is spread over the entire architecture. Additionally, the permutation problem [18] also influences the results, since the epistasis metrics do not take it into account.

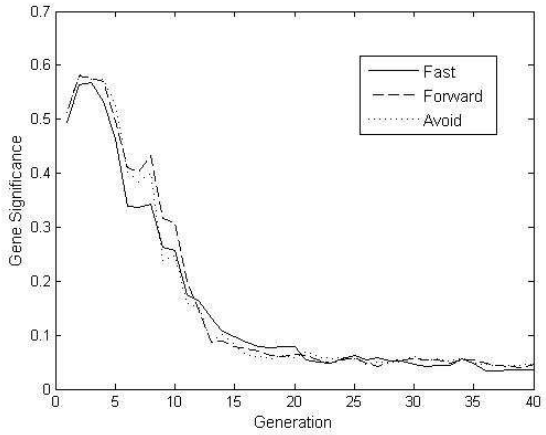

Fig. 4: Average amount of gene significance for the population.

Figure 5 plots the results of the average gene epistasis measure over the entire run. During the initial three generations moving fast and obstacle avoidance show a decrease in value, whereas moving forward is the main goal. Once it can move forward (opposed to a circular motion), then turning left and avoiding the wall is the next required behavior, followed by turning to the right.

With respect to each objective, the overall gene epistasis exhibits a destructive relationship which tends towards zero epistasis. The initial generations show a destructive interaction to a degree of about -0.30. Thus, the weights are having a detrimental effect on each other and should make the problem harder. Continuing with the evolution, this value tends towards zero indicating that the genes become nearly independent on each other, as expected.

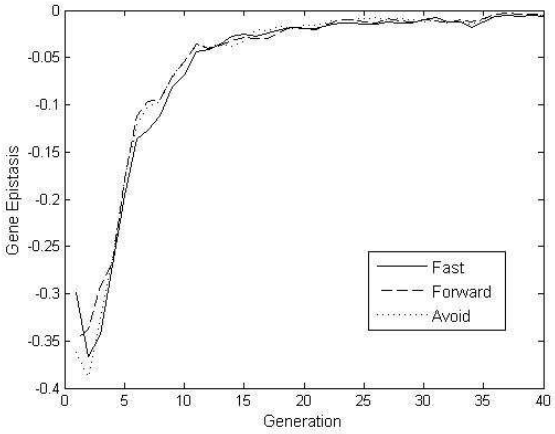

Fig. 5: Average number of constructive epistatic effects for the population.

The amount of constructive (gene epistasis values > 0.5) and destructive (values < -0.5) weights are shown in Figures 6 and 7, respectively. We show that there are very few constructive interactions, the maximum of 4 occurs at generation six (from Figure 6). The number of destructive interactions reaches a maximum of nearly 500 (about generation 2), however by the end of the run is near one. This implies that most gene interactions do not contribute in a very constructive or destructive manner. Thus, this should not be a very difficult problem.

Figure 8 confirms that according to the problem epistasis measure, this problem should not be very difficult since the amount of epistasis is less than 0.1. Therefore, evolving the CTRNN for this application was not very difficult.

B. SRNN Results

The fitness curve of the average individual for each of the three objectives is shown in Figure 9. Each of the values exhibits a similar convergence curve, indicating that each objective is complementary. That is, they seem to be very dependant on each other to achieve a high fitness.

Figure 10 shows the average gene significance results over the evolutionary runs. The curves are nearly identical, which

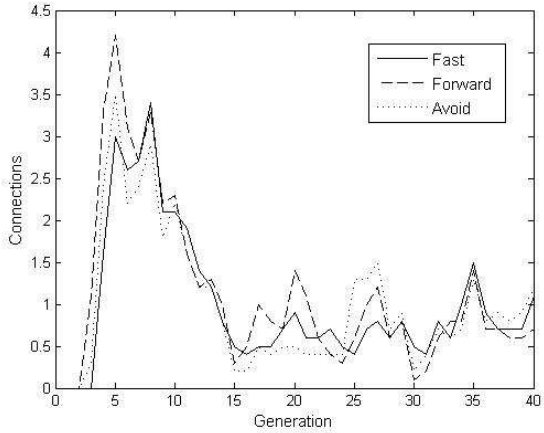

Fig. 6: The average amount of constructive gene pairs over the entire population.

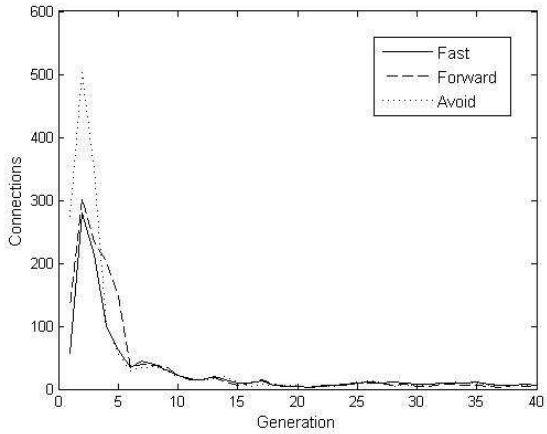

Fig. 7: The average amount of destructive gene pairs over the population.

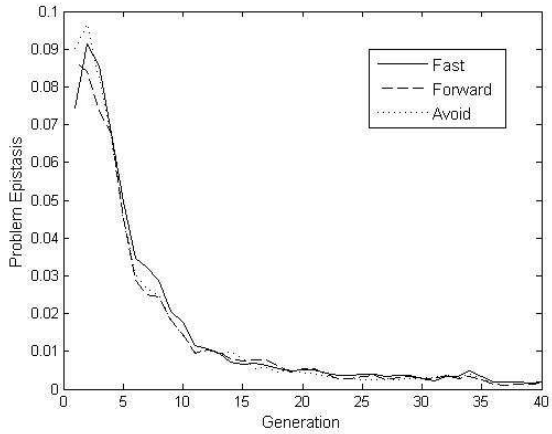

Fig. 8: The amount of problem epistasis.

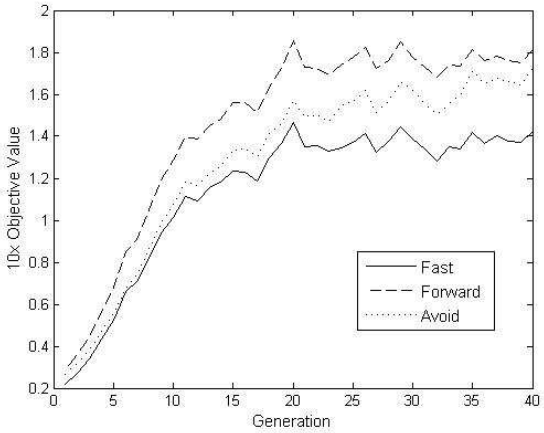

Fig. 9: The average of each objective value per generation for the entire population.

indicates that the contribution of each gene to each of the objective values is nearly the same. Additionally, while the average gene significance value has converged by generation 15, the objective values do not converge until approximately the 20^{th} generation which shows that fitness can be improved without changing the amount of epistatic interaction.

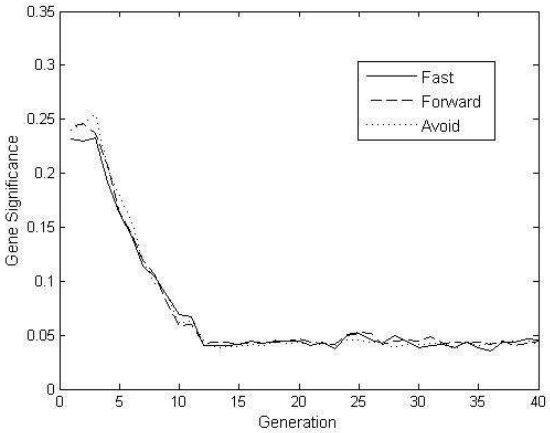

Fig. 10: Average gene significance per generation for the population.

The results for the gene epistasis measure are presented in Figure 11. During the initial 6 generations the amount of gene epistasis is negative and relatively small. This indicates that the weights of the network are working against each other, and therefore having a detrimental effect on the neuro-controller's fitness. The amount of gene epistasis increases until approximately generation 11, which is the same generation the gene significance values converged. At this point the epistasis decreases for two generations then continues to slowly increase as the robot learns to turn right.

Similar convergence behavior can be seen between Figures 11 and 12. The average number of constructive epistatic effects

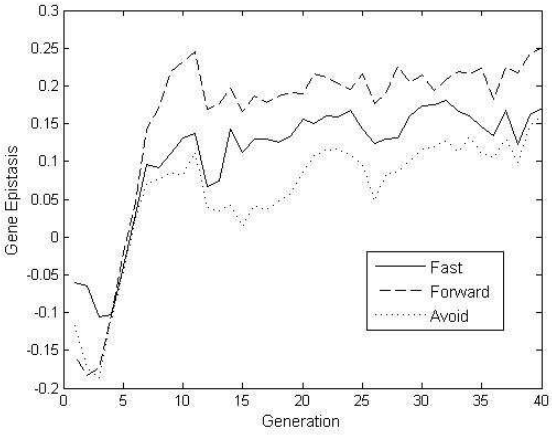

Fig. 11: Average gene epistasis per generation for the population.

(values > 0.5) is plotted for each generation. Additionally, the number of destructive effects (values < -0.5) and the amount of problem epistasis as shown in Figures 13 and 14, respectively, exhibits similar behavior.

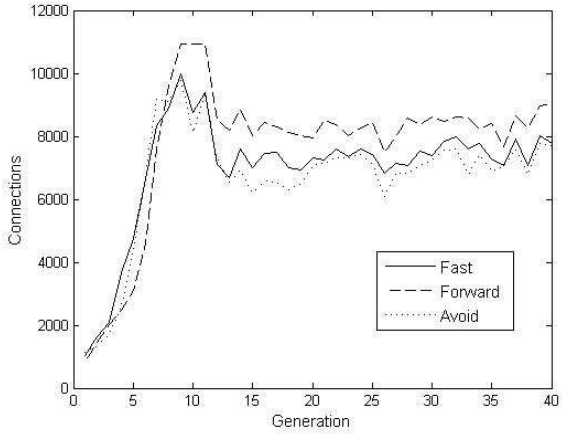

Fig. 12: Average number of constructive epistatic effects of the population.

When comparing the above results to the behavior of the simulated Khepera robot we can see the environmental influence on epistasis. The initial stages correspond to the robot's ability to learn how to turn left in order to avoid a wall. This takes about 6 or 7 generations to evolve. The respective values increase since up to this point unfit "random" solutions permeate through the population. The following five generations represent the time it takes to learn the ability to avoid a wall by turning right. Together, these two tasks make up the major contributing factor to the fitness of the robot (since they allow for a longer life by not colliding with a wall).

The remainder of the evolution concentrates on fine-tuning the robot's behavior, and convergence to that set of behaviors.

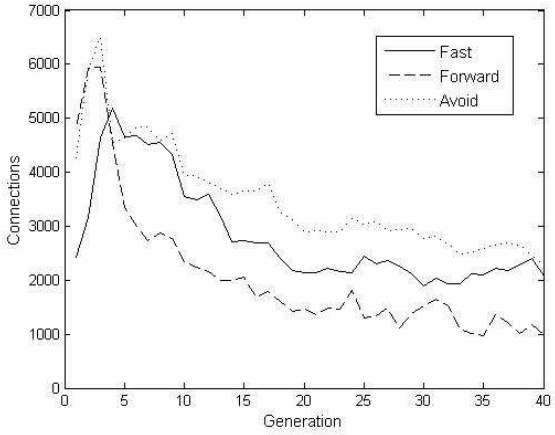

Fig. 13: Average number of destructive epistatic effects of the population.

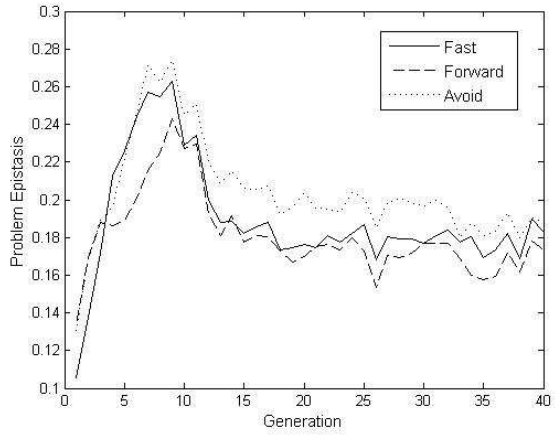

Fig. 14: Average amount of problem epistasis of the population per generation.

As a result, the amount of epistasis decreases. This behavior is summarized in Figure 15.

C. Summary

We examined two neural models for the path-finding robot application. The continuous-time models exhibited lower problem epistasis than the spiking counterpart. These results were expected since the SRNN took more generations to converge to a successful controller. However, the CTRNN controller contained a large amount of destructive interactions which is a result of its architecture. For example, a negative output neuron potential still causes the wheels to rotate.

The SRNN differed from the CTRNN in that it's neurons were much more dependant on each other, as is evident from the large number of constructive and destructive weight pairs. Additionally, the gene epistasis was also much higher. The reasoning behind this lies in the method in which SRNN operate. That is, the wheels of Khepera will turn only in the

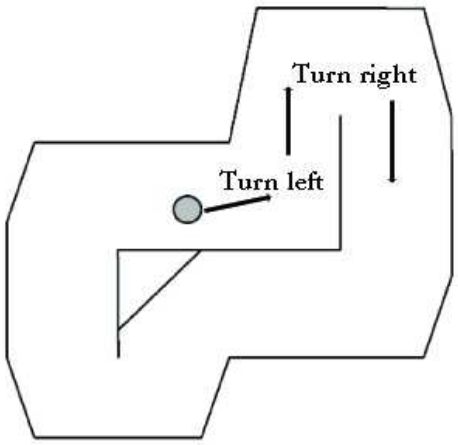

Fig. 15: Behavior of Khepera robot as it learns left and right turning behaviors.

presence of spikes. Therefore, the network must coordinate itself such that given some input, the correct number of spikes are outputted to achieve the desired behavior. As described above, the CTRNN model will always have some action potential at the output neurons.

In both cases, we find that as expected the problems are relatively easy to solve as indicated by the relatively low epistatic measures for each objective during the evolutionary process. However, the problem epistasis for CTRNN-based neuro-controllers was lower than for SRNN-based controllers.

VII. Conclusions and Future Work

This paper has utilized information theoretic measures of epistatic interactions to investigate the influence of neuro-controller weights on the problem difficulty. Additionally, we have explored how these values change during the evolutionary process. We have provided an experimental insight to aid in elucidating the theoretical aspects of epistatic interactions in evolutionary neuro-controllers, specifically those utilizing CTRNN or SRNN models. While we cannot make conclusions over all possible applications, we have shown that environmental influences (that may not be observed from fitness plots) can be observed through epistasis. Furthermore, we examined multi-objective fitness evaluation of the controllers and show that it is possible to achieve the desired behavior through Pareto ranking evaluation if robot behavior is decomposed into smaller sub-goals.

Future work is mainly concerned with understanding epistatic interactions and their influence on problem difficulty from a theoretical and practical standpoint. Additionally, the manner in which we utilized multi-objectivity to achieve the path-finding goal can be further developed. An additional direction for future research may involve the transition from simulated to real environments since results obtained from simulators can be somewhat questionable [**?**].

Appendix

A. Information-Theory Computations

Here we outline the calculations required to perform the epistasis metrics described in Section III. According to [19], entropy $H(Y)$ measures the amount of uncertainty about variable Y and is calculated by

$$H(Y) = -\sum_{y \in Y} p(y) log[p(y)].$$

The average amount of uncertainty of X reduced by knowing the value of Y is known as the mutual information between X and Y.

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \left[\frac{p(x,y)}{p(x)p(y)} \right]$$

Similarly, the conditional mutual information calculates the average amount of uncertainty of X that is reduced by knowing the value of Y. However, in this situation we take into account that the value of variable Z is given.

$$I(X;Y|Z) = \sum_{x \in X} \sum_{y \in Y} \sum_{z \in Z} p(x,y,z) log \left[\frac{p(x,y,z)p(z)}{p(x,z)p(y,z)} \right]$$

References

[1] G. Rawlins, *Foundations of Genetic Algorithms*. Morgan Kaufmann, 1991.

[2] R. Beer and J. C. Gallagher, "Evolving Dynamic Neural Networks for Adaptive Behavior," *Adaptive Behavior*, vol. 1, no. 1, pp. 91–122, 1992.

[3] W. Maass, "Computing with Spiking Neurons," in *The Handbook of Brain Theory and Neural Networks, 2nd Edition*, pp. 1080–1083, MIT Press, Cambridge, Mass., 2001.

[4] D. Goldberg, *Genetic Algorithms in Search, Optimiztion and Machine Learning*. Addison-Wesley, 1989.

[5] J. Teo and H. Abbass, "Multi-Objectivity and Complexity in Embodied Cognition," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 4, pp. 337–360, 2005.

[6] J. Teo and H. Abbass, "Embodied Legged Organisms: A Pareto Evolutionary Multi-Objective Approach," *Evolutionary Computation*, vol. 12, no. 3, pp. 355–394, 2004.

[7] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[8] D. Seo, Y. Kim, and B. Moon, "New Entropy-Based Measures of Gene Significance and Epistasis," in *Genetic and Evolutionary Computation Conference GECCO*, pp. 1345–1356, 2003.

[9] R. Beer, "On the Dynamics of Small Continuous-time Recurrent Neural Networks," *Adaptive Behavior*, vol. 3, no. 4, pp. 469–509, 1995.

[10] D. Floreano, Y. Epars, J. Zufferey, and C. Mattiussi, "Evolution of Spiking Neural Circuits in Autonomous Mobile Robots," *International Journal of Intelligent Systems*, vol. 21, no. 9, pp. 1005–1024, 2006.

[11] K. Funahashi and Y. Nakamura, "Evolution of Spiking Neural Circuits in Autonomous Mobile Robots," *Neural Networks*, vol. 6, pp. 801–806, 1993.

[12] W. Maass and C. Bishop, *Pulsed Neural Networks*. MIT Press, 1999.

[13] W. Maass, "Lower Bounds for the Computational Power of Networks of Spiking Neurons," *Neural Computation*, vol. 8, no. 1, pp. 1–40, 1996.

[14] W. Gerstner and W. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[15] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology, Intelligence and Technology of Self-Organizing Machines*. MIT Press, 2000.

[16] S. Perretta and G. J., "A General Purpose Java Mobile Robot Simulator for Artificial Intelligence Research and Education," in *Proceedings of the Thirteenth Midwest Artificial Intelligence and Cognitive Science Conference*, 2002.

[17] M. Mitchell, *An Introduction to Genetic Algorithms*. MIT Press, 1998.

[18] P. J. B. Hancock, "Genetic Algorithms and Permutation Problems: a Comparison of Recombination Operators for Neural Net Structure Specification,"

[19] A. Coolean, R. Kuhn, and P. Sollich, *Theory of Neural Information Processing Systems*. Oxford University Press, 2005.