Artificial Life and Robotics
https://doi.org/10.1007/s10015-025-01019-z

ORIGINAL ARTICLE

Neuroevolution for vision-based seeking behavior in 3D soft voxel robots

Christian Hahm[1,2]

Received: 11 November 2024 / Accepted: 27 February 2025
© The Author(s) 2025

Abstract
This paper details a simple experiment that tests two genetic encodings, NEAT and HyperNEAT, for the evolution of vision-based food-seeking behavior in neural-controlled 3D soft voxel robots. The evolution of food-seeking behavior is a preliminary step towards ecosystems of advanced artificial animals, in which the animals seek both food and mates. Two environments were tested: with and without deadly obstacles. Traditional evolutionary search was used, with an objective-based fitness function. Both NEAT and HyperNEAT encodings were tested for the evolution of robot neural controllers. The results of the experiment showed the NEAT encoding resulted in increasingly effective food-seeking behavior over time, whereas experiments with the HyperNEAT encoding did not achieve the desired behavior. This suggests that NEAT at least is a viable algorithm to evolve neural networks for the task of vision-based object-seeking in complex robots, and warrants further experimentation. On the other hand, HyperNEAT struggled with this task. This could be due to a number of reasons, including a common issue like EA being stuck in local optima, or because the encoding might struggle to evolve and represent irregular structures required for the task.

Keywords Animat · Neural network · Evolution · Voxel · Robot

1 Introduction

The purpose of this work was to test if it is possible to evolve vision-based object-seeking behavior in simulated 3D soft voxel robots, to make them act like artificial animals (animats). Evolving robots with animalistic seeking behavior is the first step towards implanting those robots in an ecosystem where they can autonomously live, learn, and evolve like animals. Seeking behavior enables food-seeking and mate-seeking in animats. Through food-seeking, they autonomously sustain themselves and their energy, rather than assuming infinite resources. Through mate-seeking, they autonomously perform sexually selection, rather than requiring an external mate-selection algorithm. Evolutionary ecosystems of artificial animals, with evolving minds and/or bodies, make for interesting experiments in the field of artificial life and the related field of artificial intelligence. The practice of evolving food-seeking (and mate-seeking) behavior to achieve self-sustaining animats was employed by [25, p 4] in the *PolyWorld* animat ecosystem.

For any such animat evolutionary setup, there is the question of frameworks. What is the framework for the processes of the (1) mind/brain, (2) body, and (3) environment? What about the genetic representations (of brain and body)? A given setup could have them all separated, such that each aspect operates using its own separate algorithm, or (more ideally) some could be combined (e.g., as in evolving both the brain and body using a single genome).

In this work, only the neural network ("brain"), more specifically its evolution for the "seeking" task, was tested. The neural network was evolved to control a soft voxel robot, a robot framework that has been suggested (by [3, 10]) as a next-generation substrate for robotics and morphological evolution. By selecting the newer soft voxel robot framework over the more traditional articulated robot framework, this work may be easier to extend in the future with various naturalistic morphologies and possibly their evolution.

✉ Christian Hahm
christian.hahm@temple.edu

[1] Department of Computer and Information Sciences, Temple University, Philadelphia, USA

[2] CleverLibre, Philadelphia, USA

Published online: 29 March 2025

2 Related works

The field of artificial life intersects with the field of artificial intelligence (AI) in the topic of "artificial animals", aka "animats" as coined by [24]. The idea by [24] is to evolve animats to perform better and better in increasingly difficult environments, which should hypothetically equate to them evolving higher intelligence. Of course, before we can even consider the environment aspect, we must first consider the evolution of the animat. There are two aspects that make an animat: a brain and a body.

Artificial bodies are essentially robots. Compared to neuroevolution, the works for evolving robotic bodies have been less fruitful, especially in terms of evolving useful high complexity. This is due to the difficulty of the problem, which seems to have persisted, in part, because robots have mostly been conceptualized as articulated (i.e., with joints) and rigid (e.g., made of metal). An incredible seminal work by [17] evolved both the brain and bodies of virtual creatures, in a single genome no less, for various locomotion behaviors like galloping, swimming, etc. The bodies were articulated robots. However, since then, the field has not since advanced significantly, due to the inherent difficulty of evolving traditional robots [3, p 1].

To tackle this stagnation in evolutionary robotics, [3, 10] suggested adopting a different kind of robot called "soft voxel robots" (SVRs). Soft voxel robots are a relatively new type of robot developed by [10] composed of soft (and/or hard) regular cubes (aka "voxels"). This framework provides interesting properties not exhibited by traditional robots, such as soft deformation and cellular spatial-discretization. SVRs may be more amenable to evolution than traditional robots, since the components (voxels) of SVRs exist in a discretized cellular space, whereas the components of traditional robots exist in a continuous space. The "soft" framework may also allow evolving robots which appear and move more naturalistically, deforming and bending in various ways, compared to rigid robots.

Artificial brains usually take the form of artificial neural networks. The field of evolving neural networks is called "neuroevolution", and has quite a rich history with various possible methods [20, p 25], though not all are equally effective across tasks. CPPN-NEAT (aka HyperNEAT) is an algorithm sometimes used to evolve aspects of robots, such as their brains (but also their bodies) for locomotion tasks. HyperNEAT has been used to evolve neural controllers for articulated robot locomotion (e.g., [4, 9]), to evolve SVR morphologies for locomotion (e.g., [3, 22]), and to evolve neural controllers for SVR locomotion (e.g., [22]).

Since the invention of SVRs, there have been various works on evolving them in various ways; their controllers/brains, their bodies, and both. This has been experimented in both 3D and 2D environments. The SVRs do not always need neural controllers to locomote, since the bodies can move by themselves via specifically-timed rhythmic contractions. For example, [3, 5] evolved pure morphologies (no controllers) of 3D SVRs for locomotion. Later works, such as [7, 14, 16] evolve controller networks for fixed-morphology 2D SVRs. More recently, co-evolution of both body and controller of 2D and 3D SVRs are being explored (e.g., by [2, 15, 22] and others). [14] enriched the sensory experience of SVRs past mere touch, equipping 2D SVRs with vision (raycast) sensors to aid in locomotion.

Still, in almost all cases, the SVRs have evolved only for the relatively simple task of locomotion. In an effort to take us past simple locomotion, [1] introduced *Evolution Gym* (*EvoGym*), an application containing various environments for evolving 2D SVRs. In *EvoGym*, there were two classes of tasks, various locomotion tasks (walk on various surfaces) and various complex object manipulation tasks (catch, throw, or lift objects). Interestingly, experiments from [1, p 7-8] resulted in data suggesting that CPPN-NEAT struggles to evolve SVR morphologies that can accomplish the more complicated tasks, compared to other genetic encodings. A similar observation was made in my experiments for evolving soft voxel neural controllers using HyperNEAT, as discussed below.

The novel contribution here is to successfully evolve neural controllers in a multi-agent environment for 3D SVRs that result in "food-seeking" behavior. This is a small step towards enabling ecosystems of advanced and intelligent artificial animals, as in the plan proposed in [8]. The idea of simulating artificial animals in an ecosystem is a fundamental topic explored in the field of artificial life for many decades. Some of the notable works related to these overall ideas include the formalization of the idea of animat evolution [24], the historic animat ecosystem simulator *PolyWorld* [25], and the modern animat ecosystem simulator *EcoTwin* [21].

3 Experiment

3.1 Evolutionary setup

An "arena" was constructed to contain the animats and their food (see Fig. 1). The arena is a flat square floor, of size 50×50 units, surrounded on all 4 sides by walls. Robots, food, and obstacles spawned within the bounds of the arena. Whenever an entity was spawned, it was spawned at least 1.5 units away from any other entity. A piece of food took the form of a thin $0.25 \times 1 \times 0.25$ units green rectangular pillar/block, whereas an obstacle took

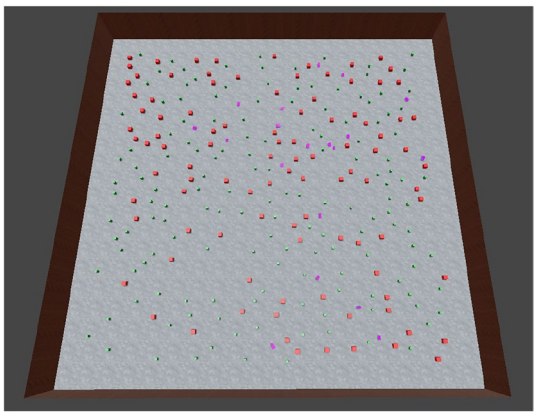


Fig. 1 The animat arena. The elongated purple blocks are the worm robots, the small green blocks are the food, and the larger red blocks are the obstacles (colour figure online)

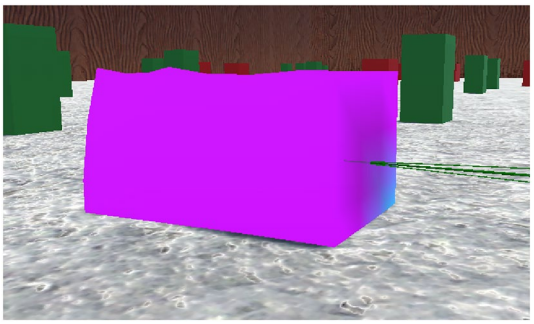


Fig. 2 The $5 \times 3 \times 3$ worm robot. Most of the body was made from purple voxels, which contained touch sensors. The single dark voxel (in the center of the 3×3 square face on the right) contained a vision/raycast sensor; the raycasts are pictured emanating from the voxel. The single blue voxel (bottom right of the face) contained a sine-wave generator (colour figure online)

the form of a thicker $0.5 \times 1 \times 0.5$ units red rectangular pillar/block. If an obstacle/food was consumed, it was moved to a new random position. There were 150 food blocks, and 100 obstacle blocks in the obstacle scenario. There were no collisions enabled between robots, or between robots and food/obstacle blocks, so if 2 robots intersect they simply overlap and move past each other without interacting.

The animats were evolved to improve their food-seeking behavior, by keeping a list of the latest (dead) animats and selecting them from the list for reproduction. The chance for an animat to be selected for reproduction was based on a weighted probability distribution, more specifically using a cumulative distribution function (CDF). The weight/probability used to select an animat was the amount of food it had eaten during its lifetime divided by the total amount of food consumed by all animals in the reproductive pool. In this way, the algorithm gave the highest chance to animats who had eaten the most food, while still giving a (lower) chance to animats who had eaten less food.

This method of propagating the latest dead organisms is an important distinction from a pure elitist approach. In the elitism approach, a set of the all-time highest-achievers is maintained and offspring are generated from it, as was done in *PolyWorld* [25, p 4]. Instead, the approach here continuously progresses through and builds on the immediately preceding generations, even if they are suboptimal compared to the highest-fitness animats of all time (i.e., the "elites"). This is more like a traditional genetic algorithm. One reason for taking this "continuous window" approach over elitism is to prevent the reproductive pool from becoming stagnant with moderate-scoring individuals stuck in local optima.

3.1.1 Robot

The robot was shaped like a rectangular worm of size $5 \times 3 \times 3$ voxels (see Fig. 2). There were 3 possible voxel types, which determined what type of sensor the voxel would contain: a touch sensor, raycast sensor (for vision), and sine-wave generator. All voxels were actuatable.

The program used for simulating soft voxel robots was called *Voxelyze* [10]. Though these robots are made of discrete voxels, they move, rotate, and deform in a continuous space (not a cellular-automata space). The parameters for *Voxelyze* were set according to Table 1.

The parameter values of *Voxelyze* were determined initially using the code of [12], then tweaking them with trial-and-error in an attempt to get a reliable simulation. There are a wide range of values that could be used, the values in Table 1 were the ones used for the data captured in *Section* 3.4. Note that, due to the small lattice dimension, the

Table 1 Voxelyze parameter values

Parameter name	Parameter value
Lattice dimension	0.01 m
Elastic modulus	50×10^6 Pa
Coefficient of thermal expansion (CTE)	$0.01°C^{-1}$
Density	1×10^6 kg/m^3
Poisson's ratio	0.35
Internal damping	1
Global damping	0.01
Collision damping	0.8
Coefficient of static friction	1
Coefficient of kinetic friction	10
Gravity multiplier	1.5

robot was additionally scaled up in size by 10 to fit better in the world.

All voxels in the robot virtually contained 2 sensory neurons and 3 motor neurons.

The activations (in $[-1, 1]$) of the 3 motor neurons determined actuation along the voxel's 3 spatial axes: X, Y, and Z. Thus, a motor neuron for axis X with activation 1 expanded the voxel's X-axis up to its maximum size/temperature (here, at +29°C), whereas activation −1 shrank the voxel down to its minimum size/temperature (here, at −29°C), and activation 0 put the voxel to its neutral-sized (resting, 0°C) state. By changing the size of specific voxels rythmically, the robot is able to translate and rotate its body.

The touch sensory neuron activation was set to 1 whenever the voxel was touching the ground (as determined by the *Voxelyze* engine API), and 0 otherwise. The sine-wave generator set its neuron activation to a time-varying value in $[-1, 1]$ according to the formula $sin(C * 2\pi * t)$, where t is the current time in seconds, and C is a coefficient that modulates the speed of oscillation (here $C = 3$). The sine-wave generator is meant to assist the robot in pumping its voxels rhythmically, a key component of SVR locomotion. Touch sensor and sine-wave generator only used the first sensory neuron, leaving the second sensory neuron inactive, since those sensor types each have one input value only.

The vision sensory voxel operated by raycasting out 5 rays, 1 straight out from the center, 4 angled from the center by 2 degrees (see Fig. 2), similarly to the SVR raycasting scheme in 2D by [6, 16]. Having multiple rays at various angles allowed the robot to capture a larger area in its visual field than a single ray could provide. Each ray had a range of $R = 15$ units. Whenever a ray detected a food or obstacle at a distance d away, the closeness C of the object was set as the appropriate sensory neuron's activation, where $C = \frac{R-d}{R} = 1 - \frac{d}{R}$. If multiple rays struck the same object, the ray that was closest (i.e., highest C value) was used. The vision sensor utilized both sensory neurons in a voxel: the first neuron was activated when a raycast detected food, the second neuron was activated when a raycast detected an obstacle.

3.2 Evolutionary algorithm

Rather than explicitly segregating different generations of animats, a "steady state genetic algorithm" similar to the one in *PolyWorld* [25, p 4] was utilized, such that there was a steady population of living animats at all times, a minimum number of animats were guaranteed to populate the world at any given time. The minimum number of living animats was enforced by spawning a new animat whenever a death occurred and the population dropped below the minimum threshold $T = 20$.

Animats were spawned/born in 1 of 3 ways: (1) creation from scratch, (2) asexual reproduction, or (3) sexual reproduction. A "reproductive pool" of maximum size $P = 50$ was maintained, to allow probabilistically selecting animats for reproduction based on their fitness score F. The pool was a FIFO queue, to which an animat was added when it died. Animats were generated from scratch until the pool reached size P, after which animats were only generated via reproduction from the pool. When an animat died, a new animat was generated via reproduction, with a 50%/50% chance for asexual/sexual reproduction.

Animats could die in 1 of 2 ways: (1) old age, or (2) touching an obstacle. Each animat had an age A, representing the maximum number of seconds it could remain alive. A was initialized to a random number in range [25, 30] when the animat was born. Death by old age occurred when $A \leq 0$. Death by obstacle occurred instantly whenever the animat's position overlapped an obstacle's position.

Animats could eat a food block by detecting it in their raycast sensor at a distance of 0.3 units or less. Eating a food restored the animat's age to the maximum value $A_{max} = 30$.[1]

An animat's fitness score F measured its performance in the arena, and determined its chances for reproduction. A higher F score meant the animat had a higher chance to be selected from the reproductive pool for reproduction. F was computed as a function of food eaten E:

$$F(E) = \begin{cases} min(\frac{B}{R}, 1) & \text{if } E = 0 \\ E + 1 & \text{if } E > 0 \end{cases} \quad (1)$$

where B is the distance that the animat travelled from its birthplace.

This fitness function acts to evolve the animat's behavior progressively from idleness to locomotion to food-seeking. Since the initial animat population will not move at all, or at least move randomly, they are unlikely to successfully eat any food. Therefore, an animat who ate zero food ($E = 0$) was scored with a fitness in [0, 1], dependent on how far it had moved from its birthplace. This encouraged the evolution of simple locomotion behavior. Otherwise, if the animat managed to eat a non-zero amount of food, then its score was determined as $F = E + 1$, thus an agent's reproductive fitness was directly proportional with a number of food eaten. This was intended to encourage the evolution of better food-seeking and food-eating behavior. In the case that an animat died by touching an obstacle, its number of food eaten E was set to zero, and its F scored accordingly.

[1] Technically the ability to infinitely replenish age means an animat who was extremely fit would never die and therefore would never enter the reproduction pool; in practice, this was not a major issue, as most animats were lower fitness and eventually died, becoming eligible for reproduction. However, this limitation is worth mentioning.

Something to keep in mind is that many of the variables and hyperparameters here can significantly impact the simulation, and thus the results. Such variables include the *Voxelyze* parameters, the raycast vision distance R, maximum age A_{max}, chance for sexual/asexual reproduction, creature size, etc. The problem is we can set any arbitrary value for them (or at least select from a wide range of values), we must manually decide them or possibly automatically adapt them. Here, the values were selected and tuned manually, based on what seemed through experimentation to produce a reasonable simulation and situation that could (and ultimately did) allow for the evolution of food-seeking behavior.

3.3 NeuroEvolution setup

The neural networks were of the simple "sum-and-squash" type. Connections had static weights, and each neuron applied the *tanh* activation function (with activation in range $[-1, 1]$) to the sum of weighted inputs. The timestep for updating the network was every 0.04 s.

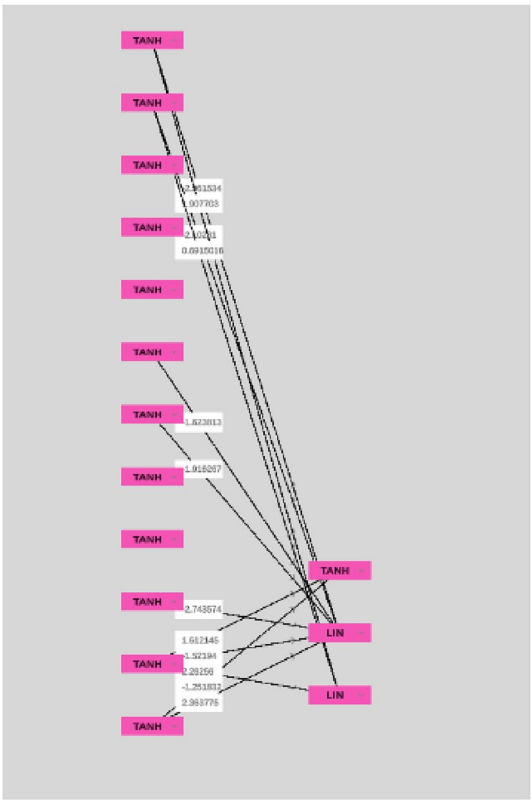


Fig. 3 An example of the CPPN genome. The 12 nodes on the left are the inputs and the 3 nodes on the right are the outputs

3.3.1 HyperNEAT

I used my own implementation of HyperNEAT for evolving weights, according to the genetic encoding described in [18] (without speciation), with a Link-Expression Output (LEO), which allows disabling connections entirely as proposed in [23] (Fig. 3).

For the network topology, I tried many different configurations. In one attempt, I adopted a similar topology as suggested in [14] of embedding each voxel of the robot with a multi-layer perceptron (MLP), including the transferring of signals between spatially adjacent MLPs. The MLP topology was the same in each voxel, though different MLPs could have different weights depending on the CPPN outputs. For the data captured in Section 3.4, there were no constraints on connectivity (i.e., the neurons were fully connected but allowed to be disabled by the LEO).

The topology of the network could not evolve, only the weights on connections between neurons, and certain characteristics of neurons. In this way, each voxel contained all 3 types of neurons in a fixed number: sensor, motor, and hidden. In addition to the two sensory neurons and three motor neurons per voxel, the MLP consisted of three hidden layers with four neurons per layer (see Fig. 4).

A voxel's position was described by 3D coordinates (x, y, z). A neuron's position within such a voxel could be described using two additional coordinates: w, the layer of the MLP (where $w = 0$ is the input layer), and v, the neuron's index in the layer (where $v = 0$ is the bottom neuron).

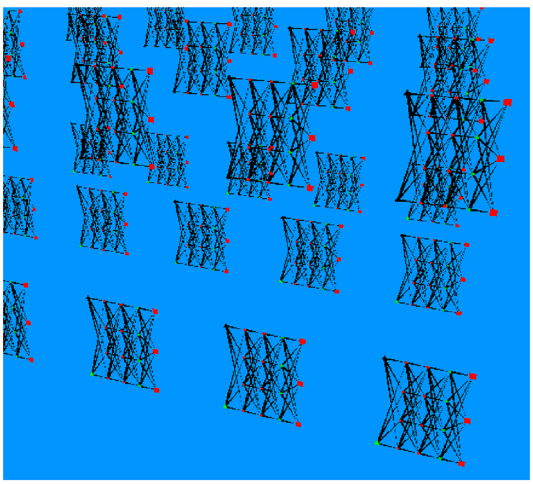


Fig. 4 A part of the HyperNEAT neural network distributed throughout the voxel robot. In this image, the connectivity was constrained to force a multi-layer perceptron (MLP) in each voxel: sensors on the left, motors on the right, hidden layers between them, and only feed-forward connections. Triangles are sensory neurons, squares are motor neurons, circles are hidden neurons. Connections between voxels not shown for clarity

HyperNEAT works by evolving a compositional pattern-producing network (CPPN). Given the input coordinates (normalized to $[-1, 1]$) of 2 neurons, the CPPN outputs the weight value on the connection between those neurons. As the CPPN evolves, the function computing these weights changes.

The CPPN in this case had 12 inputs: 1 bias node (that always outputs value 1), the 5 normalized coordinates of the first neuron $(x_1, y_1, z_1, w_1, v_1)$, the 5 normalized coordinates of the second neuron $(x_2, y_2, z_2, w_2, v_2)$, and the distance of the first neuron to the center of the substrate. The CPPN had three outputs: neuron bias, connection weight, and LEO. The neuron bias value was determined for the first neuron whenever all the (non-normalized) coordinates of the second neuron were equal to zero. When LEO had a value greater than zero, the connection was expressed in the neural network; otherwise, it was disabled.

CPPN mutation occurred during asexual reproduction only, as follows. There was a 5% chance to add a new connection. There was a 3% chance to add a new node, with a randomly selected CPPN function. Each connection weight had a 90% chance of being mutated. If it was to be mutated, there was a 90% chance to increment the weight by a random value from $[-0.1, 0.1]$, or a 10% to change the weight to a random value from $[-3, 3]$. Crossover was performed as in the standard algorithm. During crossover (sexual reproduction), a disabled connection had a 25% chance of becoming enabled again. These mutation rates are in line with those described in the NEAT paper [19].

The available CPPN functions were: Linear, Sigmoid, Gaussian, Sine, Square, Absolute Value, Step, Hypertangent, Exponential, Cosine, Square Root, Sawtooth, Triangle, ReLU, and Natural Log. This selection was meant to give the algorithm a combination of symmetric and asymmetric functions to work with, depending on what might be useful.

3.3.2 NEAT

I used my own implementation of NEAT for evolving weights and network topology, according to the genetic encoding described in [19] (without speciation).

The initial network topology contained no hidden neurons. The only initial connections were between sensory neurons and motor neurons within each respective voxel (see Fig. 5). However, connections and hidden neurons could be added to the network during evolution, such as to allow communication between voxels.

NEAT genome mutation occurred during asexual reproduction only, as follows. There was a 15% chance to add a new connection; when a new connection was added, it could come from any random neuron and synapse into any random hidden or motor neuron. There was a 9% chance to add a new node. Each connection weight and neuron bias had a 80% chance of being mutated. If the connection/neuron was to be mutated, there was a 90% chance to increment the weight/bias by a random value from $[-0.1, 0.1]$, or a 10% to change the weight/bias to a random value from $[-3, 3]$. Crossover with performed as in the standard algorithm. During crossover (sexual reproduction), a disabled connection had a 25% chance of becoming enabled again. These mutation rates are in line with those described in the NEAT paper [19].

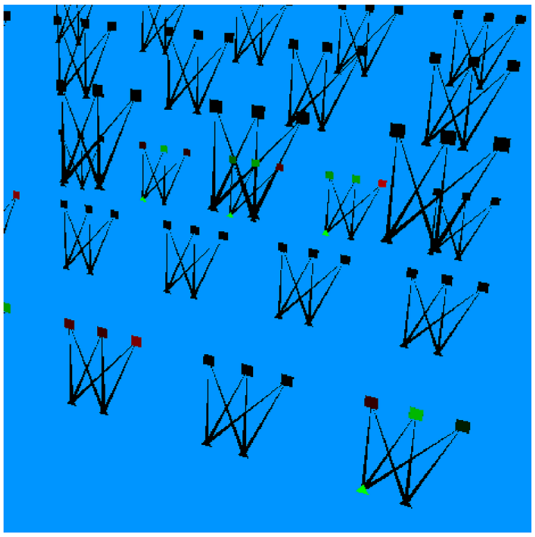


Fig. 5 A part of the initial NEAT neural network distributed throughout the voxel robot. The initial connectivity of the neural networks was to fully connect each input of voxel *V* to each output of voxel *V*. Triangles are sensory neurons, squares are motor neurons

3.4 Results

A table of the top 50 animats of all time (aka the "elite set") was maintained, from which quantitative data was recorded every 50 seconds (simulation time) for analysis. In this way, we can use performance metrics such as "number of food eaten" to analyze if the evolutionary search is finding better solutions over time and improving the elite set, or if the search is stagnating.

The trial with food and obstacles was run for 200 fifty-second (i.e., $200 \times 50s$) intervals (see Fig. 7), which is around 2.75 hours. The trial with food only was run for only 150 fifty-second (i.e., $150 \times 50s$) intervals (see Fig. 6), around 2 hours per trial (cut off early since the ideal behavior had already emerged). Note that the simulation ran in real-time (or close to it), so these times are approximately the same as the "wall-clock time".

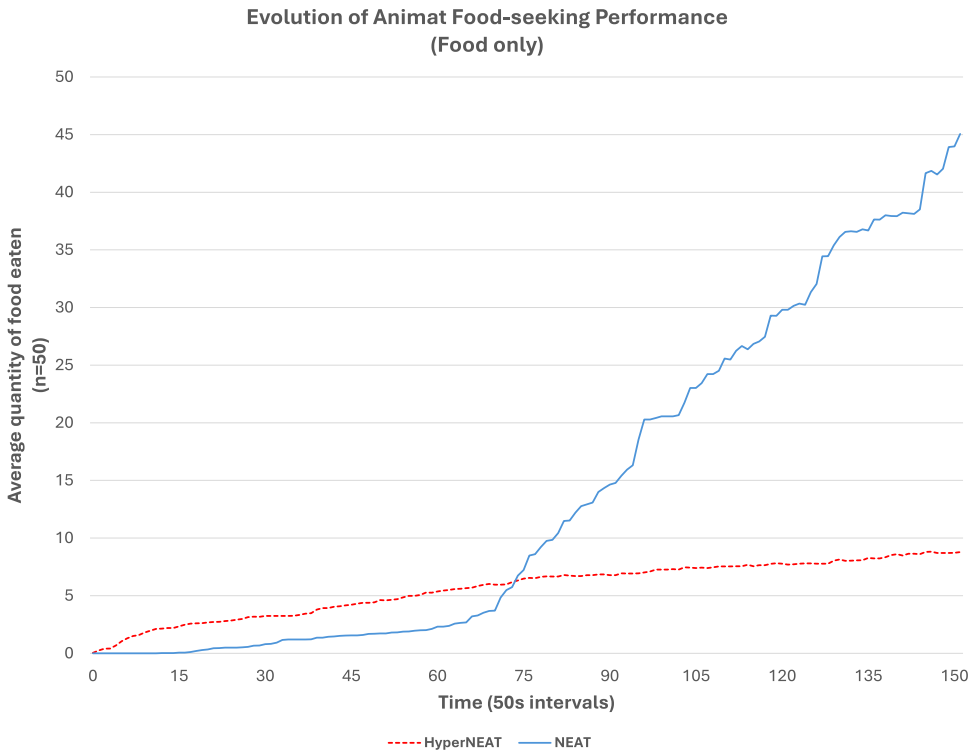


Fig. 6 Elite animat performance in the trial with food only (no obstacles). The vertical axis has a range of 50

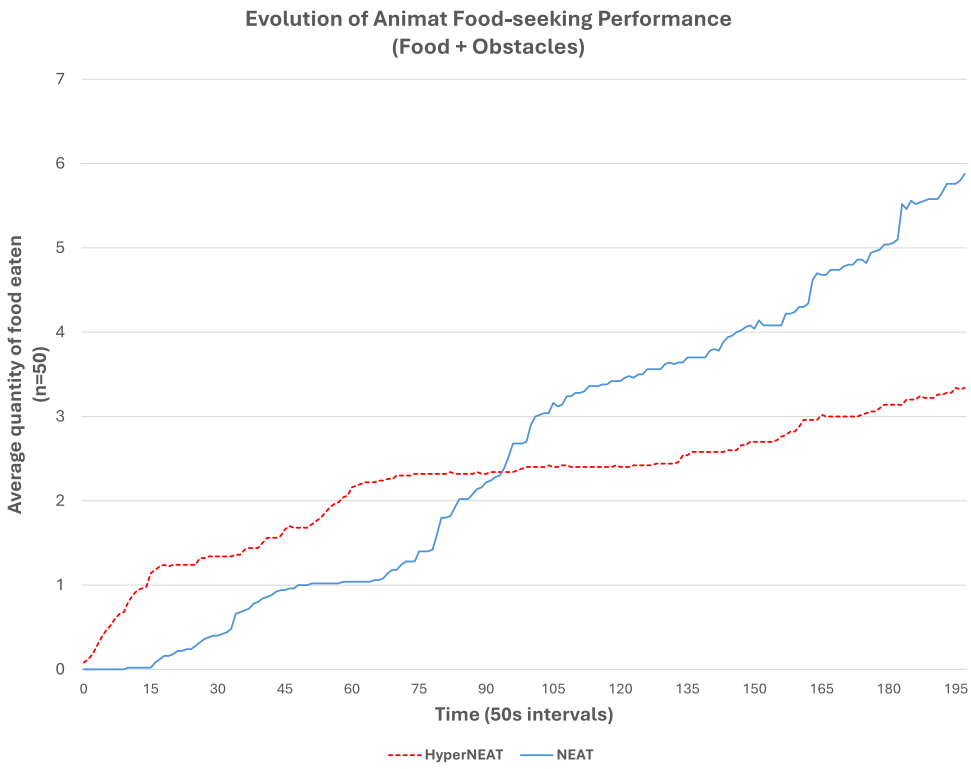


Fig. 7 Elite animat performance in the trial with both food and deadly obstacles. Note the vertical axis has a range of 7, showcasing the difficulty introduced to animats by the deadly obstacles

3.4.1 HyperNEAT

Quantitatively, for HyperNEAT, the average number of food eaten by the best animats increased very slowly in the food-only scenarios (see Fig. 6). In the food+obstacle scenario, the average number of food eaten increased rapidly for a short time, then plateaued for the rest of the simulation (see Fig. 7). We can see from the graphs that HyperNEAT

initially outperformed NEAT in the early stages of evolution, but NEAT drastically outperformed HyperNEAT later.

Qualitatively, the animats resulting from HyperNEAT hopped around erratically, and did not seem to use their sensors at all. This was especially evident with the vision sensor, which was rapidly flailed around without aim along with the body. In the neural network, the hidden and motor neurons rapidly flickered between the extreme positive and negative activations. They did evolve to become better at flailing themselves over time, going further and rotating faster, and usually moving in a circular fashion. This behavior sometimes incidentally resulted in them eating a surprisingly significant amount of food, which explains the data of HyperNEAT outperforming NEAT in the beginning. However, the performance never progressed far, since the behavior was far too erratic and imprecise to focus on any food items. So, in the case of HyperNEAT, successfully eating food was primarily based on luck. Food-eating was only achieved by rapid random-like locomotion and rotation, which caused the animat to run into food only by happenstance.

We can begin tracing the issues in animat behavior by examining the neural network. During operation, visualization of the neural network showed many of the hidden and motor neurons rapidly flickering between the extreme activation values, from −1 to 1 and back again. There appeared to be no subtlety or coordination in information transfer. This could be caused by the large number of connections present in a HyperNEAT setup since a neuron with many input connections would receive a large-magnitude value from summing its many inputs. A large magnitude of input to a neuron would result in extreme values resulting from the *tanh* activation function.

I ran many different setups in an attempt to make HyperNEAT evolve the desired behavior. This included various setups of neural topology, *Voxelyze* parameters, and HyperNEAT parameters, which are not listed here, for brevity and since none of them achieved positive results. All my setups led to the same negative outcomes described here, suggesting an issue with my use of HyperNEAT, my implementation of HyperNEAT, or HyperNEAT itself.

3.4.2 NEAT

Quantitatively, the average fitness of the best animats continuously and rapidly improved in both the food-only and food+obstacle scenarios (see Figs. 6,7).

Qualitatively, the best animats resulting from NEAT exhibited the "expected" behavior almost perfectly. When their vision sensor detected nothing, they moved forward and spun around, as if searching for food. When their vision sensor did detect a food block, the best animats would immediately charge straight towards the food.

Though this behavior is perhaps simple enough it could even be manually programmed in a simple robot, the important result is that the behavior was evolved. Consequently, at least theoretically, other more complex behaviors could develop on top of this behavior as evolution progresses. This would be especially interesting to study if the animats are placed in higher-complexity environments, though further experiments are still needed.

There was an interesting element of competition observed between animats, which may have contributed to speedy evolution. When two animats targeted the same food block, they had the appearance of racing each other. In fact, they were in a sort of meaningful competition for first place, since the winner of the race had its age *A* reset, allowing it plentiful more time to seek out other food blocks, whereas the loser(s) did not get that benefit. This may have helped to encourage the evolution of animats which were very quick and precise with their locomotion, since such animats would be more likely to win "races" against the other competitor animats.

While the best NEAT animats were excellent at food-seeking, they appeared to struggle somewhat with the secondary objective of avoiding obstacles (as shown in the disparity of NEAT scores between Figs. 6 and 7). No metrics were recorded for analyzing the specifics obstacle avoidance, though weak avoidance behaviors were observed.

4 Future directions

The results observed here corroborate the data observed in some experiments by [1, p 7-8], who suggest a possibly inherent issue in HyperNEAT when it comes to complex problems: "CPPN-NEAT generally performs well on locomotion tasks, as tested by previous works, but performs poorly on more complex manipulation tasks. This is possible because NEAT favors CPPNs with simpler structures, which encourage CPPNs to generate robots with more regular patterns. However, to succeed in complex manipulation tasks, some agile substructures of the robot must evolve, which might only exist in robots with irregular patterns".

While it may be theoretically possible for HyperNEAT to solve this problem (i.e., there exists a "correct" CPPN for the problem), it may be that in practice the encoding struggles to represent fine-tuned/irregular networks. On the other hand, the problem may stem from factors unrelated to HyperNEAT, such as using a traditional "objective-based" evolutionary search algorithm (compared to potentially better algorithms such as novelty search [13]). There were also a very large number of inputs (12) provided to the CPPN here, moreso than is usually provided to CPPNs in previous works; therefore, another problem may be that the CPPN encoding has difficulty handling so many inputs.

This work leads to a few different possible directions for future works.

One future direction is to investigate whether HyperNEAT really struggles with this problem in particular, or (more likely) if its simply due to the setup. This is important because HyperNEAT has been frequently used for the evolution of SVRs, usually morphologies, but sometimes brains too. It can produce interesting patterns and regularities, and we might want to try it in artificial animal expeiriments. We could begin such an investigation into HpyerNEAT a few ways. First, we could try the same experiment with HyperNEAT but in simpler robots, such that HyperNEAT has an easier chance to succeed. Second, we could try a new experiment using "novelty search" to escape local optima and explore a wider variety of CPPNs, instead of using the traditional objective-fitness-based evolutionary search. Finally, though perhaps most strenuous, we could directly examine the topology and activation dynamics of successfully evolved NEAT networks, then compare them to evolved HyperNEAT networks.

Secondly, we can easily extend the food-seeking behavior evolved here to mate-seeking behavior, which would thus enable autonomous sexual selection in SVR animats. If we can evolve SVR animats to exhibit food-seeking and mate-seeking behaviors, we can subsequently place them in a simulated ecosystem for us to observe their self-sustaining SVR animat populations. We could perform experiments in such ecosystems, analyzing and improving the animats' brain evolution, such as their capacity for real-time learning and intelligent complex behaviors. There is also the opportunity to pair morphological evolution with neuroevolution (aka brain-body co-evolution).

5 Conclusion

In conclusion, the results of the experiment indicate that NEAT, and not HyperNEAT, successfully evolved vision-based food-seeking behavior in 3D soft voxel worm robots.

NEAT was able to evolve neural networks for precise and deliberate food-seeking behavior. The algorithm started with behaviors that had very low, near zero fitness, such as idling, but gradually improved food-seeking behavior over time. Interestingly, the improvement was continuous, with better individuals popping up all the time, however, the trials were cut off since the desired behavior had evolved.

On the other hand, none of the experiments with HyperNEAT produced precise food-seeking behavior. While the results show that HyperNEAT had a slight advantage during the beginning generations, this was primarily due to the fact that initial HyperNEAT-controlled robots were flailing around, whereas initial NEAT-controlled robots stayed completely still. Once the NEAT networks evolved to organize and act "reasonably", they quickly began to outperform the random flailing of the HyperNEAT networks.

Acknowledgements This research was partially sponsored by CleverLibre, a free/libre and open source software incubator promoting ethics and open standards in the advancement of AI. The experiments were built with Unity game engine [11] and the soft voxel engine *Voxelyze* [10]. Thanks to Professor Pei Wang from Temple University for his comments.

Data and code availabilty A video of the evolved food-seeking behavior is available at: https://www.youtube.com/watch?v=T6kj7PIbLs0. The code is available on GitHub: https://github.com/ccrock4t/AnimatCreator/tree/c3fa1a20d4aefea6c4ae6017ea1dd24916e76a07. The code is archived on Zenodo: https://doi.org/10.5281/zenodo.14064025. The data for the graphs is at: WormbotData/. The C# code is at: Assets/Scripts/. To run the experiment, open the project directory in Unity and press play (alternatively, download the release package and run the. exe). As the experiment runs, the data will be periodically written to a file at: Assets/ExperimentDataFiles/AnimatArena/arena_score_data. txt.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Bhatia J et al (2021) Evolution gym: a large-scale benchmark for evolving soft robots. Adv Neural Inf Process Syst 34:2201–2214
2. Cheney N et al (2016) On the difficulty of co-optimizing morphology and control in evolved virtual creatures. Artificial life conference proceedings. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ... pp 226–233
3. Cheney N et al (2014) Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. ACM SIGEVOlution 7(1):11–23
4. Clune J (2009) Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. iEEE congress on evolutionary computation. IEEE. 2009:2764–2771
5. Corucci F et al (2018) Evolving soft locomotion in aquatic and terrestrial environments: effects of material properties and environmental transitions. Soft Rob 5(4):475–495
6. Ferigo A, Medvet E, Iacca G (2022) Optimizing the sensory apparatus of voxel-based soft robots through evolution and babbling. SN Comput Sci 3(2):109
7. Ferigo A et al (2022) Evolving Hebbian learning rules in voxel-based soft robots. IEEE Trans Cognit Dev Syst 15(3):1536–1546
8. Hahm C (2022) Designing naturalistic simulations for evolving AGI Species. International conference on simulation and modeling methodologies, technologies and applications. pp 296–303
9. Hahm CG (2024) Evolving Hebbian neural networks for articulated robot locomotion. Tech. rep. 20. Temple University, Department of Computer and Information Sciences, AGI Team

10. Hiller J, Lipson H (2011) Automatic design and manufacture of soft robots. IEEE Trans Robot 28(2):457–466
11. Juliani A et al (2018) Unity: a general platform for intelligent agents'. arXiv preprint arXiv:1809.02627
12. Kriegman S et al (2017) Simulating the evolution of soft and rigid-body robots. Proceedings of the genetic and evolutionary computation conference companion. pp 1117–1120. https://github.com/skriegman/evosoro
13. Lehman J, Stanley KO (2011) Abandoning objectives: evolution through the search for novelty alone. Evol Comput 19(2):189–223
14. Medvet E et al (2020) Evolution of distributed neural controllers for voxel-based soft robots. Proceedings of the 2020 genetic and evolutionary computation conference. pp 112–120
15. Mertan A, Cheney N (2024) Investigating premature convergence in co-optimization of morphology and control in evolved virtual soft robots'. European conference on genetic programming (Part of EvoStar). Springer. pp 38–55
16. Pigozzi F, Medvet E (2022) Evolving modularity in soft robots through an embodied and self-organizing neural controller. Artif Life 28(3):322–347
17. Sims K (1994) Evolving virtual creatures. Proceedings of the 21st annual conference on computer graphics and interactive techniques. pp 15–22
18. Stanley KO, D'Ambrosio DB, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15(2):185–212
19. Stanley Kenneth O, Miikkulainen Risto (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127
20. Stanley KO et al (2019) Designing neural networks through neuroevolution. Nat Mach Intell 1(1):24–35
21. Strannegård C et al (2021) The ecosystem path to AGI. International conference on artificial general intelligence. Springer. pp 269–278
22. Fabio T, Claus A (2022) Co-evolving morphology and control of soft robots using a single genome. In: 2022 IEEE symposium series on computational intelligence (SSCI). IEEE. 1235–1242
23. Verbancsics P, Stanley KO (2011) Constraining connectivity to encourage modularity in HyperNEAT. Proceedings of the 13th annual conference on Genetic and evolutionary computation. pp 1483–1490
24. Wilson Stewart W (1985) Knowledge growth in an artificial animal. Proceedings of the 1st international conference on genetic algorithms. USA: L. Erlbaum Associates Inc., pp 16–23. isbn: 0805804269
25. Yaeger L al (1994) Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or Poly World: life in a new context". Santa fe Institute Studies in the sciences of complexity-proceedings. Vol 17. Citeseer, pp 263–263

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.