Toward Evolving Robust, Deliberate Motion Planning With HyperNEAT

Ben Jolley and Alastair Channon
School of Computing and Mathematics
Keele University
ST5 5BG, UK
Email:{b.p.jolley, a.d.channon}@keele.ac.uk

***Abstract*—Previous works have used a novel hybrid network architecture to create deliberative behaviours to solve increasingly challenging tasks in two-dimensional and three-dimensional artificial worlds. At the foundation of each is a static hand-designed neural network for robust and deliberative motion planning. This paper presents results from replacing the hand-designed motion-planning subnetwork with HyperNEAT. Simulations are run on the original two-dimensional world with, and without, relative position inputs and a multi-evaluation fitness function, thus assessing the relative performance of each strategy. The focus of this work is on solutions adaptable to general environments; following evolution, each strategy's performance is evaluated on 10,000 world configurations. The results demonstrate that although HyperNEAT was not able to achieve as robust results as a hand-design approach, the best strategy was comparable, with just a 3-4% drop in performance. Relative position inputs and the multi-evaluation fitness function were both significant in achieving superior general performance, compared to those simulations without.**

1. Introduction

One goal in Artificial Life (ALife) is to construct systems that exhibit the behavioural characteristics of natural living systems. So, ALife models take abstractions from nature by focusing on key features in order to produce feasible simulation models in silco. To achieve sophisticated behaviours, some approaches have constrained evolution's search space by hand-designing aspects of their models. These approaches can provide state-of-the-art results, but require highly specialised models that demand significant domain expertise to design. For example, research into the neuroevolution of bipedal gaits has used custom network topologies [1, 2, 3], parameter optimization techniques [4] and pre-evolved bootstrapping [5, 6] to constrain the search space. These constraints make evolution feasible, but they also limit its creativity and its ability to generate novel behaviours. Achieving more sophisticated behaviours requires the removal of these constraints.

The success of Deep Learning in recent years, such as setting world record results in supervised learning tasks [7], demonstrates that advanced behaviours are obtainable without careful engineering and considerable domain expertise. Certain neuroevolutionary methods parallel this approach. Topology and Weight Evolving Artificial Neural Networks (TWEANNs) allow only the network inputs and outputs to be defined while the network topology constructs itself. NeuroEvolution of Augmenting Topologies (NEAT) [8] is the most prominent of these approaches, due to small search spaces being optimised before adding links and nodes for increasingly complex behaviour. NEAT has been extended to incorporate indirect encoding techniques, which provides scalability to more sophisticated tasks. Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) [9], Evolvable-substrate HyperNEAT (ES-HyperNEAT) [10] and HybrID [11] all utilise NEAT in their evolutionary process, and have been used for numerous tasks such as evolving Quadruped Gaits[12], general game players [13] and autonomous robot cars [14].

At the core of previous validated ALife tasks a hand-designed subnetwork, known as the Shunting Model, has been used to produce robust and deliberate motion planning in dynamic environments. The Shunting Model has been combined with additional networks to produce high-level reactive and deliberative behaviours; which have been used in 2D and 3D simulations [15, 16, 17, 18] as well as robotics [19]. The Shunting Model's behaviour is unaffected by evolution and remains consistent. A general approach attempting to replicate comparable results will theoretically, as it has not yet been attempted, face a greater challenge.

It is the aim of this work to replace the Shunting Model in an experiment that utilises it, and assess HyperNEAT's ability to produce deliberate and robust motion planning. However, a successful implementation is difficult to quantify. The transition from a static network, with perfect, predictable performance, to an evolvable general model widens the scope outside of pure performance benchmarks; each can be statistically compared, but the two have different potentials. Evolution provides freedom to explore and generate novel behaviours, thus allowing the possibility to adapt to future unforeseen changes to the task. Comparatively, a static network will require manual intervention to change its function. Therefore, successful implementations will take into consideration the performance on the current task relative to future potential.

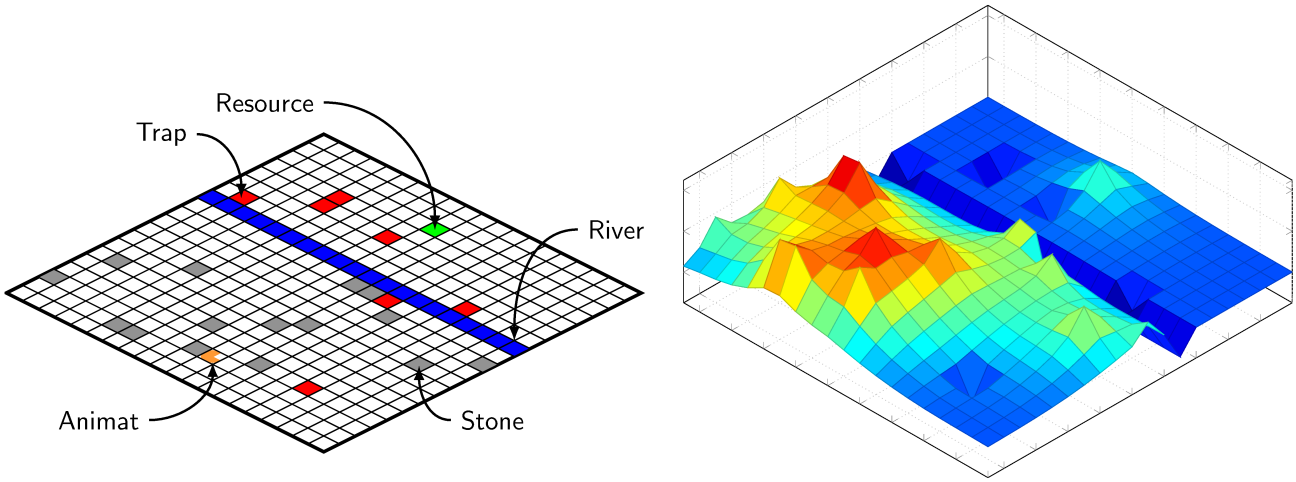


Figure 1. An RC World environment (left) with the corresponding activity landscape (right) produced via the Shunting Model.

2. Background

2.1. River Crossing Task

The River Crossing (RC) Task was devised in Robinson et al. [15] to demonstrate high-level deliberative and reactive behaviours produced by a hybrid neural architecture. The architecture consists of a *Shunting Model* (SM) and *Decision Network* (DN). The animats' goal is to locate the resource on each RC world within 100 time-steps, while avoiding harmful objects. Once the resource is obtained animats proceed to the next RC World. Each world increases in complexity via an expanding river obstruction between animat and resource. Animats must learn to build bridges to cross the river.

2.1.1. RC World. RC worlds are constructed in a 20x20 bounded grid in which each cell can contain zero or one of each of four object types: stone, trap, water and resource; a cell containing none of these is deemed to contain grass. Traversing over a trap or water kills an animat; stones can be picked up and put down. Complexity is enforced by water placed across the world, creating a river obstacle. When placed on water, stones can create a bridge. Fitness is an integer from 0 to 4, determined by the number of RC worlds in which the animat reaches the resource. Animats are evaluated first of a world with river width 0 (no river), then 1, 2 and 3, stopping at first failure.

2.1.2. The Decision Network. The DN is a standard feed-forward neural network that dictates how desirable or undesirable an object type is, and whether an animat should pick up or put down a stone. Topology consists of six inputs, four hidden nodes and five outputs. There are five binary inputs representing the presence or absence of each object type (including one for grass) and a sixth representing if the animat is carrying a stone. There are four outputs corresponding to the desirability of each object type and a fifth used to determine whether the animat should pick up or put down a stone. Output neurons use a hyperbolic tangent activation function and then the *iota values* below, within and above the range [-0.3, 0.3] are converted to -1, 0 and 1 respectively. These iota values indicate the saliency of the attributes in the environment.

2.1.3. The Shunting Model. The SM is a topographically ordered neural network that produces a short trajectory between two positions in a dynamic environment without a learning process. First used in Meng and Yang [19], the SM was applied to real-time robotics to solve maze-type problems by mapping the physical environment to positional neurons. Activity from desirable states propagates through the network to create an activity landscape. Peaks form at objectives and valleys at states to avoid. In the RC task the state attributes are provided via the DN and diffused via the following equation:

$$x_i^{new} = \min\left(\frac{1}{8}\sum_{j\in N_i}[x_j]^+ + I_i,\ \max_i\right) \qquad (1)$$

where x_i^{new} is the activation of neuron i; I_i is the external input determined by the attributes present in cell i; N_i is the receptive field of i; $\max_i$ is the maximum iota value (15). Equation 1 is iterated fifty times to allow activity states to propagate and stabilize across the 20x20 array of SM neurons, as shown in figure 1. Previous RC implementations of the SM used the *Additive Model*, but in extended work the implementation has changed for simplicity and clarity while maintaining the same behaviour [18].

The SM works well in combination with relatively small worlds by representing a practical overview of one's surroundings. However, this SM implementation is impractical with larger worlds, since it provides animats information outside of realistic visual range via an omnipresent view of

their environment. A limited visual radius has been used to address this problem in Borg and Channon [16].

2.2. NeuroEvolution of Augmenting Topologies

NeuroEvolution of Augmenting Topologies (NEAT) was introduced by Stanley and Miikkulainen [8] as a novel neuroevolutionary method that concurrently evolves the weights and topology of an artificial neural network. NEATs evolutionary process utilises an initial population of small and simple networks which complexify over generations, thus leading to increasingly sophisticated behaviour. The three main components of this process are: *1) Crossover, 2) Speciation* and 3) *Complexification*; each made practical via historical markings which incorporates an incremental innovation number to each gene. The genome representation is a linear order of link and node mutations. Topologies are compared by lining up each gene and comparing innovation numbers and content, thus avoiding expensive analysis. Genes either contain the same links (matching) or do not (disjoints and excess); these factors can be used to define species. Speciation operates dynamically via a similarity threshold. Speciation prevents the global population from converging on local optimums by individuals only competing in their niches. Therefore, structural innovations are protected and allowed time for optimisation via mutation and crossover. Crossover between structurally similar individuals, in theory, avoids the Competing Conventions problem. NEAT has proven itself in various task domains such as competitive robot control [20], a crash warning system [21] and Go [8]. Extensions include FT-NEAT, rtNEAT and CPPN-NEAT, the latter being fundamental to HyperNEAT.

2.3. Hypercube-based NEAT (HyperNEAT)

HyperNEAT is the indirect encoding extension of NEAT [9] that utilises a *Compositional Pattern Producing Network* (CPPN) to encode the weights of a representation network, known as the substrate. CPPNs were devised as an abstraction of natural development encoding to generate complex, regular and geometric patterns [22]. CPPNs have been shown to produce patterns with structural components similar to those observed in nature, including repetition, repetition with variation, symmetry and imperfect symmetry. Regularities are tied to specific activation functions, such as sine for repetition and Gaussian for symmetry. Varying activations in combination have the potential to produce complex patterns. Visual affirmation of CPPNs ability to produce natural biological patterns is seen on the on-line service Picbreeder [22]; users collaborate to construct images through interactive evolution. Deep neural networks and other human users are able to recognize and classify the produced images into sensible categories [23]. CPPNs' structural similarity to ANNs allows evolution to use NEATs established benefits, with the addition of an evolvable activation function within each node. Weights for each link are provided by a CPPN using the positional coordinates of each links starting and ending node. Geometric properties of a task domain can therefore be exploited, provided that an appropriate substrate is used.

3. Experiment Setup

The following sections will address various design considerations with the RC task and HyperNEAT.

3.1. RC Task

The DN will be pre-evolved to achieve the highest fitness in the conventional RC task, for this work aims to replicate the SM. The SMs full capabilities would include preventing negative values from propagating in the activity landscape. This addition would require contrasting activation functions in the substrate, such as Rectified Linear Unit (ReLU), or an architectural change, such as the addition of a hidden layer or separate inputs for negative and positive values. HyperNEATs ability to achieve the primal navigation functionality of the SM is not yet known, so this feature will be reserved for future work. Therefore, undesirable objects will not be used for inputs, but instead deducted directly from the activity landscape so they will continue to be avoided. Inputs will be provided by the pre-evolved DN. Outputs will map directly to the activity landscape. Animats will follow the highest activation neighbour, as in the conventional RC task and other HyperNEAT implementations [13].

An additional fitness function will be simulated due to the change in task domain. The original fitness function assesses how successful an animat was on a selection of four RC worlds and does not translate to the animats general success. An SM replacement must be able to achieve equivalent results in all RC world combinations. For more granular feedback, the new fitness function uses the mean from 10 RC task runs. Animats will have to consistently build bridges in different environments to survive. This modified fitness function can be viewed as a larger environment with each RC task being a limited section of that environment.

3.2. HyperNEAT

The substrate design is important to the performance of HyperNEAT, as established in previous work; a general substrate design can prove successful in a variety of different task domains [24]. The *sandwich substrate* is one highlighted in existing literature; it has been used successfully in locomotion with no obvious geometric relationship between sensor positions and substrate inputs [24, 25]. To avoid the bias of expert domain knowledge, this work uses the sandwich substrate with 20x20 input and output layers. Additionally, links with weight values less than 0.2 or greater than -0.2 were discarded in previous implementations. This restriction is removed in this work, thus allowing greater nuances in the encoding pattern.

Two CPPN input variations will be simulated. Standard CPPN inputs for the majority of HyperNEAT experiments include $x1$, $x2$, $y1$, $y2$ and a bias. Delta inputs ($x1$ -

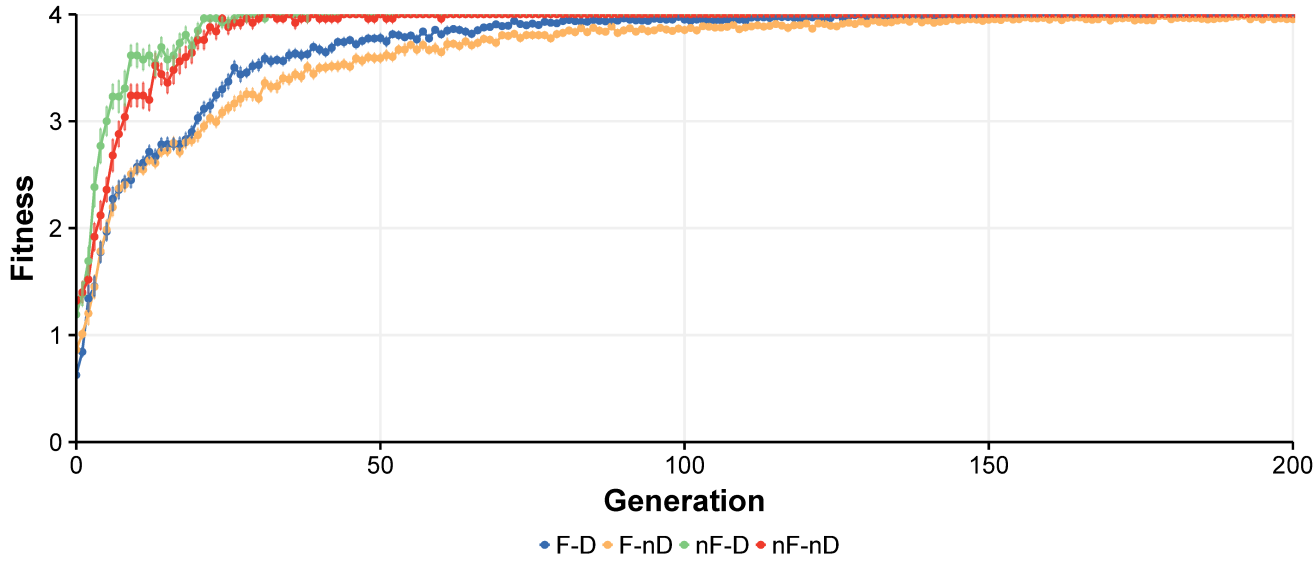


Figure 2. Mean fitness of the fittest individuals from twenty-five runs of the RC Task. Error bars represent standard deviations. (F-D = Fitness Modification with Delta inputs. F-nD = Fitness Modification without Delta inputs. nF-D = No Fitness Modification with Delta inputs. nF-nD = No Fitness Modification without Delta inputs.)

$x2$) ($y1$ - $y2$) are included in certain experiments allowing patterns to emerge from relative distances. Delta may aid in the SMs reliance on relative patterns, as activity values degrade more the further away from a desirable object they are. A comparison between delta and non-delta will reveal whether distance information yields benefits in this task, or if absolute inputs can achieve the same quality results, as seen in other work [26].

HyperNEAT parameters are vital for tuning [27]; however, the parameters within the original package (Checkers, Go, Robot Arm) remain broadly similar with only the topological mutation rate and population size deviating for each experiment. A lower than average topological mutation rate (0.05%) and a large population size (1000) will be used, as appears to be the trend with complex tasks in the original package.

Finally, HyperNEAT is used in this work over the more advanced HybrID and ES-HyperNEAT, due to this task's requirements. Until recent extensions to HybrID, which uses both direct and indirect encoding, a pre-known evolutionary generation was required to switch encoding strategy once performance had plateaued [28]; domain specific knowledge, such as this, will be avoided in this work. ES-HyperNEAT is only applicable for networks with hidden layers, which this task does not require, due to the theoretical simplicity of a successful encoding pattern.

3.3. Results

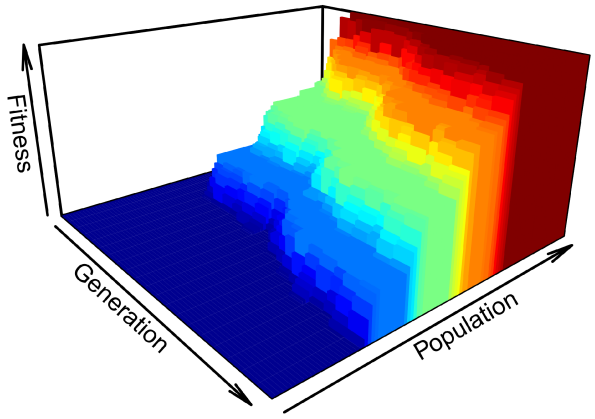
Twenty-five runs for 200 generations were carried out, for each HyperNEAT approach, on the RC task. Each approach is separated with the use of a fitness modifier (Section 3.1) and delta inputs (Section 3.2). Each method is named as follows:

- F-D: Fitness Modification with Delta inputs
- F-nD: Fitness Modification without Delta inputs
- nF-D: No Fitness Modification with Delta inputs
- nF-nD: No Fitness Modification without Delta inputs

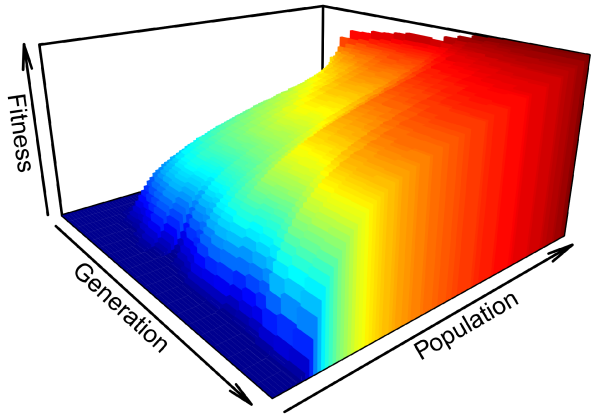
The goal is to assess performance of each approach comparatively and against the SM, with a focus on general performance. Performance is represented as the fittest individual, due to NEAT's use of speciation, which purposely produces suboptimal individuals to prevent the population from converging on local optimums, as discussed in section 2.2. Figure 3 demonstrates this with an overview of a population's performance over generations using the original RC fitness function and the fitness modification.

Scores for the fittest individuals from all the runs were collected and aggregated, as shown in figure 2. From this graph every run was able to solve the highest level of difficulty the RC task requires, and maintains it. Each delta counterpart saw some statistical advantage in early generations over the non-delta inputs ($p < 0.05$), but this fluctuated each generation and by the 123rd generation there was no statistically significant difference between strategies.

The fittest animats from each run were evaluated in the robustness test for each river width, which simulated the animats through 10^4 static RC world configurations. Animat's performance on this test represents their ability to adapt to general environments. Comparing figure 2 and 4 it can be seen that performance on the fitness functions did



(a) Orignal RC Fitness Function



(b) Fitness Modification

Figure 3. Representation of HyperNEAT's fitness score in a population without (a) and with (b) the fitness modification. Simulated over 200 generations, a maximum fitness of 4 and population size of 1000.

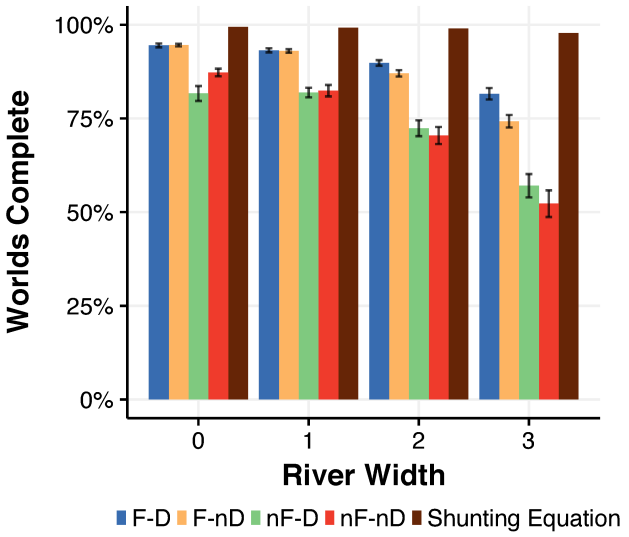


Figure 4. Completion rates of Robustness Test, using each HyperNEAT approach. Individuals simulated are the fittest at generation 200 from all runs, Twenty-five runs, per approach, per river width. Error bars represent standard deviations. (F-D = Fitness Modification with Delta inputs. F-nD = Fitness Modification without Delta inputs. nF-D = No Fitness Modification with Delta inputs. nF-nD = No Fitness Modification without Delta inputs.)

TABLE 1. p VALUES FROM TWO SAMPLE T-TESTS BETWEEN DELTA AND NON-DELTA APPROACHES, USING THE ORIGINAL FITNESS FUNCTION AND FITNESS MODIFICATION, IN THE ROBUSTNESS TEST. INDIVIDUALS SIMULATED ARE THE FITTEST AT GENERATIONS 200.

Two Sample T-Test (p values)				
	River 0	River 1	River 2	River 3
F-D				
F-nD	0.8902	0.809	0.01796	0.002184
nF-D				
nF-nD	0.01862	0.806	0.5363	0.3156

not translate to general performance. RC worlds with river width 0 and 1 achieved a consistent 93-94% completion with the fitness modification, with and without delta. The original fitness function's performance saw a larger deviation and spread across a lower completion range, 80-87%. RC worlds with river width 2 see a drop in performance on all approaches, and a subsequent drop at river width 3. Table 1 displays the p values from a two sample t-test between delta and non delta approaches. When using the fitness modification, delta inputs provide a significantly higher completion percentage during the more complicated RC worlds with river width 2 and 3. All approaches with the fitness modification saw a statistically significant advantage, compared to their counterparts without.

Runs were extended to higher generations to assess if fitness had plateaued in general performance, due to the disparity between the fitness functions and the general performance. Figure 5 displays the mean completion in the robustness test at river width 3 with an extension to 500 generations. Performance for each fitness function appears to stay relatively consistent with results achieved at the original generation 200. However, the fittest individual achieved its highest performing completion at generation 356 and after 300+ generations the performance appears less volatile than in the preceding generations. The original fitness function appears noisy in general performance. In contrast, the fitness modification can produce greater consistency at a higher completion rate and in turn produced the fittest individual.

Individuals with high completion percentages in the robustness test produced similar functioning activity landscapes. However, none resembled the SM. Figure 6 shows an example of activity landscapes produced by an individual with a 95.9% completion on the robustness test at river width 3. Animats which require stones are directed south of the river by the activity landscape. Once a stone is carried, animats are directed to the furthest north position with a peak at the resources location. This forces the animat north

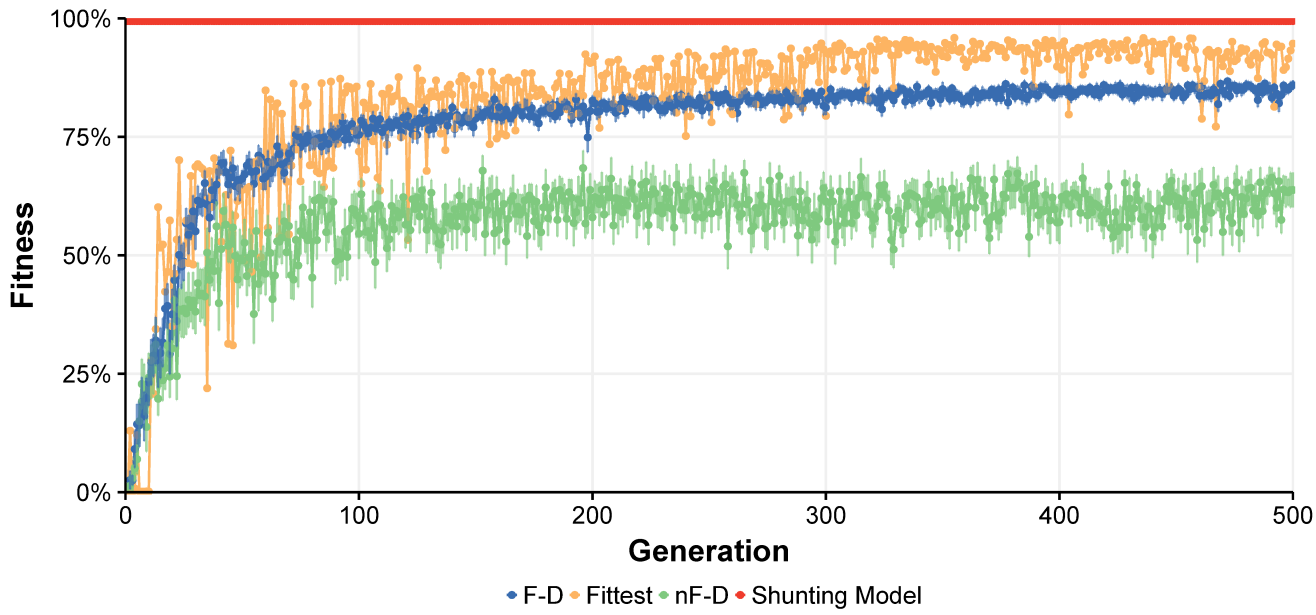


Figure 5. Completion rates of Robustness Test at River Width 3 using HyperNEAT with and without the fitness modification. Twenty-five runs, per approach, for 500 generations. Error bars represent standard deviations. (F-D = Fitness Modification with Delta inputs. nF-D = No Fitness Modification with Delta inputs. Fittest = The individual with the the highest completion percentage from all runs from any approach.)

until the river is interacted with, at which point the stone is placed upon the river. Animats will then again return south to acquire stones, and continue until a bridge is complete.

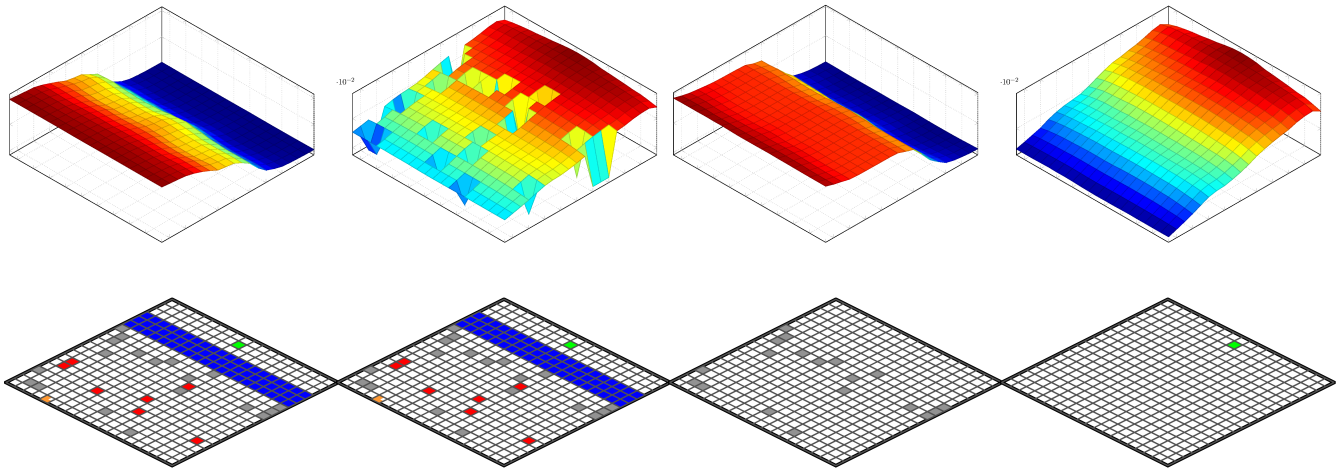
3.4. Discussion

No HyperNEAT implementation could produce the same quality of deliberate, robust motion planning when compared to the SM at any river size. Solutions at river width 0 and 1 produce relatively consistent results, but a performance degradation occurs at river width 2 and there is a further drop at river width 3. Beyond river width 1 the task requires new behaviours: such as the deliberate movements from river to stone after placing a stone on the river and the construction of a connecting bridge. Each increment in river width correlates to the need for greater sophistication in both these behaviours. Therefore, the drop in performance suggests HyperNEAT solutions lack these deliberate planning skills. However, practical HyperNEAT implementations are possible. Individuals with the fitness modification and delta inputs were found achieving 90%+ RC world completions on the most difficult tasks.

Figures 4 and 5 shows that the fitness modification provides superior general performance. The modification allows the obvious benefit of multi-performance feedback. Figure 3 also shows the benefit during evolution. In comparison, the original RC fitness function only provides five fitness states for diversity to occupy. A relatively large amount of the population then remains at fitness 0 throughout. The fitness modification was used to simulate a larger environment for greater avenues of learning feedback; however, there are issues with the current representation. Currently animats can fail a RC world, such as drowning, and continue to attempt the task again. This would not be possible in a larger environment, as animats may still fail at these early stages leading to poor un-robust evolved behaviours as seen in the original fitness function. The fitness modifier provides a larger impact overall when compared to the delta counterpart. However, used in combination there is a statistical advantage in the more difficult tasks.

Delta inputs and the fitness modification show an advantage at river width 2 and 3, as seen in table 1. At these river widths, animats require greater deliberative behaviour. The benefits of deltas may be related to creating relational patterns around activated inputs, therefore providing contextual awareness of nearby objects such as stones. Absolute inputs could theoretically also achieve these patterns, albeit at greater difficulty and evolutionary cost. However, just how deltas provide an advantage is not clear from observing successful activity landscapes, as solutions appear to rely on linear patterns.

Successful HyperNEAT solutions exhibit a funnelling type behaviour that was proven reliable for general performance on the robustness test, as shown in figure 6. At the simplest interpretation, an animat will move north or south depending on whether a stone is being held. Figure 6 (*a* and *c*) shows that when stones are desirable evolution has discovered being south of the river is beneficial for locating



(a) RC World *(b) Holding Stone* *(c) Stones Only* *(d) Resource only*

Figure 6. RC worlds (below) with their corresponding activity landscapes (above). Each activity landscape is produced by the fittest individual in the robustness test.

stones. The highest activation points are those furthest south. Once a stone is placed on the river and another has to be collected, the further south an animat moves the greater its likelihood of interacting with another stone. Figures 6 (*b* and *d*) show how an animat will traverse to the resource once a stone is obtained. Despite positive general results on river widths 0 and 1, there is a problem in practice, as animats' movement lack the compelling behaviours of believable, deliberate motion "planning required for robust performance on wider rivers.

Animats take complex, longer paths to stones in the RC world while ignoring those in their immediate neighbourhood. This can lower animats' success due to their 100 steps limit, but more importantly it appears to show a lack of intelligent or deliberate decision-making. This is most common after the completion of a bridge, since animats will proceed back to a stone before obtaining the resource. In comparison, the SM produces a local activity field around the resource that allows animats to proceed toward it once the bridge is complete. Animats from the RC 3D simulation in Stanton and Channon are able to exhibit life-like responses, due to their reaction to the activity space [18], which some observers have described as resembling surprise, confusion and even happiness. HyperNEAT's fittest model may lose these subtle behaviours if it were to replace the SM in the RC 3D system. Although, a case can be made that if animats are completing RC worlds within a reasonable time frame, which all completions are required to do, their behaviours are deliberate but inefficient.

3.5. Conclusion and Future Work

HyperNEAT has demonstrated the capability to produce feasible deliberate and robust motion planning, but in this task not to the quality of a pre-designed solution. These results were achieved using a general HyperNEAT configuration with no problem-specific aspects of the network design. The performance maintained a great level of consistency (90%+) in locating the resource and simple bridge-building with a multi-evaluation fitness function, show in in figure 4. Mean performance drops to a respectable level (80%) at the most difficult deliberate task, but individuals can still be found 3-4% below the performance of the SM, as shown in figure 5. Additional evaluations may further improve performance and should be investigated. This work also demonstrates the importance of relative distance information in producing greater general solutions at more difficult deliberate tasks.

During simulation, there are inefficient choices in animats' motion planning decisions. However, their actions are still considered deliberate due to the completion of difficult tasks within a time constraint. Simulating a larger environment with a multi-evaluation fitness function may not be applicable in practice. Future iterations of this model could introduce fatal events, to prevent animats that fail at a task from receiving a fitness reward.

Finally, further work can now be taken to incorporate the other network functions. Such as the feature of avoiding undesirable inputs and incorporating the decision network's ability of defining objects desirability. Removal of these constraints would better allow the possibility to adapt to future tasks

References

[1] L. Wiklendt, S. K. Chalup, and M. M. Seron, "Simulated 3D biped walking with an evolution-strategy tuned spiking neural network," *Neural Network World*, vol. 19, no. 2, p. 235, 2009.

[2] E. D. Vaughan, "The evolution of an omni-directional bipedal robot," Ph.D. dissertation, Sussex, 2007.

[3] K. Wolff, J. Pettersson, A. Heralic, and M. Wahde, "Structural evolution of central pattern generators for bipedal walking in 3d simulation," in *Systems, Man and Cybernetics, 2006. SMC'06. IEEE International Conference on*, vol. 1. IEEE, 2006, pp. 227–234.

[4] M. Van de Panne and A. Lamouret, "Guided optimization for balanced locomotion," in *Computer animation and simulation*, vol. 95. Springer, 1995, pp. 165–177.

[5] D. Hein, M. Hild, and R. Berger, "Evolution of biped walking using neural oscillators and physical simulation," *RoboCup 2007: Robot Soccer World Cup XI*, pp. 433–440, 2008.

[6] J. H. Solomon, M. A. Locascio, and M. J. Hartmann, "Linear reactive control for efficient 2d and 3d bipedal walking over rough terrain," *Adaptive Behavior*, vol. 21, no. 1, pp. 29–46, 2013.

[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[8] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[9] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.

[10] S. Risi and K. O. Stanley, "An enhanced hypercube-based encoding for evolving the placement, density, and connectivity of neurons," *Artificial life*, vol. 18, no. 4, pp. 331–363, 2012.

[11] J. Clune, K. O. Stanley, R. T. Pennock, and C. Ofria, "On the performance of indirect encoding across the continuum of regularity," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 3, pp. 346–367, 2011.

[12] J. Clune, B. E. Beckmann, C. Ofria, and R. T. Pennock, "Evolving coordinated quadruped gaits with the HyperNEAT generative encoding," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 2764–2771.

[13] M. Hausknecht, P. Khandelwal, R. Miikkulainen, and P. Stone, "HyperNEAT-GGP: A HyperNEAT-based atari general game player," in *Proceedings of the 14th annual conference on Genetic and evolutionary computation*. ACM, 2012, pp. 217–224.

[14] J. Drchal, J. Koutník, and M. Snorek, "HyperNEAT controlled robots learn how to drive on roads in simulated environment," in *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE, 2009, pp. 1087–1092.

[15] E. Robinson, T. Ellis, and A. Channon, "Neuroevolution of agents capable of reactive and deliberative behaviours in novel and dynamic environments," in *European Conference on Artificial Life*, 2007, pp. 345–354.

[16] J. M. Borg and A. Channon, "Evolutionary adaptation to social information use without learning," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2017, pp. 837–852.

[17] B. P. Jolley, J. M. Borg, and A. Channon, "Analysis of social learning strategies when discovering and maintaining behaviours inaccessible to incremental genetic evolution." in *SAB*, 2016, pp. 293–304.

[18] A. Stanton and A. Channon, "Incremental neuroevolution of reactive and deliberative 3d agents," in *Advances in Artificial Life, ECAL 2015: Proceedings of the Thirteenth European Conference on the Synthesis and Simulation of Living Systems*. Keele University, 2015, pp. 341–348.

[19] M. Meng and X. Yang, "A neural network approach to real-time trajectory generation [mobile robots]," in *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, vol. 2. IEEE, 1998, pp. 1725–1730.

[20] K. O. Stanley and R. Miikkulainen, "Competitive coevolution through evolutionary complexification," *J. Artif. Intell. Res.(JAIR)*, vol. 21, pp. 63–100, 2004.

[21] K. Stanley, N. Kohl, R. Sherony, and R. Miikkulainen, "Neuroevolution of an automobile crash warning system," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*. ACM, 2005, pp. 1977–1984.

[22] J. Secretan, N. Beato, D. B. D Ambrosio, A. Rodriguez, A. Campbell, and K. O. Stanley, "Picbreeder: evolving pictures collaboratively online," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008, pp. 1759–1768.

[23] A. M. Nguyen, J. Yosinski, and J. Clune, "Innovation engines: Automated creativity and improved stochastic optimization via deep learning," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2015, pp. 959–966.

[24] J. Clune, C. Ofria, and R. T. Pennock, "The sensitivity of HyperNEAT to different geometric representations of a problem," in *the 11th Annual conference*. New York, New York, USA: ACM Press, 2009, pp. 675–682.

[25] S. Lee, J. Yosinski, K. Glette, H. Lipson, and J. Clune, "Evolving gaits for physical robots with the HyperNEAT generative encoding: The benefits of simulation," in *European Conference on the Applications of Evolutionary Computation*. Springer, 2013, pp. 540–549.

[26] J. Gauci and K. Stanley, "Generating large-scale neural networks through discovering geometric regularities," in *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. ACM, 2007, pp. 997–1004.

[27] K. O. Stanley, J. Clune, D. B. DAmbrosio, C. D. Green, J. Lehman, G. Morse, J. K. Pugh, S. Risi, and P. Szerlip, "CPPNs effectively encode fracture: A response to critical factors in the performance of HyperNEAT," *Technical Report CS-TR-13-05*, 2013.

[28] L. Helms and J. Clune, "Improving HybrID: How to best combine indirect and direct encoding in evolutionary algorithms," *PloS one*, vol. 12, no. 3, p. e0174635, 2017.