Experiments in Advancing the Evolution of Artificial Animals

Christian Hahm
christian.hahm@temple.edu

Department of Computer and Information Sciences
Temple University

May 5, 2025

Abstract

This paper details the results of various experimental setups for the evolution of artificial animals. Various frameworks in mind (AI), body (robots), environment, and evolutionary algorithm were tested. For the mind, continuous-time (CTNN) and sum-and-squash neural networks were both tested, with and without Hebbian learning, as well as the non-axiomatic reasoning system (NARS) without learning. Frameworks tested for the body included wheeled robots, articulated robots, and soft voxel robots. For the environment, a static flat plane world and interactive voxel world were both tested. The results show the evolutionary simulation's ability to produce AI artificial animals (animats) which are capable of autonomously upkeeping their own energy stores and reproducing, both sexually and asexually. The artificial animals were autonomously able to maintain, at their best, up to nearly 50% of their population without external support, though the performances varied greatly depending on the given simulation configuration. A few interesting results are reported. Sum-and-Squash neural networks with Hebbian ABCD learning outperformed the other AI methods. Soft voxel robots outperformed articulated robots. NARS was successfully evolved for the first time. Soft voxel robots were evolved in animat ecosystems for the first time. Finally, for the environment, an interactive cellular automaton voxel world was discovered to challenge the animats more than a flat world, the usual choice in animat simulations.

Keywords: *artificial animal, AI, evolution, voxel, neural network, reasoning*

Contents

1 Introduction

Though most AI projects today seek to emulate some aspects of human-level intelligence, there is a small movement in the overlap of the fields of AI and Artificial Life (AL) with the goal to simulate animal-level intelligence, and then to build upon it towards holistic human-level intelligence. After all, animals are intelligent too, even though they are not as intelligent as humans. The "artificial animal" approach, or "animat" approach, as coined by [Wilson, 1986], is an interesting alternative approach to AI that seeks to simulate AI agents which act like animals and evolve them into increasingly intelligent animals. This approach creates AI agents which are animal-like, rather than human-like.

This might seem curious, since it is sometimes claimed that human-level AI, or at least the basis for it, has already been invented in the modern day with the large language models (LLMs) and transformer models [Bubeck et al., 2023]. It is true animal-like AI systems are not necessarily going to be useful or productive to human society in the same ways the human-like AI systems are, nor will they be immediately incredible at generalization like humans are and outperform the human-level AI systems of today in all areas. But, animal AI should still be attractive to us in AI research and cognitive science for a variety of reasons.

Firstly, since animal behaviors are simpler than human behaviors, their artificial minds can be made simpler too. This provides us an opportunity to dissect and analyze simple yet effective neural network topologies and dynamics, to learn from them and make our neural networks more "explainable". The simplicity also provides a rich opportunity to try more realistic and underexplored neural network abstractions, such as Hebbian learning and spiking neurons, in their original naturalistic context. We can also use the evolution of artificial animals to "scale up" symbolic logical AI methods, which is a historically unsolved problem that drove much of the shift from symbolic AI methods to neural network AI [Russell and Norvig, 2016, p.22]. The paradigm shift subsequently resulted in the AI "explainability crisis" whereby it is now difficult or impossible to explain the reasons behind our most advanced AI's behavior [Gunning and Aha, 2019], including the wildly popularized LLMs, which often make basic logical errors [Arkoudas, 2023, Mirzadeh et al., 2024, Nezhurina et al., 2024].

Secondly, we know that in the natural world, human-level intelligence was "created" via an evolutionary process by building on top of lower-level animal intelligence. It seems reasonable to think the artificial animal method could similarly be a path to "human-level AI", and/or "artificial general intelligence". This "ladder of intelligence" approach, starting from simple animal-like AI and building up, has been argued as a useful and potentially necessary paradigm to achieve truly intelligent and conscious machines, by [Wilson, 1991, Yaeger et al., 1994, Adams and Burbeck, 2012, Crosby, 2020, Strannegård et al., 2021, Hahm, 2022] and others. The fact that animals share an ecosystem where they learn, interact, and evolve together is not to be trivialized as an unnecessary part of nature, but viewed as a necessity and part of holistic intelligence. Animal socialization and culture are important in the evolution of intelligence and in the learning process, since animals can teach each other, transferring knowledge from one animal to another, and effectively store that knowledge over many generations [Adams and Burbeck, 2012, p.58-63]. The "arms race" phenomenon [Dawkins and Krebs, 1979] is also critical to consider, whereby the evolution of one species, such as a wolf's increased ability to track animals, pressures the evolution of another, such as a rabbit's ability to camouflage and run faster.

Thirdly, by using animals as our standard of intelligence, we have countless real-world examples of intelligence to compare to, a class of intelligent entities that is much broader than humans alone. We can compare the artificial animals to real animals, such as their behaviors, their brain architectures, their dynamics, and even perform similar psychological tests to measure their intelligent abilities in the same as we do with real animals. For example, [Adams and Burbeck, 2012, p.51-52] considered the adaptive nature of the octopus, such as its ability to learn how to open jars and disable electric lights, in the context of general intelligence, and [Niv et al., 2001] compared the behavior of their evolved neural network agent, that utilized Hebbian learning and reinforcement learning, to the foraging behavior of bees.

Finally, the use of evolutionary algorithms (EAs) has some benefits over gradient-descent-based methods. Evolutionary algorithms can explore the fitness function landscape via "neutral mutations" (i.e., mutations which do not change fitness) [Lehman and Stanley, 2011, p.8] and even mutations which are initially harmful but later prove to be useful. Gradient-descent-based methods, on the other hand, get stuck when the gradient is zero, and when it is not, they must follow the gradient to some locally optimal point, after which the learning usually "converges" and the network is very unlikely to reach a more globally optimal point. Also,

gradient descent and backpropagation are limited in that they only work for specific types of layered neural networks, while evolutionary algorithms can evolve any arbitrary framework, even multiple frameworks at the same, such as other types of AI and robot morphologies.

When compared to reinforcement learning (RL) in AI in general, EAs do not require a "state space" for their inputs or an "action space" to select their outputs, whereas RL does, though both methods use a concept of "reward" or "fitness". The major difference is that the RL algorithm itself explicitly tracks actions and states, and tries to assign credit to earned rewards, by learning a policy function that directly maps specific states to specific actions, whereas an evolutionary algorithm does none of this explicitly, only modifying an implicit "policy", the neural network weights, over multiple generations in a blind attempt to increase the reward. The RL algorithm tracks how the agent did something that led to a reward, whereas the EA only tracks the reward itself. This makes EA a much simpler approach, and also EA can escape the trap of local optima by evaluating a multitude of solutions across the fitness landscape, whereas RL is more likely to get stuck in local optima than EA because it only explores one solution at a time, which is also true of most gradient descent approaches. Although, it is possible to combine both approaches to leverage their strengths, e.g., as in [Niv et al., 2001, Weel et al., 2014, Strannegård et al., 2020].

The evolved animat approach is not only relevant to AI, but falls directly under the fields of artificial life and evolutionary algorithms. Furthermore, this idea is related to many other fields including sociology, ecology, psychology, perception, ethology, physiology, metabolism, robotics, etc. Consequently, it is a heavily interdisciplinary approach to AI, and its study can be enhanced by every field related to cognitive science and more. For example, simulations can be enhanced by a rich knowledge of seemingly unrelated fields like physics and chemistry.

The animat idea has been tested many times before in various limited capacities. The field's goal is to take the idea to its limits, achieving ever richer simulations to hopefully evolve increasingly complex and intelligent animals [Lehman and Stanley, 2011, p.8-9]. The problem may seem daunting at first due to its massive scope, but at its core, simulating artificial animals requires only 3 fundamental components: 1.) *mind*, 2.) *body*, and 3.) *environment*. To adapt the animats over generations requires us to consider a 4th category, 4.) *evolution*, though its consideration is mostly independent of the individual animat components. Therefore, in pursuing the scientific advancement of evolutionary animat ecosystems, we want to perform experiments in advancing one or more of these categories, and especially to compare their combinations, in a full-blown evolutionary animat ecosystem. From there, we can decide what techniques showed the most promise, to focus on them in the future, refine them, and advance the research further.

The purposes of the 3 animat components are as follows. The *mind* is the cognitive algorithm that controls the animat; it coordinates the body's sensorymotor signals and directs the body towards its goals. The *body* is a robot, the physical manifestation of the animat in world space; it is necessary for the animat to exist at all, and provides the sensorymotor interface between the mind and environment. The *environment* is the world space containing all the animat bodies, the terrain, and the physical objects with which the animats can interact.

The 3 animat components interface with each other in a feedback loop. The mind controls the body's motors to affect the environment. The body captures sensory information from the environment and channels it to the mind for processing. Completing the loop, this processing informs the mind's future decisions for controlling the body. This loop constantly cycles. In this way, the mind never directly interfaces with the environment; instead mental signals and environmental signals are channeled bidirectionally through the body. So, the importance of the richness of these 3 components cannot be understated. The mind determines everything about the agent cognitive capacities. The body is the only lens through which the cognitive agent can view and interact with the world. The environment provides the challenges and pressures through which the agents live and adapt.

This work presents the results of some experiments testing state-of-the-art techniques in artificial animal evolutionary ecosystems, for mind (AI), body (robots), environment, and evolution.

2 Related Works

The original coining of the term "*animat*", short for artificial animal, was in the mid-1980's by Stewart Wilson [Wilson, 1986]. That work presented the results of modeling artificial animals, albeit in a extremely

simple world, which was 2D and text-based, without graphics. The experiment tested for how long it takes the animat to find food, and took a measure of generality regarding the animat's behavior. [Wilson, 1991] formalized "*a research methodology for understanding intelligence through simulation of artificial animals ('animats') in progressively more challenging environments while retaining characteristics of holism, pragmatism, perception, categorization, and adaptation that are often underrepresented in standard AI approaches to intelligence*". The main idea is that "*the methodology [of a simulation] should include a theory/staxonomy of environments by which they can be ordered in difficulty... and a theory of animat efficiency*". Wilson criticized "Standard AI" and approaches to AI that attempted to explicitly model isolate human competencies, arguing that most AI systems ignore the fact that real creatures are always embodied in sensory environments and have varying needs which must be satisfied [Wilson, 1991, p.15].

There were two artificial life projects in 1994 that pushed the state of the art in simulating artificial animals.

The first flagship project to mention is Karl Sims' *virtual creatures* [Sims, 1994], which is emulated and referenced to this day. This work evolved various articulated robots, both their mind and body, for locomotion and movement tasks. Especially of note is the fact that both mind and body were evolved together, it took place in a realistic 3D simulation, and it resulted in a very eye-catching rhythmic and naturalistic motion of robots. However, despite being called "virtual creatures", the robots did not perform any behaviors except those related to locomotion, which is a somewhat trivial task, especially depending on your criteria for "locomotion". The mind was also not a traditional neural network, but a network of mathematical functions. Also, these robots were evolved in isolation or at most with 1 other creature, not in a massive shared ecosystem where they could interact.

The second flagship project to mention is *PolyWorld* by Larry Yaeger [Yaeger et al., 1994], which may be the first "full-scale" 3D animat evolutionary ecosystem that was created. The project accounted for all the major components one would expect in an evolutionary animat simulation, at least to a basic level: evolution, multi-agent ecosystems, body evolution, brain evolution, vision, learning, natural selection, and sexual reproduction. This was not a 2D highly abstract world such as was used in past animat simulations, but a 3-dimensional physical environment, albeit a very simple one. The animat bodies were like wheeled robots (see *Section 3.2.2*), and the brains were layered networks which operated with Hebbian learning.

While *PolyWorld* is an extremely impressive project for its time, and fully fleshed out for its capacities, it is still limited overall. Firstly, the simulation runs incredibly slow; 13 seconds per time-step in the original simulation [Yaeger et al., 1994, p.5], and my own tests running *PolyWorld* also yielded a very slow simulation with a few seconds between each timsep. Secondly, though the world is simulated to look 3D, the animats cannot move up or down, they cannot see up or down, and they live on a flat plane, so the world is effectively fully 2D. Finally, only wheeled robots were tested, not complex robots such as to replicate Sims-type robots. Though Yaeger acknowledged these limitations: "*Perhaps extensions to PolyWorld, or the next researcher's ALife environment will successfully evolve more intelligent, more obviously alive organisms*" [Yaeger et al., 1994, p.18]. Yaeger also expressed interest in evolving other types of cognitive and learning algorithms [Yaeger et al., 1994, p.3,19].

Modern animat ecosystem projects have taken inspiration from these historic ideas and tested more realistic and richer simulations.

[Weel et al., 2014] created a 3D ecosystem of AI robots with evolvable minds and bodies. The ecosystem was simulated in the *Webots* simulation software. The robots, called "Roombots", were made of modular spheres or rounded blocks, and they could rotate at the joint between two modules. The environment was a flat circular plane surrounded by tall walls. The AI used to control the robot utilized reinforcement learning whose "policy" was evolved, in addition to the modular morphology which also evolved. The robots were evolved for reproduction and locomotion; reproduction occurred whenever two Roombots got close in range, during which a broadcast signal sent the genome to nearby Roombots who could then sexually reproduce using it after a short time delay. There was no fitness function, rather the simulation utilized entirely "implicit fitness" such that reproduction did not occur from a central evolutionary algorithm, but only from autonomous sexual reproduction.

[Strannegård et al., 2021] introduced the photorealistic 3D animat simulator *EcoTwin*, which simulated the evolution of animal cognition plus real-time learning. *EcoTwin* was built in the *Unity* engine. The robots moved like wheeled robots (see *Section 3.2.2*), though they were modeled and texture photorealistically to match real animals such as wolves, deer, and goats. Earlier designs of the *EcoTwin* animat brain used neu-

rosymbolic/logical graphs paired with a form of reinforcement learning [Strannegård et al., 2017, Strannegård et al., 2020, Strannegård et al., 2023, p.25], while later versions used 3 networks: a happiness network for motivation, a direct reflex network for reflexes, and a policy network that learns via reinforcement learning [Strannegård et al., 2021, p.5]. The reflex and happiness networks were evolved, while the animats were able to learn in real-time by updating the policy network at each time step with reinforcement learning. This was achieved by calculating the reward signal for the reinforcement learning algorithm based on the value provided by the happiness network. Unlike most other 3D animat simulators which have only tested flat worlds, *EcoTwin* simulated more complex and realistic terrain with mountainous terrain and valleys, trees, and bodies of water like lakes, in line with the plan of [Wilson, 1991].

[De Carlo et al., 2023] developed a 3D ecosystem of evolving and interacting robots using the *Revolve* simulation platform. The environment was a flat plane. The robots were made of modular bricks, which could rotate at the joint. The brain was not a neural network, but a special type of network called central pattern generator which outputs oscillatory signals. Both the brain and body could evolve. The body was encoded directly using a tree-based encoding, and the brain was encoded indirectly using the CPPN-NEAT (aka HyperNEAT). Rather than a pure on-line genetic algorithm which continuously ran, this system was partially generational-based or "oracle-based", such that at a certain time 75% of the population is culled and replaced with offspring. The offspring parents were determined from by keeping a list of which robots when near each other during the evaluation period. The fitness function was a measure of speed, or how fast the robots moved.

Still, despite the progress made so far, there are other promising frameworks to try in animat ecosystems in pursuit of richer and more lifelike simulations; for example, soft robots, voxel worlds, and other types of AI. In a past work [Hahm, 2022], I laid out the research plan to develop my own advanced animat ecosystem. First, I developed the cellular automaton voxel environment [Hahm, 2023], then evolved Hebbian networks to control articulated robots for locomotion [Hahm, 2024b]. After casually experimenting with morphological evolution and brain-body evolution for a period of a few months, and experiencing difficulties, I decided to forego morphological evolution entirely for this iteration of the project, especially when discovering some theoretical limitations that prevent performing brain-body evolution, as analyzed by [Cheney et al., 2016]. Next, I had to evolve the creatures' brains for food-seeking so they could self-sustain, so I evolved neural networks to control soft voxel robots for the food-seeking task using vision [Hahm, 2025]. Here, the project reaches its culmination, where the animats are made capable of reproducing and various other actions. This paper reports the results of evolving multiple robot types and cognitive algorithms in various environments for the two major animal tasks: eating food and reproducing.

3 Techniques

The experiments tested various techniques for each of the 3 animat categories: mind, body, and environment. Multiple techniques were also used for the evolutionary algorithm.

3.1 I. Mind

Firstly, in the category of ***mind***, this study tested three cognitive algorithms, plus a real-time neural learning algorithm. The three cognitive algorithms tested were:

1. sum-and-squash neural network
2. continuous-time neural network (CTNN)
3. non-axiomatic reasoning system (NARS)

The neural networks were tested both without learning (static weights), and also with real-time learning enabled (using a Hebbian learning abstraction called ABCD). NARS was tested without learning only, such that the system was only capable of using two logical inference rules: deduction and revision.

3.1.1 Sum-and-squash ANN

The traditional **sum-and-squash** method was selected because the most common type of neural network and also the simplest and most straightforward. In this method, at each discrete time step, each neuron's activation value is computed.

To compute the activation value, each neuron sums the activations of its inputs, weighted by their respective connection strengths. Then, the sum is put through an activation function which constrains its output to a certain range.

At the current timestep t, the new activation value y for a neuron with number of input connections N is calculated using:

$$y = \boldsymbol{\sigma}\left(\theta + \sum_{i=1}^{N} w_i y_i\right) \tag{1}$$

Where $\theta \in (-\infty, \infty)$ is the neuron's bias. For the i-th input neuron, y_i is its activation value computed at the previous timestep $(t-1)$, and $w_i \in (-\infty, \infty)$ is the weight on its connection. $\boldsymbol{\sigma}$ is the activation function.

In these experiments, the activation function used is sigmoid. It is a nonlinear function that constrains the output from $(0, 1)$:

$$\boldsymbol{\sigma}(x) = \frac{1}{1 + e^{-\alpha x}} \tag{2}$$

Where α is a parameter which modulates the sigmoid's slope (here $\alpha = 1$). The neuron's activation value is an abstraction of a biological neuron's firing rate; near 0 represents the neuron is not firing at all, near 1 represents the neuron is firing at its maximum possible rate.

The sum-and-squash ANN is a well-tested, popular, and powerful neural abstraction, and so it makes for a good baseline to compare with the other methods. A special property about the sum-and-squash ANN is that, for a network with multiple layers of sum-and-squash neurons are connected and at least one hidden layer, the network is a "universal function approximator" [Hornik et al., 1989]. This means sum-and-squash ANNs can approximate any arbitrary continuous mathematical function so that, given specific values at its input layer, the network will compute an approximation of the function's expected values at its output layer. Consequently, it is considered a powerful method for any problem that can be addressed with direct input-output mappings.

3.1.2 CTNN

Continuous-time neural network (CTNN, aka CTRNN) was selected because it has been used historically in evolutionary robotics (e.g., [Beer, 2006, Bongard et al., 2008]) and can theoretically better model the dynamics exhibited by biological neurons than regular sum-and-squash ANNs.

Rather than calculating each neuron's activation from scratch at each time step as in sum-and-squash ANN, in CTNN there is a concept of persistent internal neuronal voltage, that is carried over through multiple time steps. At each time step, we calculate the *change* of the voltage, instead of the voltage's value directly. That is to say, the neuron stores its voltage value internally, giving it a sort of internal "memory", and that value changes over time according to a specific differential equation (the CTNN equation). The voltage is then used to calculate the neuron's activation, also by putting it through a squashing activation function.

Using [Bongard, 2025], for a neuron with N incoming connections, the CTNN differential equation describing the rate of change $\frac{dV}{dt}$ for the neuron's voltage V is given by:

$$\tau \frac{dV}{dt} = -V_{t-1} + \sum_{i=1}^{N} w_i \boldsymbol{\sigma}(g_i(V_i + \theta_i)) \tag{3}$$

Where $\tau \in (0, \infty)$ is the neuron's time constant, a number which modulates how sensitive the neuron is to changes in voltage; a large time constant means the neuron voltage changes very slowly and gradually,

whereas a small time constant means the neuron voltage very quickly and drastically. V_{t-1} is the neuron's voltage computed at the previous timestep $(t-1)$. In the summation, for the i-th input neuron, V_i is its voltage computed at the previous timestep $(t-1)$, w_i is the weight on its connection, $g_i \in (0, \infty)$ is its gain (a value which amplifies the neuron's raw unsquashed output), and θ_i is its bias. $\boldsymbol{\sigma}$ is the activation function; in the CTNN experiments, sigmoid with range $[0, 1]$ was the activation function used, just as in the sum-and-squash experiments.

We can rearrange the equation to determine the formula to compute the rate of change directly:

$$\frac{dV}{dt} = \frac{-V_{t-1} + \sum_{i=1}^{N} w_i \boldsymbol{\sigma}(g_i(V_i + \theta_i))}{\tau} \tag{4}$$

Since these experiments took place in a discrete simulation, technically this formula is used to compute a discrete approximation of the rate of change, $\frac{\Delta V}{\Delta t}$, rather than the infinitesimal derivative $\frac{dV}{dt}$.

The rate of change is numerically integrated using the Euler method, such that at each time step t we directly compute the change in neuron voltage since the last time step $t-1$ using the discrete finite difference in time between timesteps Δt:

$$\Delta V = \frac{-V_{t-1} + \sum_{i=1}^{N} w_i \boldsymbol{\sigma}(g_i(V_i + \theta_i))}{\tau} \Delta t \tag{5}$$

then use the voltage change ΔV to compute the new voltage value V_t at time $t+1$:

$$V_t = V_{t-1} + \Delta V \tag{6}$$

Looking at the CTNN formula, notice that there is a summation in it that looks very similar to the summation in the regular sum-and-squash formula, *Equation 1*. The difference is instead of assuming the input neuron's activation value y_i was precomputed last timestep, in the summation we explicitly have the formula for computing the input neuron's activation value $\boldsymbol{\sigma}(g_i(V_i + \theta_i))$. Therefore, just to simplify things, at each timestep of the simulation, we can compute the activation value y to be used in the next timestep:

$$y = \boldsymbol{\sigma}(g(V + \theta)) \tag{7}$$

In this way, the CTNN equation can be represented in a simpler form more similar to the regular sum-and-squash equation:

$$\Delta V = \frac{-V_{t-1} + \sum_{i=1}^{N} w_i y_i}{\tau} \Delta t \tag{8}$$

where y_i is the activation value or firing rate of the i-th input neuron computed at the previous timestep $(t-1)$.

The properties of CTNN are such that it has the potential to represent realistic neural behavior: "*small networks of CTRNNs can reproduce qualitatively the full range of nerve cell phenomenology, including spiking, plateau potentials, bursting, etc. More importantly, CTRNNs are known to be universal approximators of smooth dynamics... Thus, at least in principle, the use of CTRNNs implies no restriction whatsoever on biological realism*" [Beer, 2006, p.4-5].

3.1.3 Hebbian ABCD

For the neural networks, a real-time local learning algorithm called Hebbian ABCD [Niv et al., 2001, p.255] was tested.

It is very straightforward. The weight on a connection from neuron i to neuron j is perturbed by Δw at each timestep according to the following equation:

$$\Delta w = r(Ay_i y_j + By_i + Cy_j + D) \tag{9}$$

where y_i is the activation value of neuron i, y_j is the activation value of neuron j, r is the evolved learning rate for the connection, A is the evolved correlation coefficient, B is the evolved presynaptic coefficient, C is the evolved postsynaptic coefficient, and D is the evolved learning bias. The evolved parameters r, A, B, C, D have no maximum/minimum limit for their values.

The algorithm was selected for a couple reasons. Firstly, to give the neural networks the ability to learn, rather than to be stuck with fixed weights; this gives the opportunity for the agents to better adapt to their environment. Secondly, because it is a real-time learning algorithm; this lets the agents learn and adapt on the fly, unlike many modern AI systems which undergo a pre-training phase only and stop learning after deployment; [Najarro and Risi, 2020] showed that quadruped robots can adapt their locomotion gaits when using this formulation. Thirdly, adding Hebbian learning makes the neural abstraction more realistic; Hebb-type learning is based on the natural biological neural learning process.

3.1.4 NARS

The general-purpose **non-axiomatic reasoning system** (NARS) [Wang, 1995, Wang, 2013b] was selected as an interesting alternative cognitive framework to neural networks, since evolutionary algorithms are not restricted to neural networks. There are 2 primary classes of AI [Minsky, 1991]: neural networks (connectionism) and symbolic systems. NARS is in the latter class, operating on symbolic statements using logical reasoning to derive new symbols and truth values.

The memory of NARS is organized using many instances of a data structure called *concept*. A concept is labeled by a symbol called a *term*, and the concept contains *knowledge* about its symbol. There are two types of knowledge: *judgment* and *goal*. Judgments are facts about the world, or more precisely, represent a small piece of evidence in favor of a fact. Goals determine the system's motivation, and similarly are considered according to the available evidence. The content of every judgment or goal is represented using a symbolic *statement*. The standard statement represents a hierarchical relation and contains 3 components:

$$<S \rightarrow P> \tag{10}$$

In this statement, S is the *subject* term, P is the *predicate* term, and $\rightarrow$ is the primitive relation between them called the *inheritance copula*. The brackets $<$ and $>$ simply denote the statement and contain its components. This statement means "S is a P". The subject S is the more specific form of a concept, whereas the predicate is the more general form. For example, the statement:

$$<raven \rightarrow bird> \tag{11}$$

represents "a raven is a bird". Putting a *punctuation* at the end of a statement turns it into a *sentence*. Judgments are sentences ending in ".", whereas goals are sentences ending in "!". Therefore, a judgment $<raven \rightarrow bird>.$ in the system's memory means the system *believes* that ravens are a type of bird, whereas a goal $<raven \rightarrow bird>!$ means the system *wants* ravens to be a type of bird.

Goals in NARS can be very specific or very general, and can be in agreement with each other or conflicting [Hahm et al., 2021]. Usually, goals are not used for declarative and encyclopedic knowledge, but in the context of specific events. For example, if we had a door that was open and that we wanted NARS to shut, we could enter the following goal into the NARS system:

$$<\{door\} \rightarrow [shut]>! \tag{12}$$

The "!" punctuation denotes that the sentence is a goal, the curly braces and denote that *door* is a specific instance of a door (not a general concept of door), and the square brackets [and] denotes that *shut* is a directly observable sensory property.

Each sentence has a dual-valued truth-value, (f, c). f is the *frequency* or "degree of positive evidence" and c is the confidence or "degree of total evidence". The truth-value of a sentence is continuous, in $[0, 1]$, which allows representing truth as a "matter of degree", according to available units of *evidence*, rather than discrete Boolean $true$ or $false$.

The specific NARS system evolved here is called *NARS-Unity*, a C# implementation of NARS ported from *NARS-Python v0.4* [Hahm, 2021], though other NARS systems could easily be evolved too, like the official implementation *OpenNARS* [Hammer et al., 2016] and the application-oriented *OpenNARS-for-Applications* (ONA) [Hammer, 2021].

Though neural networks have replaced symbolic systems for most tasks, NARS has shown success in experiments often delegated to neural networks including vision [Wang, 2018, Wang et al., 2019, Wang et al., 2022, Hahm, 2024a], natural language processing [Wang, 2013a, Kilic, 2015, Ireland, 2024], and speech

recognition [van der Sluis, 2023]. The benefit of the NARS system is that it guarantees logical cognition and, as a symbolic and reasoning system, is the type of AI that directly addresses the "unexplainabity crisis" plaguing AI [Hahm and Suereth, 2025]. Along that note, it is also possible to appeal to the emotions of NARS [Wang et al., 2016, Li et al., 2018, Li, 2021] and directly monitor its derived symbolic goals.

While NARS does have certain benefits over neural networks, it also has some of the same limitations and difficulties as previous symbolic and logic systems. Firstly, the system struggles to effectively process numeric (e.g., sensory and motor) data. Secondly, it is unknown how to scale the system up to highly complicated levels, with huge numbers of statements and symbols, especially while managing combinatorial explosion of symbols and keeping track of the evidence. For both of these problems, evolutionary search may provide a potential solution.

3.2 II. Body

Secondly, in the category of ***body***, this study tested three types of robots. These were:

1. wheeled robot
2. articulated robot
3. soft voxel robot (SVR)

For each robot type, a certain morphology was hand-designed, based on what made the robots relatively stable and realistic looking.

3.2.1 Universal robot components

Each robot type had its own unique set of sensors and motors. However, there was also a shared set or universal set of sensors and motors, which every robot had no matter its type. These sensors and motors managed the body, its metabolism, and vision.

3.2.1.1 General and Metabolism All robots had a body and components related to metabolism. There were **6 sensors** related to metabolism and general body functioning that all robots had:

1. Internal energy
2. Internal health
3. Mouth
4. Pain
5. Internal voxels
6. Sinewave

The *internal energy* sensor informed the animat about its current energy storage status, a scalar value E. An animat is born with $E = E_{\text{birth}}$ energy, where E_{birth} is the energy cost to generate one offspring, here $E_{\text{birth}} = \frac{E_{\text{food}}}{2} = 50$. This energy is passed down from the parent, thus enforcing a conservation of energy. The internal energy sensor's activation value is set to E/E_{max}. An animat can store up to $E_{\text{max}} = 500$ energy at maximum, which is the equivalent of 5 fully-grown food cubes. Internal energy gradually depletes by 0.1 energy per timestep. The animat dies if internal energy reaches zero, $E \leq 0$.

The *internal health* sensor informed the animat about its current health amount H. An animat was born with $H = H_{\text{max}}$ health, where $H_{\text{max}} = 100$. An animat's health amount could only be depleted when another animat attacks it. However, like energy, health is restored when food is consumed, up to H_{max}. An animat died if its internal health amount reached zero, $H \leq 0$. It did not drop food when it died, however it did drop all of its held voxels (in a vertical column).

The *mouth* sensor informed the animat about whether or not it is successfully eating food. Its activation was simply the amount of food eaten in timestep, which in this case is a value in $[0, 2]$.

The *pain* sensor informed the animat about whether its being attacked. The activation is set to the amount of health damage which was taken during the timestep. This is a value in $[0, N]$, if N animats are attacking during the timestep.

The *internal voxels* sensor informed the animat about how many voxels ξ it has picked up and is currently holding. An animat could hold up to $\xi_{\max}$ voxels, where $\xi_{\max} = 5$. The sensor's activation value was set to $\frac{\xi}{\xi_{\max}}$.

The *sinewave* sensor provided a time-varying sinewave signal, in case it might help with oscillatory movements for locomotion. The activation for the sinewave sensor was set to $sin(\nu * 2 * \pi * t)$ where t is the current time in seconds and ν modulates the sinewave frequency, here $\nu = 3$.

There were **6 motors** related to metabolism and general body functioning that all robots had:

1. Eat
2. Fight
3. Clone (asexual reproduce)
4. Mate (sexual reproduce)
5. Pickup voxel
6. Place voxel

For all of these actions, there was a threshold: the action only executed if the motor activation was 0.5 or above. If it was below the 0.5 threshold, then the action was not executed. The exception was the *clone* action, which had a smaller threshold of 0.05, to encourage the animats to express the action.

The *eat* motor action was used by the animat to consume the green food blocks in the world. It can only be used if a food block is in action range. A green food block holds some amount of food, up to E_{food}, here $E_{\text{food}} = 100$. If a food block was in action range, the activation of the motor neuron in $[0, 1]$ determined how much food the animat consumed and converted to energy, assuming the activation exceeded the threshold. So, if 1 unit of food was eaten, 1 energy was gained by the animat. The food/energy was removed from the food block, and if reduced below zero the food block was removed and respawned elsewhere. A parameter called the "eat rate", here set to 2, was a multiplier for the motor activation; so if an animat fully activated its neuron to 1, the agent would eat 2 food in that timestep (or for activation 0.5, the agent would eat 1 food, etc.).

The *fight* motor action was used by the animat to attack other animats. It can only be used if another animat is in action range. If another animat was in action range, the activation of the motor neuron in $[0, 1]$ determined how much health the other animat lost, assuming the activation exceeded the threshold. In *PolyWorld*, animats who died in a fight would drop food [Yaeger et al., 1994, p.4], though this technique was not used in these experiments since in preliminary tests the world become too full of food.

The *clone* motor action was used by the animat to perform asexual reproduction. It can be used at any time. This action had two requirements to execute: the motor expression threshold, (0.05), and also an energy requirement to have a minimum of $2 * E_{\text{birth}}$ energy. If the action succeeded, the amount E_{birth} of energy was subtracted from the animat and imparted to the offspring as its starting reserve of energy. This motor action is particularly useful because it almost perfectly preserves the parent's genome, with the exception of a few small mutations. Thus, animats which are very good at surviving and cloning themselves will tend to produce offspring very similar to themselves, which are also good at surviving and cloning themselves.

The *mate* motor action was used by the animat to perform sexual reproduction. It could only be used if another animat was in action range. This action had four requirements to execute: the animat's motor activation must exceed the expression threshold (0.5), the animat must exceed the minimum energy requirement of E_{birth} energy, the *other* animat must exceed the expression threshold (0.5) for the "mate" motor action, and the *other* animat must exceed the minimum energy requirement of E_{birth} energy. Despite its difficult requirements, this motor action is useful because it allows drastic recombinations of genomes, which helps explore the search space. It is also a remarkable feat, for two animats to achieve these requirements and reproduce, so ideally their offspring would be viable (though recombination might make this idea a struggle).

The *pickup voxel* motor action was used only in the 3D interactive voxel world, and was used to pickup certain types of voxels out of the world and store them ("in the body", virtually). It can only be executed if an interactive voxel is in action range. The animat must also have enough internal space to hold more voxels (i.e., $\xi \leq \xi_{\max}$). When a voxel was picked up, the internal voxel store went up by 1.

The *place voxel* motor action was used only in the 3D interactive voxel world, and was used to place voxels in front of the animat. It can be executed as long as there is no voxel in front of the animat. The animat must also be holding at least one voxel (i.e., $\xi > 0$). When a voxel was placed, the internal voxel store went down by 1.

3.2.1.2 Vision Vision played a particularly important role in the simulation, since it acted as both sensor and motor. It not only allowed the animats to detect food and eachother from a distance, but it was also the mechanism through which interaction motor actions were implemented. The animat had to point its face directly at the object of interest, and be within a certain distance to it, to be considered within "action range". It was not enough to merely be within a certain distance; the vision sensor had to be oriented towards the object to interact with it. This requirement was meant to produce more interesting behaviors than a distance-only approach.

The *action range* was 2 meters, and determined the maximum distance at which certain motor actions could be executed. An animat had to be in action range to *eat* food, *mate* with another animat, *fight* another animat, and to *pickup* a voxel.

In these experiments, vision was not implemented like RGB vision, since I guess it might over-complicate this initial problem of achieving robots which eat food and reproduce. RGB vision is saved for future experiments. Instead, vision was implemented using raycasts which explicitly detected objects of interest in the world.

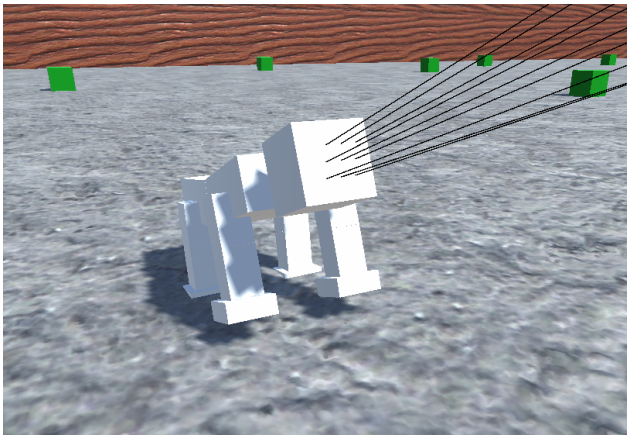


Figure 1: The raycasts for vision, emanating from an animat robot's face. The animat uses the vision raycasts to detect objects and direct its behavior.

The dimensions of the eye were $\text{dim}_{\text{eye}} \times \text{dim}_{\text{eye}}$, here $\text{dim}_{\text{eye}} = 3$ so the eye dimensions were 3×3, for a total of 9 raycasts (changed to $5 \times 5 = 25$ raycasts in the voxel world). The raycast origins were arranged in a square, approximately 0.1 meters apart, with the raycast directions of the outer ring angled approximately 2 degrees outward from the center (see *Figure 1*).

Each ray travelled a distance rayDist, up to a maximum distance of $\text{rayDist}_{\max}$ meters, here $\text{rayDist}_{\max} = 40$. The "closeness" of an object can therefore be calculated by $\text{closeness} = \text{rayDist}_{\max} - \text{rayDist}$, and it is a value in $[0, \text{rayDist}_{\max}]$. Therefore, because the neural activation should be in $[0, 1]$ and represent how close the object is, the sensory vision activation y for a detected object was set to:

$$y = \frac{\text{closeness}}{\text{rayDist}_{\max}} = \frac{\text{rayDist}_{\max} - \text{rayDist}}{\text{rayDist}_{\max}} \tag{13}$$

3 types of objects could be detected (4 total in the voxel world):

1. Food
2. Animat
3. Obstacle
4. Interactive voxel (*voxel world only*)

Therefore, with 9 vision raycasts and 3 sensory neurons per raycast, there were $9 \times 3 = 27$ (in the voxel world, $25 \times 4 = 100$) vision sensory neurons.

3.2.2 Wheeled robot

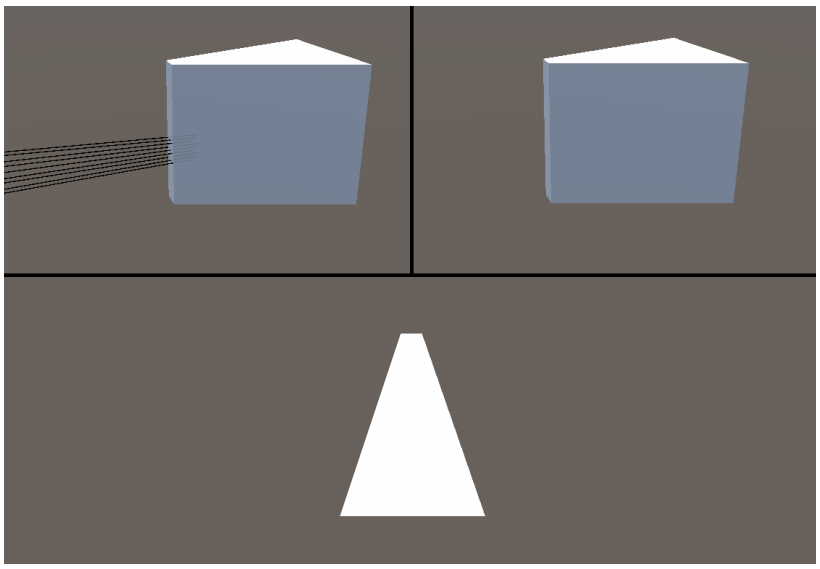


Figure 2: Wheeled robot. *Top left*: side view with vision rays. *Top right*: side view. *Bottom*: Aerial view.

The **wheeled robot** is one of the simplest forms of mobile robot. It is a rigid metallic body on a set of wheels (see *Figure 2*). In terms of its movement, this type of robot is similar to the R2-D2 robot from the movie *Star Wars*. It was selected for experimentation because this type of robot has been used often historically in AI, and is used even to this day for many AI experiments. It is simple to setup and use, and is reliable because it is stable. There is little to no risk of a wheeled robot falling over, unless it is particularly taller than it is wide.

Especially of note, this type of robot has been used in animat ecosystem experiments. The particular wheeled robot developed here is inspired by the animats of *PolyWorld*, including very similar behaviors [Yaeger et al., 1994, p.10] and shape [Yaeger et al., 1994, p.24].

The **sensory modalities** which were unique to the wheeled robot only were:

- whole-body touch sensor

The *whole-body touch sensor* fired 1 when any part of the robot is touching something, and 0 otherwise.

The **motor actions** which were unique to the wheeled robot only were:

- drive forward
- turn left
- turn right
- jump

The robot could *drive forward* to move around in the environment. This action moves the robot forward by a certain distance, directly proportional to the motor neuron's activation. Concretely, given the robot's current position $\vec{P_t}$ as a 3D vector at time t, and the motor activation $y_{\text{move}} \in [0, 1]$, the robot's next position $\vec{P_{t+1}}$ was computed for time $(t + 1)$ using:

$$\vec{P_{t+1}} = \vec{P_t} + s_{\text{move}} * \hat{\mathbf{k}} * y_{\text{move}} \tag{14}$$

where $\hat{\mathbf{k}}$ is the normalized 3D forward vector (relative to the robot), which is the +z-direction here, and s_{move} is a scalar determining the "maximum wheeled robot movement speed", a value which can be configured by the robot designer (here $s_{\text{move}} = 0.25$).

The robot can *turn left* and *turn right*. Each action is represented by a separate motor neuron, though the final result determining turn direction is calculated by subtracting the two motor activations. Concretely, given the robot's current rotation R_t as a 4D quaternion at time t, and the motor activations $y_{\text{left}} \in [0, 1]$ for turning left and $y_{\text{right}} \in [0, 1]$ for turning right, the robot's next rotation R_{t+1} was computed for time $(t + 1)$ using:

$$R_{t+1} = R_t * \text{EulerToQuaternion}\left(abs(y_{\text{right}} - y_{\text{left}}) * \hat{\mathbf{j}} * s_{\text{rotate}}\right) \tag{15}$$

where $\hat{\mathbf{j}}$ is the normalized 3D up vector (relative to the robot), which is the +y-direction here, and s_{rotate} is a scalar determining the "maximum wheeled robot rotation speed", a value which can be configured by the robot designer (here $s_{\text{rotate}} = 5.0$). The function "EulerToQuaternion" converts a 3D vector $\vec{v}$ representing an Euler rotation into a quaternion, by assuming the rotation occurs by rotating the identity first by $\vec{v_z}$ degrees on the z-axis, then by $\vec{v_x}$ degrees on the x-axis, and finally by $\vec{v_y}$ degrees on the y-axis.

With forward movement and rotation, the robot's movement is like that of a *PolyWorld* bot. If the robot had these actions only, its movement would be restricted to 2-dimensions. This is fine for flat or slightly sloped worlds, but will not suffice for mountainous 3-dimensional worlds which require climbing, such as voxel worlds. To remedy this, the wheeled robot was also equipped with a *jump* or *climb* action, which allowed the robot to move upwards, but only if it is touching an object.

The jump action is implemented by adding a scalar value $s_{\text{jump}} * y_{\text{jump}}$ to the y-component of the robot's position. $y_{\text{jump}} \in [0, 1]$ is the motor activation, and s_{jump} is a scalar determining the "maximum wheeled robot jump speed" (here $s_{\text{jump}} = 0.2$). To implement jumping, we modify *Equation 14*, resulting in the final movement equation:

$$\vec{P_{t+1}} = \vec{P_t} + s_{\text{move}} * \hat{\mathbf{k}} * y_{\text{move}} + s_{\text{jump}} * \hat{\mathbf{j}} * y_{\text{jump}} \tag{16}$$

3.2.3 Articulated robot

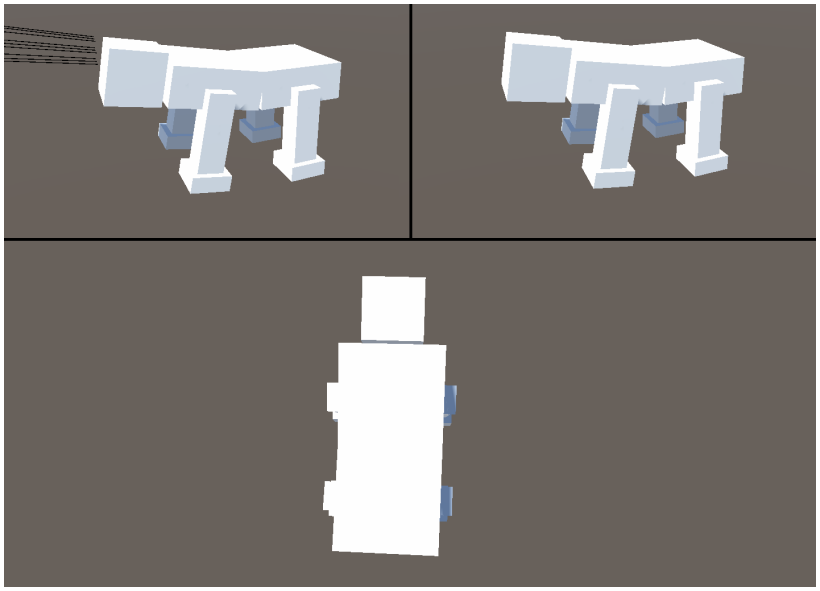


Figure 3: Articulated robot. *Top left*: side view with vision rays. *Top right*: side view. *Bottom*: Aerial view.

The **articulated robot** is the traditional "complex" robot. It is made from rigid metallic girders for body segments, connected by rotating joints (see *Figure 3*). This type of robot is how robots are usually depicted in popular culture, especially biped humanoids, like the *Terminator* or C-3PO from *Star Wars*.

The robots move by rotating their joints to maneuver their limbs. Unlike the wheeled robot, it is not necessarily stable when standing up, but has a tendency to fall over. This type of robot was selected for experimentation because it is the quintessential embodiment of what comes to mind when one thinks about a robot, and it is a frequently used type of robot in AI and robotics research.

The robot used in the experiments was designed to have 4 jointed legs, flat moveable feet, and a moveable head/eye.

The **sensory modalities** which were unique to the articulated robot only were:

- 6-face touch sensor
- segment rotation sensor

Since the articulated robot's body segments were cuboid, they had 6 sides which could be used for touch sensation. Consequently, for each segment in the robot, there were 6 touch sensor neurons, 1 corresponding to each side of the body segment. If that side of the body segment was touching something, the sensory neuron would fire 1, otherwise 0.

There were also 4 sensory neurons in each body segment to detect its rotation. The 4 sensory neurons corresponded to the 4 components of the quaternion (values in range $[-1, 1]$) representing the body segment's rotation in world space.

The **motor actions** which were unique to the articulated robot only were:

- rotate joint drive (3 DoF, x-, y-, and z-axis)

Each joint on the robot had 3 degrees of freedom: rotation on the x-axis, y-axis, and z-axis. The drive limits of the x-axis were $[-10°, 10°]$. The drive limits of the y-axis were $[-25°, 25°]$. The drive limits of the z-axis were $[-10°, 10°]$. A motor activation of 0 set the drive target rotation to the minimum value, whereas a motor activation of 1 set the drive target rotation to the maximum value.

The joints were simulated using Unity's built-in physics engine and "articulation body" component in "target mode", which attempts to bring the joint towards a certain target rotation (determined here by the motor neuron). The torque T on a joint is computed using a spring-damper equation [Unity, 2024] similar to:

$$T = s(\theta_{t+1} - \theta_t) - d\omega \tag{17}$$

where θ_t is the joint drive's current angle in radians, θ_{t+1} is the desired joint angle in radians, ω is the joint's current angular velocity in radians per second, s is the joint stiffness in $\mathrm{N\,m\,rad^{-1}}$ (here $s = 1000$), and d is the joint motion damping in $\mathrm{N\,m\,s\,rad^{-1}}$ (here $d = 107.75$).

Since this morphology had 15 segments, the articulated robot had an additional $15 \times 10 = 150$ sensory neurons and $15 \times 3 = 45$ motor neurons.

3.2.4 Soft voxel robot

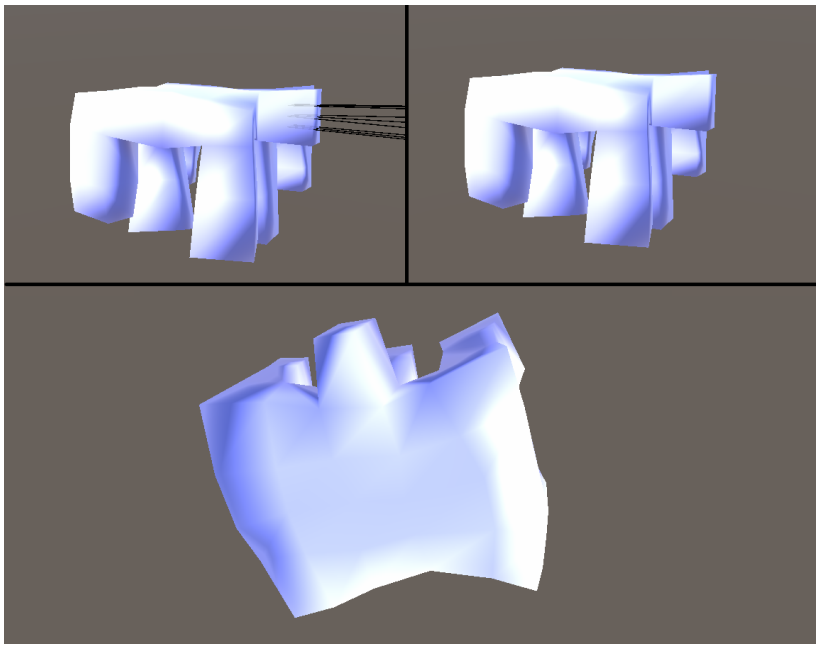


Figure 4: Soft voxel robot. *Top left*: side view with vision rays. *Top right*: side view. *Bottom*: Aerial view.

The **soft voxel robot** [Hiller and Lipson, 2011] is a newer type of complex robot, made from multiple connected soft and deformable cubes (see 4). This framework allows for robots with new and varied material properties, including homogeneous and heterogeneous tightly-connected regions of hard and soft materials, and new robot capabilities, such as squeezing through tight crevices, deforming around surfaces, etc. These types of robots are more similar to squishy biological organisms than the usual depictions of robots.

The soft voxel robot was simulated by using the *Voxelyze* engine [Hiller and Lipson, 2011, Hiller and Lipson, 2014] in *Unity* via a custom-created API and DLL. The *Voxelyze* simulator simulates the soft voxel physics and provides the mesh information which was drawn in *Unity*. *Voxelyze* simulates soft voxel robots using a mass-spring lattice system, the full details and equations for which can be found in [Hiller and Lipson, 2014].

The parameters set for the simulation are recorded in *Table 1*. The robot used in the experiments was designed to have 6 legs and a moveable head/eye.

This type of robot was selected for experimentation because it is a next-generation robotic framework, which can provide some of the same capabilities as articulated robots plus new additional capabilities like soft deformation. It is also especially amenable to morphological evolution, such as using CPPN-NEAT [Cheney et al., 2014], which makes it interesting for future work in this domain.

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	0.25×10^6 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	0.7
Gravity multiplier	1.0

Table 1: *Voxelyze* parameter values.

This kind of robot moves by inflating and deflating its voxels. This robots is unstable and can fall over, but in general it is much more stable than the articulated robot because it has a wider, squishier base.

The **sensory modalities** which were unique to the soft voxel robot only were:

- voxel ground touch sensor
- voxel rotation sensor

For each voxel in the robot, there were 3 sensory neurons: 1 ground touch neuron, and 2 voxel rotation sensors. The ground touch sensor returned 1 when the voxel intersected the ground (according to the *Voxelyze* engine), and 0 otherwise. For the rotation, custom sensors were developed that measured the pitch and roll angles of the voxel. The rotation sensor activation value was the pitch/roll angle in degrees (usually from a value from $[-90, 90]$) divided by 90 degrees, to get a value in range $[-1, 1]$.

The **motor actions** which were unique to the soft voxel robot only were:

- expand voxel (3 DoF, x-,y-, and z-axis)

For each voxel in the robot, there were 3 motor neurons. Each neuron corresponded to 1 axis of the voxel (x, y, or z). A motor activation of 0 shrank the axis to its minimum length, whereas a motor activation of 1 expanded the axis to its maximum length.

3.3 III. Environment

Thirdly, in the category of ***environment***, this study tested two types of environments. These were:

1. static flat plane
2. interactive voxel world

Each environment had dimensions of 256×256 meters, and was surrounded on all 4 sides by tall walls. Animats who happened to fall through the world, such as by climbing over the walls, or accidentally glitching through the terrain, were killed (determined by checking if the animat's y-position fell below the -5 meters line.)

3.3.1 Static flat plane

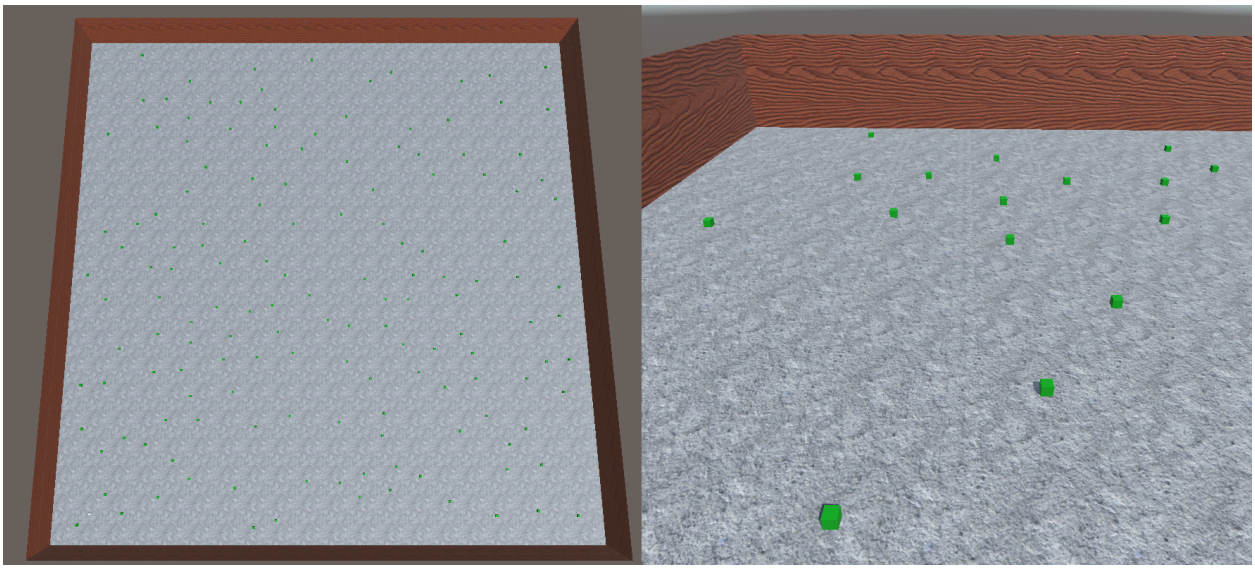


Figure 5: The flat plane environment.

The **static flat plane** environment is the simplest possible type of environment, similar to the environment of *PolyWorld* [Yaeger et al., 1994, p.11]. It was a completely flat terrain in the shape of a square (see *Figure 5*). In this environment, the terrain never changed throughout evolution or the animats' lives, though food randomly spawned within it.

There is not much to say about this world, except to mention its simplicity. It was chosen for experimentation as a way to eliminate unnecessary challenges, to allow the best chance for the desired animat behaviors to evolve. The idea is to find a method that works for the flat world, then gradually increase the difficulty and complexity of the environment, such as by moving over to the interactive voxel world.

3.3.2 Interactive voxel world

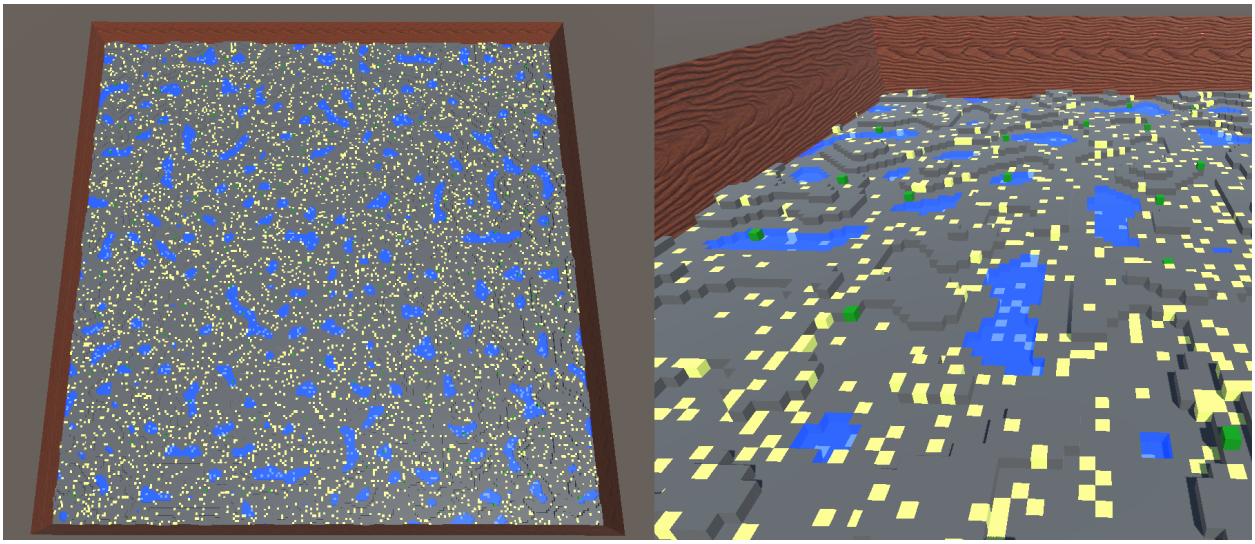


Figure 6: The interactive voxel world environment. The yellow blocks represent sand, the gray blocks represent stone, the blue blocks represent water.

The **interactive voxel world** environment was a world made of granular particles, which can move around autonomously and be interacted with by the animats. The world was discretized into a regular cubic grid, so that the environment is volumetric and composed of uniformly-sized cubes called "voxels" (see *Figure 6*). This voxel world was custom developed [Hahm, 2023], using a cellular automaton framework.

The environment contained different types of voxels representing different real-world elements like *sand*, *stone*, and *water*. Sand and stone acted like solids; they fall down due to gravity, then stay where they are. Sand is particularly physically unstable, so unlike stone, it can topple down slopes, resulting in piles of sand whenever many sand particles are grouped together. Water on the other hand, acted like a liquid. It falls down due to gravity, but never stays in place; instead, it flows horizontally, bouncing around its container and flowing to fill gaps in the terrain.

The agents were given the ability to interact with the world by picking up and placing down voxels. Thus, the terrain landscape had the capacity to continuously change throughout generations and even within a single organism's lifetime. They could only pick up and put down sand voxels. Stone voxels were like immovable obstacles. Water voxels also acted like obstacles, but they flowed around the map, making them a dynamic obstacle that could push animats around and interrupt their locomotion path.

3.4 Evolutionary Algorithm

The *evolutionary algorithm* (EA) adapted the Mind structures over multiple generations to achieve higher performance on the animat tasks of survival and reproduction.

Rather than testing variations on the evolutionary algorithm in different experiments, instead various methods of evolutionary algorithms were combined and used all at once for every experiment. The techniques used included fitness-based search, novelty search, hall-of-fame, roulette-wheel, and tournament selection.

3.4.1 Evolutionary Search Strategy

For the ***evolutionary search*** strategy, the two major techniques were used:

1. fitness-based search
2. novelty search

The minor techniques used include hall-of-fame, roulette-wheel selection, and tournament selection. These techniques complement each other well, one fixing the deficiencies of another.

For both cases, a data structure called *animat table* was developed. It acted as a set of animats which the EA could pull from for reproduction, to generate new animats in the world as needed.

3.4.1.1 Fitness-based search The first search strategy was traditional objective **fitness-based search**. In this search method, the animats were more likely to be selected for reproduction if they had a higher *fitness score* $\mathcal{F}_{obj}$. The magnitude of the fitness score was directly determined by how well the animat performed on achieving their objectives in the environment.

"Fitness" was a combination of three objectives: 1.) moving around, 2.) eating food, and 3.) mating. The multi-objective fitness function was calculated using "scalarisation" [Eiben and Smith, 2015, p.196], which simply means to take a weighted sum or product of the fitness scores for each individual objective.

Concretely, the fitness score $\mathcal{F}_{obj}$ was computed upon an animat's death, as:

$$\mathcal{F}_{obj} = B * L * \frac{E_{\text{eaten}}}{E_{\text{food}}} * (1 + \mathcal{R}) \tag{18}$$

where $B \in [0, 1]$ is the displacement of an animat from its birthplace divided by 20 meters (to encourage walking), with a max value of one. $L \in [0, 1]$ is the ratio of time that an animat spent looking at food (to encourage seeking food), with a value that got higher the closer the animat was to the food. E_{eaten} is the amount of food eaten or energy consumed during the animat's lifetime (to encourage food eating). Finally, $\mathcal{R}$ is the total number of times that an animat reproduced in its lifetime, asexually and sexually.

Fitness-based search is useful because it is intuitive, and it simply works, at least for very simple problems. The major issue with fitness-based search is that it easily gets stuck in local optima; it does not reward the

suboptimal "stepping stones" to the globally optimal structure [Lehman and Stanley, 2011, p.11]. The search will help to find some structures which immediately increase fitness, but it will not necessarily increase fitness over the long term.

3.4.1.2 Novelty-based search To preemptively counter the problem of getting stuck in local optima, a second modern approach called **novelty search** [Lehman and Stanley, 2011] was also employed. In this search method, the animats were more likely to be selected for reproduction if they had a higher *novelty score* $\mathcal{F}_{nov}$. The magnitude of the novelty score was directly determined by how novel the animats acted in the environment compared to previous animats, by using a data structure that remembers past behaviors called the *behavior archive*.

To calculate novelty is much more complicated than calculating fitness. While fitness can merely be computed at the end of the animat's life, computing novelty requires recording the animat's behavior throughout its entire life. Every 1.5 seconds of the animat's life, a *behavior snapshot* was sampled. A snapshot method similar to the one from [Lehman and Stanley, 2011, p.26] was used, recording the animat's current *X and Z offset from its birthplace*. Also, for the purposes of this animat simulation, another metric recorded was the amount of food eaten since the last snapshot.

These snapshots are recorded in a list, or vector, called the animat's *behavior characterization*. This characterization was considered as describing the animat's behavior during its life; in this way, computing the distance between two behavior characterization vectors provides a quantitative measure of how differently two animats behaved. An animat which behaved very differently from animats in the past is considered highly "novel".

To find the distance d between two behavior characterization vectors, first they were aligned. If one vector was shorter than the other, and the longer vector is size N, because the animat did not live as long, it was padded at the end with zero for all values. Once they were the same length, the element-wise distance was computed. For a given snapshot comparison, the displacement from the animat's birthplace on the $x-$axis was B_{x1} for animat 1 and was B_{x2} for animat 2. The displacement from the animat's birthplace on the $z-$axis was B_{z1} for animat 1 and was B_{z2} for animat 2. The quantity of food/energy consumed since the last snapshot was E_1 and E_2 for animat 1 and animat 2 respectively. The distance d_i between two behavioral snapshots was computed as:

$$d_i = abs(E_2 - E_1) + \sqrt{(B_{x2} - B_{x1})^2 + (B_{y2} - B_{y1})^2} \tag{19}$$

This equation effectively says two animats are very different if their displacements were different, *or* if their amount of food-eating was different.

Then, the total distance d between the vectors was computed by summing the element-wise distances:

$$d = \sum_{i=0}^{N-1} d_i \tag{20}$$

To calculate an animat's novelty score $\mathcal{F}_{nov}$, first the algorithm must find the distance between the animat's behavior characterization vector and every characterization vector in the historical behavior archive. The archive contained the behavior characterization vectors of past animats. Since the number of animats is massive, we cannot save every single animat in the archive, without expecting major computational resource strain (both space and time). Therefore, to save on space, an animat must meet some criteria to be allowed to enter the archive. In traditional novelty search [Lehman and Stanley, 2011, p.32], the criteria was a novelty threshold, such that $\mathcal{F}_{nov}$ must exceed a certain value (e.g., $\mathcal{F}_{nov} > 1$). The criterion employed here is simply randomness; the animat has a certain percent chance to be added to the archive (here the criterion is 3%). This decision was informed by the experimental results of [Gomes et al., 2015, p.947], which found that a randomness archive criterion was more effective than threshold-based criteria.

Once the animat's behavior is compared to every behavior in the archive, the result is a list or vector of distances. This list is sorted in ascending order, to find the $k-$nearest neighbors (here $k = 15$, informed by [Gomes et al., 2015, p.947]). The average of these k values is the animat's novelty score $\mathcal{F}_{nov}$.

Novelty search is useful because it forces the search to explore a variety of behaviors, whereas traditional fitness-based search keeps trying to branch off the most fit structure, which may result in the same behaviors

over and over. In this way, novelty search helps to escape local optima. On the flip side, the benefit of novelty search is also its limitation: novel behaviors are not necessarily highly fit behaviors.

NOTE: since this is a computationally expensive search algorithm, initial experiments showed that it overwhelmed the CPU and lagged the simulation, especially while other computationally-expensive algorithms like the brains, robot bodies, and voxel world were running on the CPU. To relieve this pressure, parts of novelty search were implemented to run on the GPU using compute shaders. The first compute shader computed the distance between an animat and the archive, resulting in a list of distances. The second compute shader sorted this list using a bitonic sort. Then, the smallest k values were fetched to the CPU.

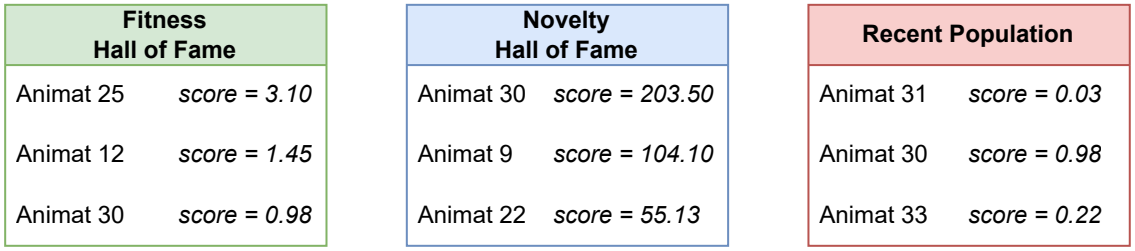


Figure 7: A mockup representation of the three animat tables used by the evolutionary algorithm to track the best performing animats and to generate new animats via reproduction.

3.4.1.3 Animat table A data structure called *animat table* was developed to use in the evolutionary search process. There were three animat tables:

1. fitness hall of fame
2. novelty hall of fame
3. fitness continuous

An animat table holds up to N animats (here $N = 100$), and is either *Hall of Fame* (sorted by animat score) or *Recent Population* (unsorted). See *Figure 7* for a representation of the three tables.

A *hall of fame* animat table stored all animats in sorted order, determined by their score (fitness or novelty). When the algorithm tries to add an animat to the table, its score must exceed at least the worst score in the table. If so, then it is added to the table. If not, then it is not added. If the number of animats in the table exceeds N, then the entry with the worst score is removed.

There were two Hall of Fame tables, one that stored the most fit animats of all time (measured by $\mathcal{F}_{obj}$), and one that stored the most novel animats of all time (measured by $\mathcal{F}_{nov}$). Of course, an animat's novelty score changes over time as more behaviors are added to the archive; a behavior that used to be very novel early on in the simulation, might not be very novel at all any more later in the simulation, especially if that animat produced many offspring that acted similar to their parent. Therefore, the novelty score $\mathcal{F}_{nov}$ was recomputed every time an animat was added to the historical behavior archive, to keep the scores up to date.

The *Recent Population* or *continuous* animat table stored all animats in first-in-first-out (FIFO) order. This idea is similar to traditional evolutionary algorithms which perform reproduction/mutation in explicit waves or generations, selecting from the most recent generation for reproduction. The Hall-of-Fame method may become stuck in local optima, since the pool of animats might never update if they never improve, causing the search to be stuck branching off from the same solutions over and over, with no hope of escaping. The generations of solutions never advance in that case. However, the continuous method might help to prevent the search process from becoming stuck in local optima, since it forces the structure to evolve further and test modifications. Even if the performance of the population does not improve immediately, or even becomes worse in the short-term, with this method, later and later generations will continue to be tested. By allowing long-term adaptation, this method provides an opportunity for a more globally optimal structure to

emerge. In fact, mutations which advance the structure but do not immediately impact performance, called "neutral mutations", may be key to successfully evolving high-complexity structures [Lehman and Stanley, 2011, p.8].

When it came time for the evolutionary algorithm to generate a new animat, it had a 1-in-20 chance to spawn a completely fresh animat, a 1-in-9 chance to spawn an animat from a table using asexual reproduction (one parent), and a 1-in-10 chance to spawn two new animats from a table using sexual reproduction (two parents). If an animat was to be selected from a table from reproduction, a table was randomly selected. This was the case for both parents in sexual reproduction, meaning that two parents could come from two different animat tables, or possibly the same table. Once a table was chosen, an animat was selected from it using either *roulette-wheel* or *tournament selection*, according to 50/50 chance. These are standard evolutionary algorithm selection methods [Brownlee, 2025].

In *roulette-wheel*, or fitness-proportionate, selection, an animat was probabilistically selected from the table, according to a probability proportional to the animat's score. Therefore, an animat with twice the score of another has twice the chance to be selected. This method is good because it tends to choose high-performing animats, but for that reason it is also bad: the same high-performing animats got selected over and over, not giving much of a chance to low-performing animats which might be good "stepping stones" to globally optimal solutions. Therefore, a second selection method called *tournament selection* was implemented, which first selects a size k subset of animats (here $k = 7$) using uniform random chance, and ultimately returns the highest-scoring animat from that subset. In this way, all animats in the table have an equal chance to be selected into the subset, but a high-performing one is still selected.

3.4.2 Genetic Encodings

Only the mind was evolved. Genetic encodings were selected for evolving neural and symbolic cognitive algorithms.

3.4.2.1 NEAT for neuroevolution For evolving the neural networks, the neuroevolution of augmenting topologies (NEAT) genetic encoding was selected. This method is well-established as a reliable method for evolving complicated neural networks.

This method is a direct encoding, meaning the genome directly represents every individual neuron and connection of the phenotype. This is both a weakness and a strength compared to other encodings. It is a weakness because, unlike indirect and generative encodings, it cannot compactly or efficiently represent modularity and regularities in the network. If a module must be replicated 100 times, then NEAT is forced to evolve it independently 100 separate times. However, the direct encoding is also a strength, because it allows fine-tuning the network and producing precise structures. Indirect and generative encodings are often overly regular and repetitious, making it difficult to evolve fine-tuned networks with irregularities and asymmetries in the structure.

Unlike most neural encodings, NEAT not only evolves the weights on the connections, but also evolves the topology of the network. The "topology" is another word for the network architecture, including the neurons, connections, and their positions. This means we are able to start with an empty or mostly empty neural network and simply let NEAT add new connections and neurons as needed, without requiring us to manually pre-specify a decent neural architecture.

The NEAT genome contains two lists, one for neurons and one for connections. Each neuron in the list has a global neuron ID to identify the gene, and each connection has a global connection ID. These IDs are "global" in the sense that they are shared by all animats; two animats with gene ID "5" share an evolutionary history, they could not both separately evolve genes with the same ID.

Mutation worked as follows:

There was a 35% chance to add a new connection. There was a 9% chance to add a new node.

There was an 80% chance to modify the weights of all connections. If so, each connection weight had a 90% chance of being perturbed, and a 10% chance of being replaced by a random value in range $[-3, 3]$. Perturbation values were pulled from a Gaussian (normal) distribution, with mean = 0 and standard deviation = 0.2. This was accomplished using the Box-Muller transformation method. The bias of a neuron is sometimes treated as a connection in neuroevolution experiments. Here, though it was not a connection but an attribute of a neuron, the biases was mutated whenever the connection weights were mutated. The bias

had a 90% chance of being perturbed by a small random value from the Gaussian distribution, and a 10% chance of being perturbed by a large random value from range $[-3, 3]$.

In the cases where Hebbian ABCD parameters were mutated, or CTNN gain and time constant, they were perturbed in the same way as bias. There was a 80% chance to mutate all 5 Hebbian ABCD parameters on all connections, a separate 80% chance to mutate the time constant on all nodes, and a separate 80% chance to mutate the gain on all nodes.

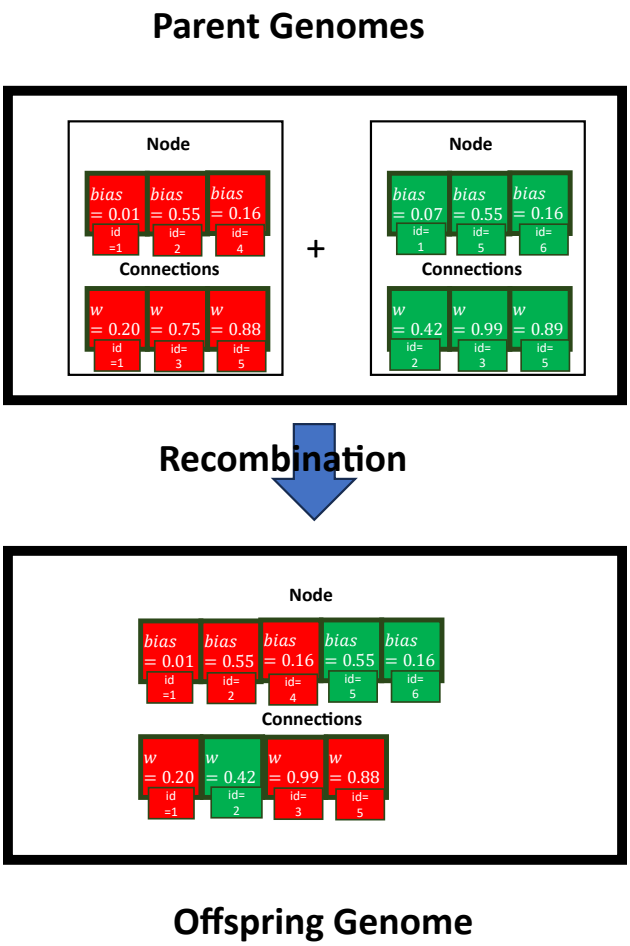


Figure 8: A visualization of NEAT genome recombination.

Reproduction and genome recombination were implemented as in standard NEAT [Stanley and Miikkulainen, 2002, p.109] (see *Figure 8*), as follows:

An algorithm iterated over each gene in the genome. For a given gene, If both parent genomes contained that gene, in the sense that they had the same ID, all the offspring values for that gene were copied over from a random parent. If the gene was only contained in one parent, its values were simply passed on to the offspring genome. In this way, recombination can greatly augment the topology of the network. When a new node is added to the network, it is added in place of an existing connection, which is then disabled. During crossover, a disabled connection had a 25% chance to become re-enabled. Two offsprings were generated this way.

3.4.2.2 NARS encoding For evolving NARS, a custom genetic encoding was developed [Hahm and Wang, 2025]. This encoding allowed evolving both the meta level and the object level of NARS.

The meta-level parameters influence the system's control mechanism. Though the fundamental processes of NARS should be the same across all systems, such as the logic, there is some room for variation that may allow certain system configurations to perform better than others. Concretely, the meta-level parameters

selected here are called the *personality parameters* of NARS, and influence aspects of NARS related to its personality.

The first personality parameter selected was the *evidential horizon* k; this very important parameter determines the weight of a single unit of evidence in NAL. A higher k value means a single unit of evidence is weaker, and more evidence must be accumulated to achieve high confidence, whereas lower k value means a single unit of evidence is stronger, and less evidence must be accumulated to achieve high confidence. The range[1] for $k \in [1, \text{inf})$.

The second personality parameter selected was *cautiousness* T, aka the *desire threshold*. This parameter determines the threshold of desirability that a goal must exceed in order for NARS to pursue it. If the goal initially selected for processing by NARS does not exceed the threshold, it is not processed further. If it is processed further, then subgoals are derived from the goal in order to achieve it. A high T value means that NARS is more cautious, and will not pursue goals unless it is very confident about them. A low T value means NARS is more impulsive, and will pursue nearly any goal even if it is uncertain about them. The range for $T \in (0, 1.0)$.

The object-level parameters determine the system's knowledge. Though object-level knowledge can and should be learned by the system during its lifetime, some knowledge can also be evolved, just like in neural networks the weights (which encode the network's knowledge) can be evolved or learned. The genome contains knowledge, judgments, in the form of *sensorimotor contingencies*:

$$<(S, \Uparrow M) \Rightarrow P>.$$

These judgments provide "instincts" to the system about how its action impact the world, and under which sensory stimuli it should apply those actions. S is the sensory precondition, under which the motor operation $\Uparrow M$ should be executed, in order to achieve sensory postcondition P. For example, the sensorimotor contigency:

$$<<\{food\} \rightarrow [near]>, \Uparrow eat \Rightarrow <\{ENERGY\} \rightarrow [full]>)>. \langle 1.0, 0.9 \rangle \tag{21}$$

tells the agent how to get food. It can be roughly translated to English: "`when food is nearby and you eat, then your energy will become full`". The truth-value is the pair of numbers in angle brackets, $\langle 1.0, 0.9 \rangle$, representing frequency of 1.0 and a confidence of 0.9.

The NARS genome was composed of 3 components: the instinctual belief set β, the instinctual goal set Γ, and the personality parameter vector Π. β contained the animat's beliefs regarding sensorimotor contingencies. Γ contained the goals which were input to the NARS system throughout its lifetime. Π contained the numeric values for the personality parameters k and T. Only β and Π could evolve, while Γ was the same for all agents, containing 2 goals: eat food, and reproduce.

A concrete example of a NARS genome which resulted in NARS performing the desired animat tasks is as follows:

Example NARS Animat Genome

β:
$J_1 = <<\{food\} \rightarrow [far]>, \Uparrow move \Rightarrow <\{food\} \rightarrow [near]>>. \langle 0.99, 0.99 \rangle$
$J_2 = <<\{food\} \rightarrow [near]>, \Uparrow eat \Rightarrow <\{ENERGY\} \rightarrow [full]>)>. \langle 1.0, 0.99 \rangle$
$J_3 = <<\{ENERGY\} \rightarrow [full]>, \Uparrow clone \Rightarrow <\{SELF\} \rightarrow [mated]>>. \langle 0.95, 0.99 \rangle$
$J_4 = <<\{food\} \rightarrow [unseen]>, \Uparrow turn \Rightarrow <\{food\} \rightarrow [far]>>. \langle 1.0, 0.99 \rangle$
$J_5 = <<\{food\} \rightarrow [unseen]>, \Uparrow move \Rightarrow <\{food\} \rightarrow [far]>>. \langle 1.0, 0.99 \rangle$

Γ:
$G_1 = <\{ENERGY\} \rightarrow [full]>! \langle 1.0, 0.99 \rangle$

[1]Note that technically, k is not allowed to go below 1 [Wang, 2013b, p.52] or else it can cause some trouble with the reasoning process, though in these experiments this constraint was not accounted for, and k did drop below 1 in some agents.

$G_2 = <\{SELF\} \rightarrow [mated]>! \ \langle 0.9, 0.99 \rangle$

Π:
$k = 1$
$T = 0.51$

The beliefs in β were evolved using statements from the "sensorymotor" set. To generate a sensorimotor contingency, two random sensory statements plus one random motor statement were selected.

The **sensory set** $\mathbb{S}$ contained all possible sensory statements:

$\mathbb{S}$	$<\{food\} \rightarrow [near]>$
	$<\{food\} \rightarrow [far]>$
	$<\{food\} \rightarrow [unseen]>$
	$<\{animat\} \rightarrow [near]>$
	$<\{animat\} \rightarrow [far]>$
	$<\{animat\} \rightarrow [unseen]>$
	$<\{ENERGY\} \rightarrow [full]>$
	$<\{SELF\} \rightarrow [mated]>$

Table 2: The set of possible sensory events that NARS was able to experience.

The sensations were all related to vision and metabolism. The subject of the sensory statement is called an "instance", and the predicate is called a "property" of the instance [Wang, 2013b, p.72-75].

For the sensory instances, there were two typess of entities that NARS could detect: $\{animat\}$, which was another animat in the world, and $\{food\}$, which was a food block. $\{ENERGY\}$ represented the agent's internal energy store.

Then, for the properties, they corresponded to states of the sensor instances.

- The vision sensor had two possible properties.
  - $[near]$ - object is detected very close in the visual field
  - $[far]$ - object is detected far away in the visual field
- There were also two properties for metabolism and physiology.
  - $[full]$ - used to indicate the $\{ENERGY\}$ stores are sufficiently high to reproduce
  - $[mated]$ - used to indicate that the NARS agent has successfully reproduced.

The **motor set** $\mathbb{M}$ contained all possible motor statements for controlling the wheeled robot:

$\mathbb{M}$	$\Uparrow move$
	$\Uparrow turn$
	$\Uparrow eat$
	$\Uparrow clone$
	$\Uparrow mate$

Table 3: The set of motor operations that NARS was able to use.

- The $\Uparrow move$ operation moved the animat forward by a certain distance.
- The $\Uparrow turn$ operation rotated the animat by a certain angle clockwise.
- The $\Uparrow eat$ operation transferred energy from a food block to the animat's internal energy store, if a food block was visually detected in action range.

- The $\Uparrow$ *clone* operation triggered asexual reproduction and generated one offspring, if the animat had enough energy.
- The $\Uparrow$ *mate* operation triggered sexual reproduction (crossover) and generated two offspring, if the animat had enough energy, and if another animat was visually detected in action range, had enough energy, and was also triggering its $\Uparrow$ *mate* operation.

Mutation worked as follows:

First, there was a 35% chance to mutate the content of the belief set β. If so, 1 of 3 mutations occurred: add new random belief, remove random belief, and modify random belief. For add new random belief, a new sensorimotor contingency was generated by randomly selecting 2 sensory statements from $\mathbb{S}$ and 1 motor statement from $\mathbb{M}$, then adding it to β. For remove random belief, a random belief from β was deleted. For modify random belief, a random belief was selected from β, one of its 3 components was randomly selected, and replaced with a random statement from the relevant set $\mathbb{S}$ or $\mathbb{M}$.

Next, there was an 80% chance to modify the truth-values of all beliefs in β. If so, there was a 90% chance to perturb the frequency by a value randomly selected from range $[-0.1, 0.1]$, and a 10% chance to replace the frequency with a value randomly selected from range $[0.5, 1.0]$. The constraint on frequency if $0.5 \leq f \leq 1.0$ (f should not go below 0.5 or else the statement becomes "false" rather than "true").

Finally, for the personality parameters in Π, each value was perturbed by a value randomly selected from range $[-0.1, 0.1]$.

Reproduction and genome recombination worked as follows:

First, the genomes of the two parents were aligned. Then, uniform crossover was performed. There was a 50/50 chance to determine which parent contributed to which child. A 50% chance that parent 1's gene went to offspring 1 and parent 2 to offspring 2, and a 50% chance that parent 1's gene went to offspring 2 and parent 2 to offspring 2. If the given parent's genome was too short, no gene was added. This crossover method was performed for both β and Π.

4 Experiments

The experiments tested the various state-of-the-art animat methods. The results are presented and reviewed here.

The timestep mentioned in the text, and for everything related to the brains, was 0.04 seconds per step. The physics simulators ran in timesteps of 0.02 seconds per step. The cellular automaton ran with a timestep of 0.25 seconds per step.

4.1 Experimental Setup

Firstly, a "control" or "base" animat configuration was defined.

Control configuration:

- **Mind:** sum-and-squash ANN
- **Body:** wheeled robot
- **Environment:** static flat plane

This was the base configuration from which the experiments deviated.

Experiments were run in each of the 3 categories. A given experiment varied the base configuration for the relevant category, while keeping the variables for all the other categories the same. For example, one experiment in the Mind category tested CTNN, wheeled robot, and static flat world; the configuration remained the same as in the base configuration, except for the Mind category which as changed to CTNN. This approach allows us to examine the effects of each method independently and better compare them.

The experiment scenario was as follows. Animats were placed in a contained environment. The environment contained 150 food blocks scattered around the arena, that, starting from containing 0 food, grew at a rate of 0.02 food per timestep to contain up to 100 maximum food. When a food was consumed, a new food was spawned in a random location. In early experiments, there was an issue caused by animats spawning

right next to food, or vice versa, and gaining a high fitness score even by doing nothing but eating the food in front of them. To prevent this, when spawning food and animats, they were spawned at a far distance away (8 meters) from other food/animats, or the closest position that could be found in a reasonable time. This forced to animats to evolve locomotion abilities first, and then subsequently to eat food.

The animats were evolved using a steady-state evolutionary algorithm, such that the minimum population quantity was guaranteed to be at least 50 at any given time. Animats had energy which slowly drained as time went on, but could be replenished by eating food. Animats also had a maximum lifespan, determined at birth from a random range of $[85, 95]$ seconds; when an animat's time alive exceeded this lifespan, they died.

When an animat died, such as by running out of energy or from old age, their fitness and novelty were scored. Then, they were added to the Continuous Animat Table, and if necessary, the Hall-of-Fame Animat Tables. If the animat's death resulted in the population dropping below the minimum value of 50, a new animat was generated according to the algorithm in *Section 3.4.1.3*.

Each experiment was run for 3 trials. The results from the 3 trials were averaged to get the final results. Each trial was run for 2 hours, which is 7200 seconds. Datapoints were recorded every 15 seconds, resulting in 480 datapoints per experimental trial.

4.2 Experiments in *Mind*

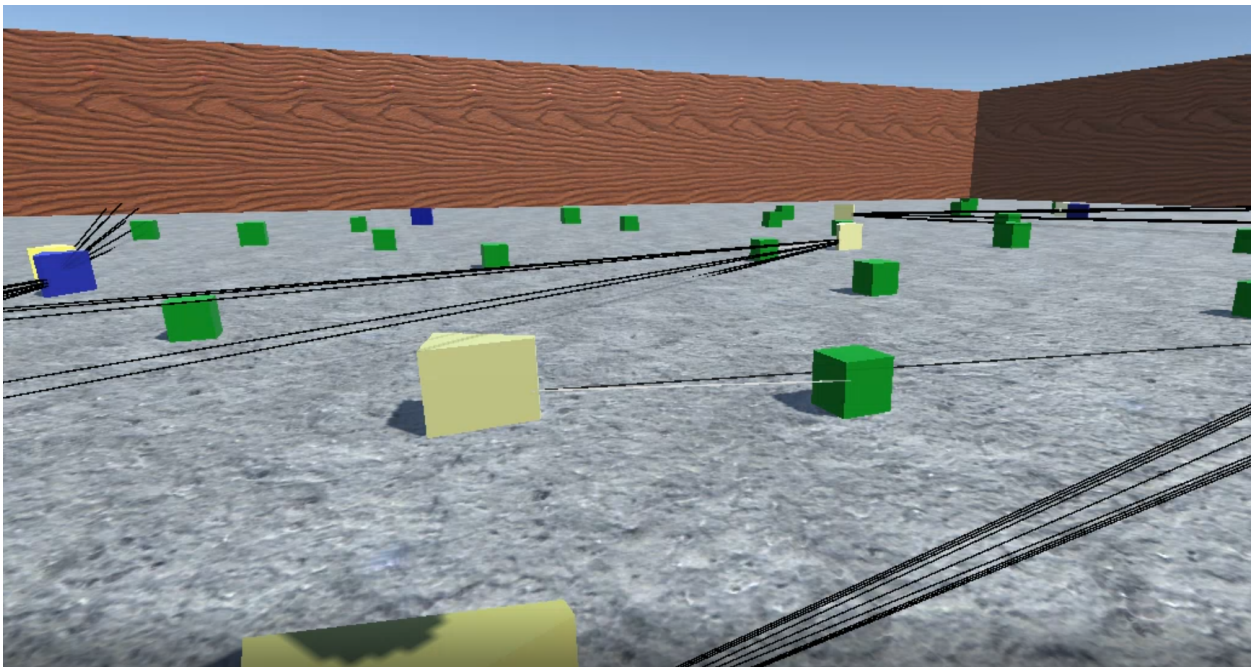


Figure 9: Wheeled robot animats in the flat plane world. The main pictured animat, the yellow one in the center, has spotted a food and is moving towards it. These robots are controlled by static-weighted Sum and Squash networks.

For the mind, 6 different configurations were tested. The body was the wheeled robot. The environment was the static flat plane.

The 6 different configurations tested for mind were: Sum and Squash (no learning), Sum and Squash (with Hebb ABCD), CTNN (no learning), CTNN (with Hebb ABCD), NARS, and Random. The last configuration, Random, had no cognitive processing at all, but simply performed random motor actions. It is included to test if the problem could simply be trivially solved by inputting random actions, rather than requiring deliberate behaviors that can improve over time.

First, let us examine the most important graph, *Figure 10*, which shows the percentage of animats alive in the world that were "born" through autonomous reproduction (as opposed to animats which were "created"

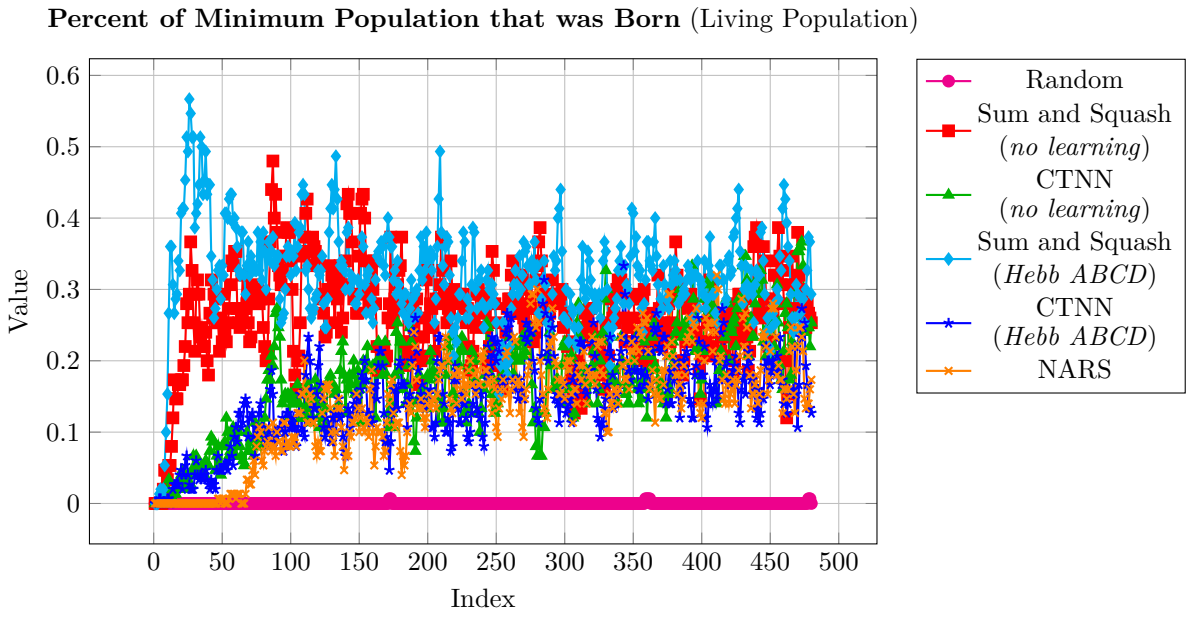


Figure 10: Ratio of birthed animats to total population in the *Mind* experiments.

by the evolutionary algorithm). This shows how successful the animat population as whole ultimately was, and their actual physical impact on the world.

From visually analyzing *Figure 10*, we can note a few points of interest.

Firstly, we see that the regular Sum and Squash methods performed the best, at one point achieving more than half of the population born autonomously. We can also see that the Random method did not accomplish the task at all. Though no method far outperformed all the others, the sum and squash methods were consistently on top throughout the duration of the simulation. This is an unexpected result, since CTNN is considered a more powerful neural framework, and more useful for evolutionary robotics. Even if the CTNN took longer to evolve, I would have expected it to eventually outperform sum and squash, though this was not the case. The Sum and Squash performed approximately 2x-3x better than the CTNN methods and NARS.

Secondly, we see that the Hebb ABCD method did appear to improve the Sum and Squash method, though had little to no impact on the CTNN method. The Sum and Squash Hebb method achieved the highest percentage overall, with 57% of the population born autonomously at index 27 (i.e., about 7 minutes in), and in general the Sum and Squash Hebb method stayed +5% to +10% higher than the non-Hebb method for the entirety of the simulation. Also of note, the Sum and Squash methods peaked early, by about index 25, then plateaued for the rest of the simulation, whereas the other methods started with low performance but continuously and gradually improved until the end of the simulation. Especially, CTNN without Hebb was beginning to approach the levels of the Sum and Squash methods by the very end, around index 480.

Thirdly, NARS was able to compete with the neural methods, and its performance stayed roughly in line with the CTNN methods throughout the simulation. Of note, NARS started off with near-zero performance at the beginning, far below the other methods, but quickly caught up with the CTNN methods after generation 50. This is likely because the actions of NARS are very deliberate, requiring specific reasons to execute motor actions, which requires specific beliefs and symbols, whereas the neural networks act more sporadically, requiring only causal activations, which bounce around the network via connections that do not discriminate symbolically. So in the beginning, the NARS agents were likely still building up their symbolic knowledge base about the proper sequence of how to move, eat, and mate, whereas the neural networks "mash the controller" so to speak and, using their changing quantitative sensory inputs, move quickly around the map, executing operations in parallel, which makes them more likely to run into food and reproduce by chance. This is illustrated by *Figure 11*; notice how in the NARS world, many of the NARS agents are

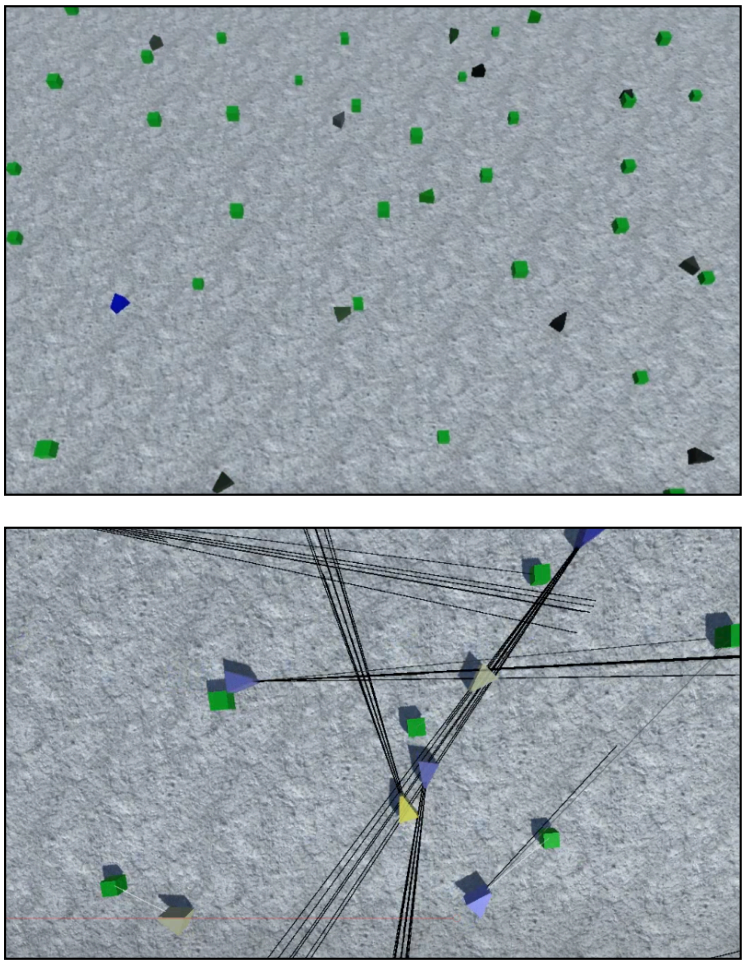


Figure 11: Side-by-side screenshots of the NARS animats (top) and the Sum and Squash no learning animats (bottom).

black, indicating they are not executing any action, whereas in the neural network world, all of the agents are all sorts of varied colors, because they are rapidly executing various motor actions.

Figure 12 shows the objective fitness scores $\mathcal{F}_{obj}$ of the animats in the Fitness Hall of Fame. From these results, we see a similar trend of the Sum and Squash methods outperforming the other methods, with Sum and Squash Hebb ABCD on top having a score of 42. Also in this graph, we can see the same trend of the Sum and Squash methods peaking very high early in the simulation then plateauing, with the CTNN methods catching up. By the end of the simulation, the CTNN methods achieved nearly the same maximum value as the Sum and Squash methods. In particular, towards the very end of the simulation, the CTNN method without learning was able to find a most fit animat that matched the Sum and Squash Hebb's most fit animat. The most fit CTNN animat had $\mathcal{F}_{obj} = 43.50$, which is almost equivalent to the most fit Sum and Squash Hebb animat's $\mathcal{F}_{obj} = 42.76$. When we examine the average scores, we can see that Sum and Squash maintained a significant lead throughout the whole simulation, though CTNN without learning approached similar performance towards the end. Here we can see the NARS performed worse than the neural methods, on both the maximum and average, achieving approximately less than half the score of the neural methods. The maximum NARS animat fitness was $\mathcal{F}_{obj} = 17.55$.

Figure 13 shows the amount of food eaten E_{eaten} by the animats in the Fitness Hall of Fame. Just like in the previous graphs, we can see that the Sum and Squash methods outperformed the rest, with the best animat being Sum and Squash Hebb eating 538 food. Some of the animats via the Random method were

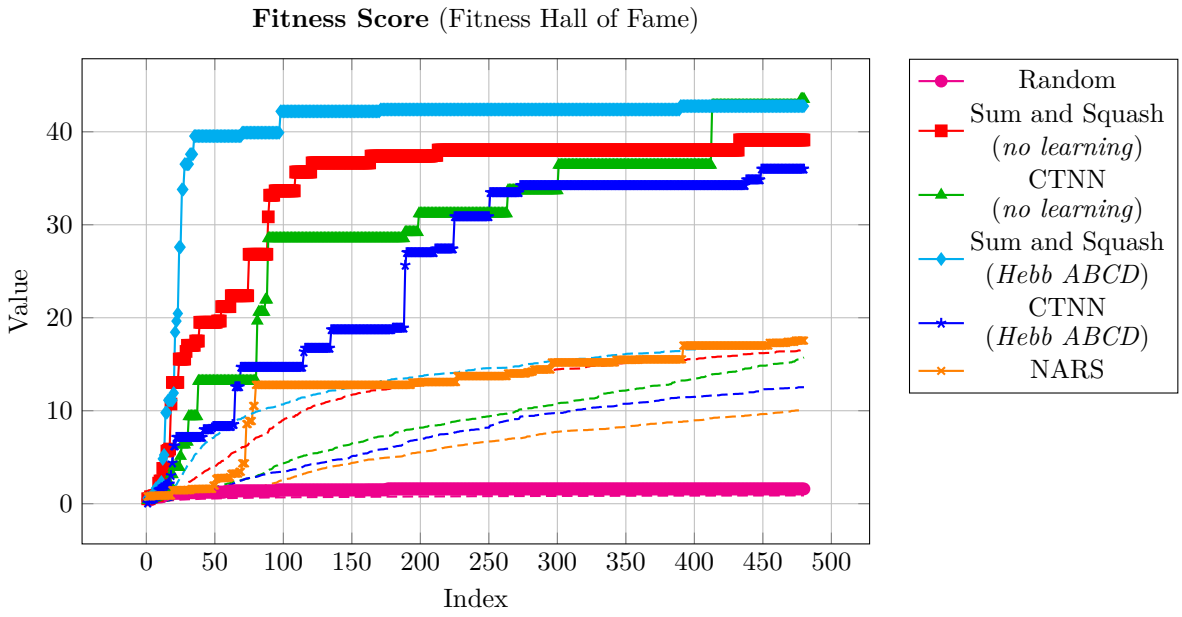


Figure 12: The fitness scores in the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum score achieved, the dashed line is the mean score of the table ($n = 100$).

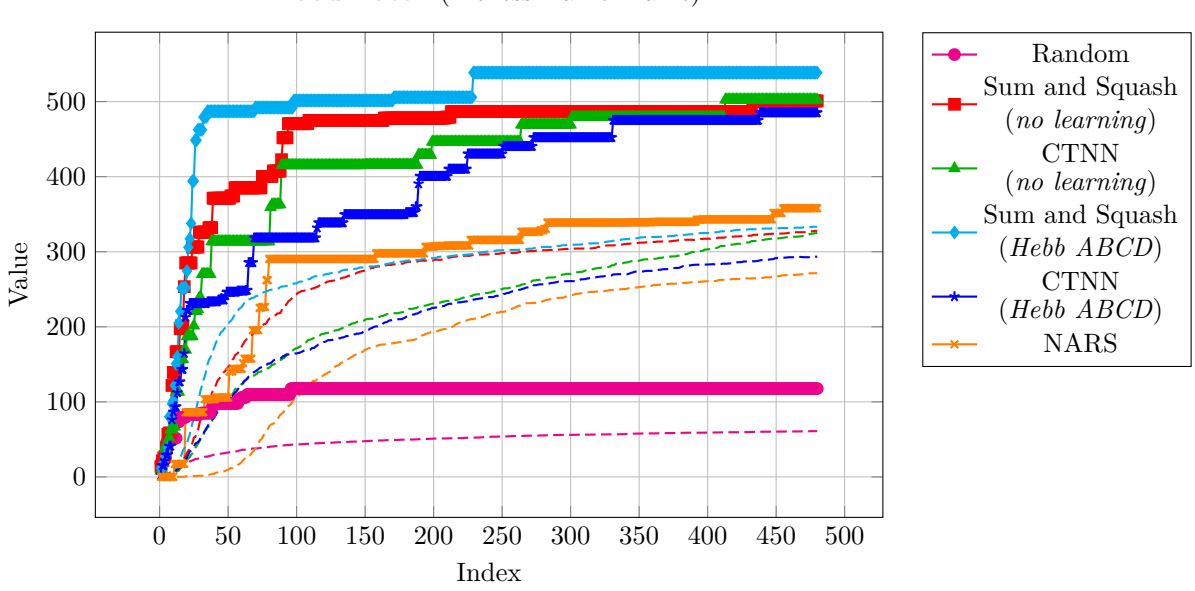


Figure 13: The food eaten scores in the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum food eaten, the dashed line is the mean food eaten of the table ($n = 100$).

apparently able to eat 100 food (i.e., 1 entire food block). Still, all cognitive methods greatly outperformed the Random method, for both the maximum food eaten and on average, and evolved to become better at eating food throughout the course of the simulation.

There appears to be some limit around 500 food, or 5 food blocks, because all of the methods appeared to asymptotically approach that amount. It could be because of the limited lifespan of the animat; perhaps they cannot eat more than 500 food in that limited time. But, since it only takes an animat only 2 seconds to eat an entire food block, and the animats can survive for up to 90 seconds, the fact that this limit appears is

curious. Theoretically, they should be able to eat many more blocks, potentially 20 or more. It might be that, over the course of evolution, the environment became too competitive, with many high-performing animats in the same environment competing for the limited food resources. Perhaps no one animat could survive for too long without eventually being starved out. This could explain the plateauing of birth performance observed in *Figure 10*.

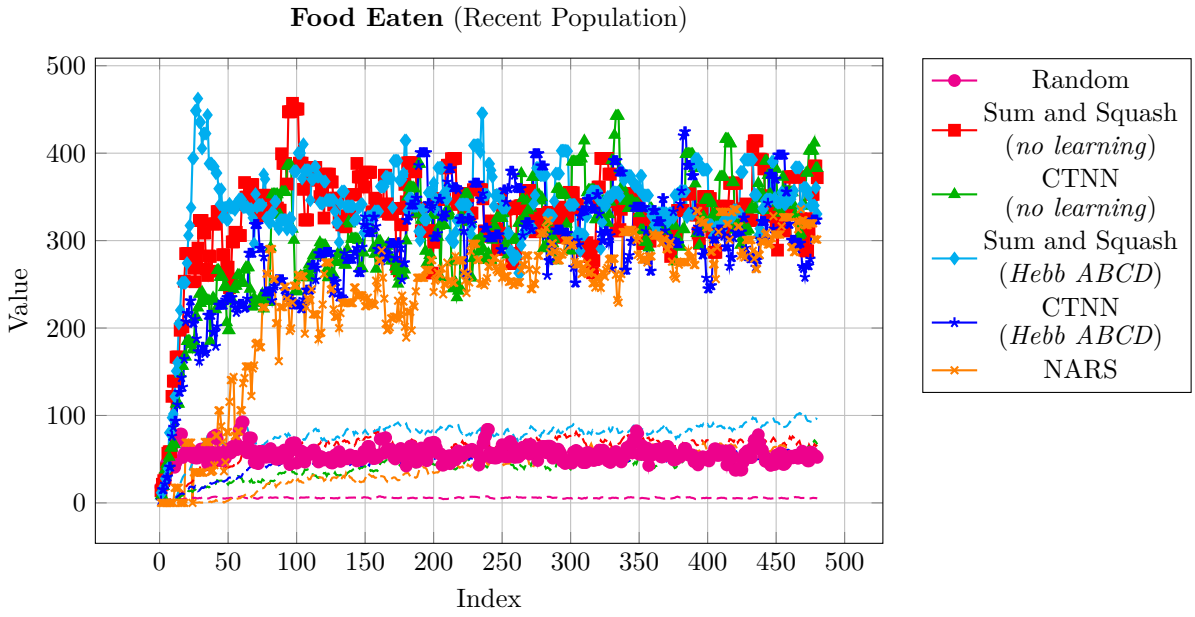


Figure 14: The food eaten in the Recent Population animat table, in the *Mind* experiments. The bold line is the maximum food eaten, the dashed line is the mean food eaten of the table ($n = 100$).

We can analyze the possible validity of this "overcompetition" hypothesis by examining the trend of the Recent Population table in *14*. From this graph, we can see that there is always at least one ravenous animat in the population, probably more; the maximum food eaten achieved by animats in the recent population mirrors the maximum food eaten achieved in the overall simulation. Thus, it is reasonable to think that at any given moment, there are multiple skilled animats competing for the limited food resources, and possibly outcompeting one another. Though, the average food eaten by the recent population is quite low, never breaching more than 1 food. This suggests that there are many low-skilled animats being tested in the population, or that majority of the population is dominated by a few skilled animats at any given moment.

To test if overcompetition is causing this cap or limit, and to remedy it, the environment should be made larger to accommodate all the animats. This woud provide ample resources for the animats to eat as they please, and a space to migrate to in order to escape from massive competition. Though, some competition is surely a useful evolutionary pressure towards better performance, so not all competition should be alleviated.

The consumption of food is directly related to the number of offspring which could be created. *Figure 15* shows the number of times that animats in the Fitness Hall of Fame asexually reproduced, and *Figure 16* shows the number of times they sexually reproduced. As in the previous graph, in terms of times reproduced, Sum and Squash outperformed the other methods, though CTNN caught up with Sum and Squash by the end of the simulation.

What is particularly surprising to note is the fact that sexual reproduction happened to such a large extent. I expected the animats to asexually reproduce the most, and to rarely sexually reproduce, because the animats could reproduce asexually at any moment they choose, as soon as they met the energy requirements. Therefore, it would be most efficient to immediately reproduce after eating, then move on to the next food. So once the animats are full of energy, in order to reproduce sexually, the animats have to inhibit asexual reproduction, and search around until they find a mate. The mate must do the exact same thing, *also* meeting the energy requirements, inhibiting asexual reproduction for a time period, and must also be expressing the sexual reproduction behavior at the same time as the other animat when the animats meet. This is a most

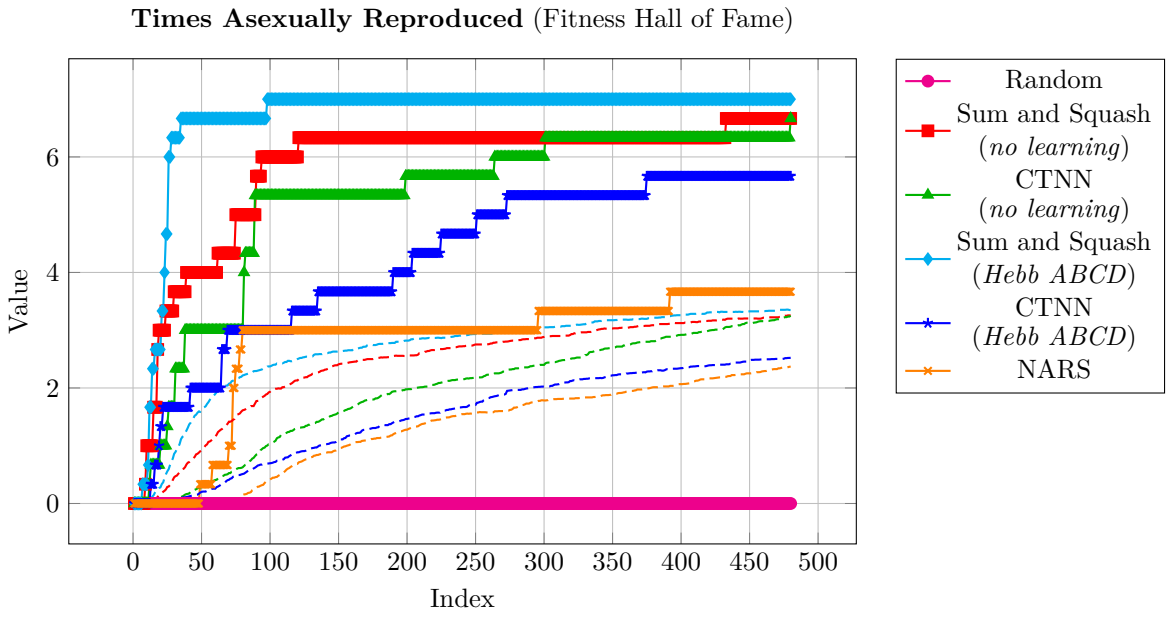


Figure 15: The times asexually reproduced scores in the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum times asexually reproduced, the dashed line is the mean times asexually reproduced of the table ($n = 100$).

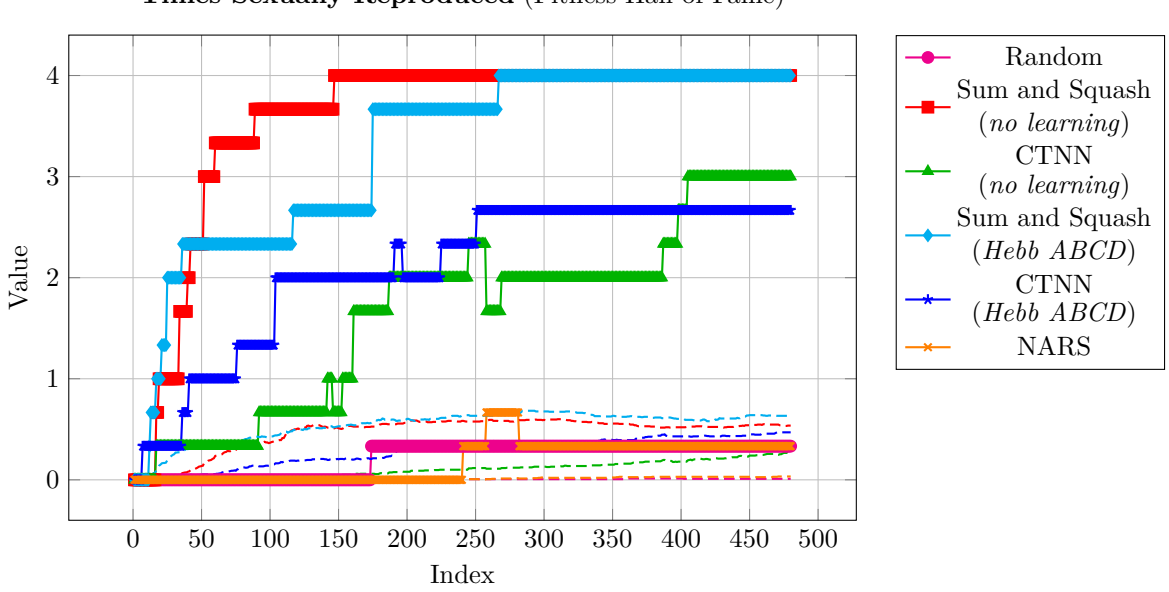


Figure 16: The times sexually reproduced the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum times sexually reproduced, the dashed line is the mean times sexually reproduced of the table ($n = 100$).

complicated sequence of events, and it seems on average the animats did tend to asexually reproduce more frequently; yet, certain animats where able to sexually reproduce 4 times in their life, which is not far below the high score of 7 times asexually reproduced.

Figure 17 shows the *reproduction chain*, which is like the generation number of the animat, except that it only increments when the animat was generated from an *autonomous reproduction*, aka a "birth." An animat

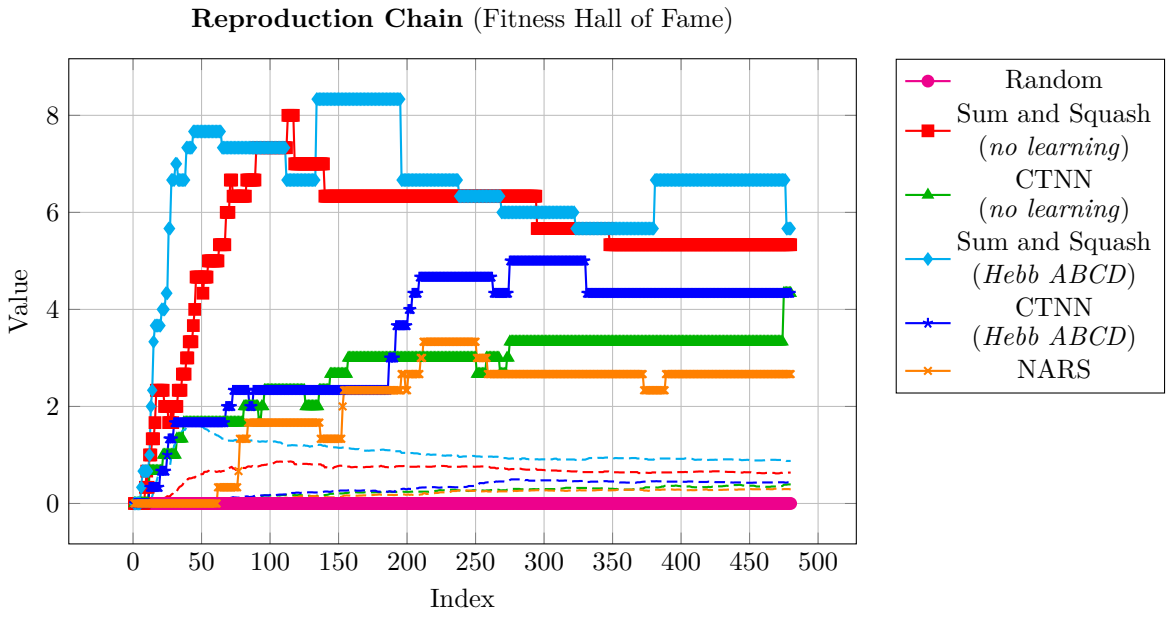


Figure 17: The reproduction chain scores in the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum reproduction chain score, the dashed line is the mean reproduction chain score of the table ($n = 100$).

that was generated by the evolutionary algorithm, or "created", has a reproduction chain value of zero. If it autonomously reproduces sexually or asexually, the offspring gets a reproduction chain value of one; if that offspring also autonomously reproduces, *its* offspring gets a reproduction chain value of two, and so on. This allows us to see the longest autonomous lineages that existed in the simulation. Unsurprisingly, the Sum and Squash methods achieved the highest reproduction chain value, with a chain of 8 (or 9, depending on how you count it) generations of animats maintaining their lineage through autonomous reproduction. CTNN achieved a max reproduction chain of approximately 4, and NARS achieved a max reproduction chain of approximately 3.

Looking at the number of neural connections that evolved, in *Figure 18*, we see that, on average, the CTNN no learning method and both Sum and Squash methods evolved roughly the same number of connections. The CTNN Hebb was an outlier, having nearly twice as many connections as any other method at any given point in time. This is interesting to note, since despite evolving more connections than any other neural method, it also performed worse than all other neural methods. This data suggests that the CTNN Hebb method may have evolved more connections to compensate for its poor performance, only performing better as more connections were added.

Figure 19 shows the evolved values for the NARS personality parameters. The k value appeared to stay roughly constant over time, on average. This means that changing the weighting of unit evidence was not particularly helpful. The T value, on the other hand, decreased throughout the simulation, dipping below 0.5 to 0.45 on average. This has the effect of making NARS more impulsive, or less cautious, allowing NARS to pursue goals with even low desirability. This may have evolved because early NARS animats were too inactive or inhibited, so removing those inhibitions caused the NARS animats to be more active.

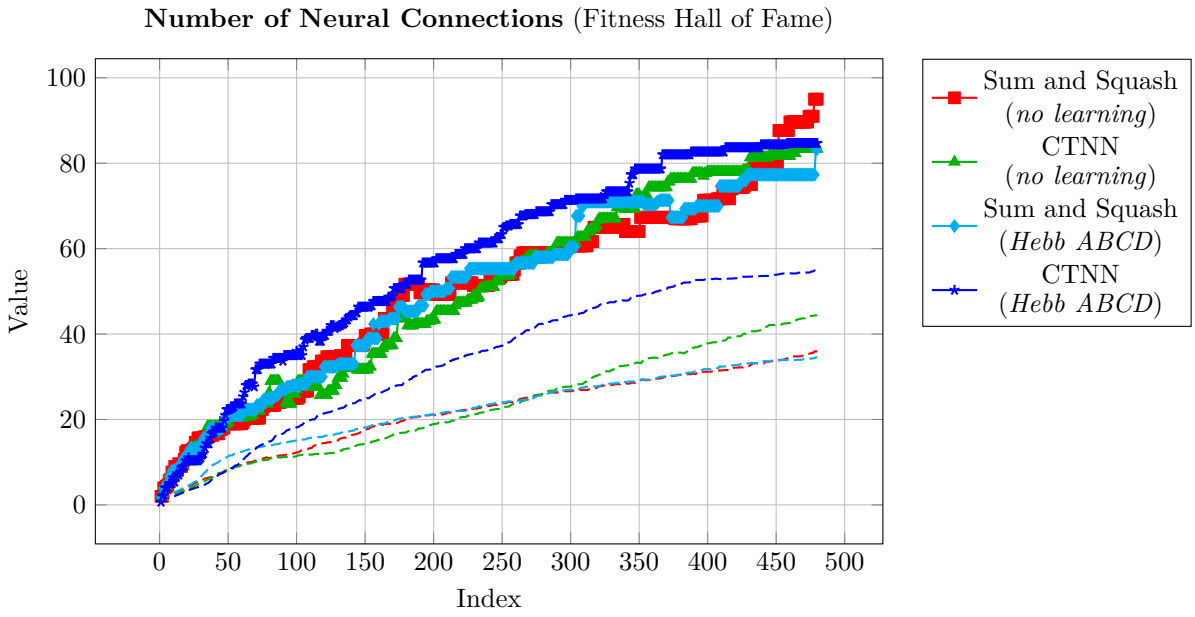


Figure 18: The number of synapses for animats in the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum number of synapses, the dashed line is the mean number of synapses of the table ($n = 100$).

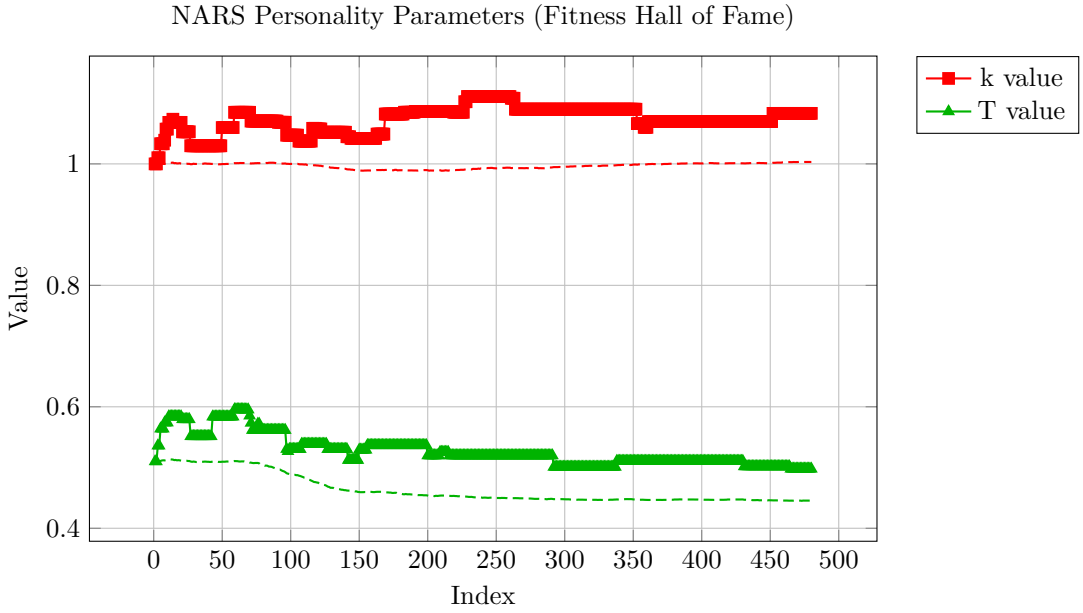


Figure 19: The NARS personality parameters for animats in the Fitness Hall of Fame animat table, in the *Mind* experiments. The bold line is the maximum NARS value, the dashed line is the mean NARS value of the table ($n = 100$).

4.3 Experiments in *Body*

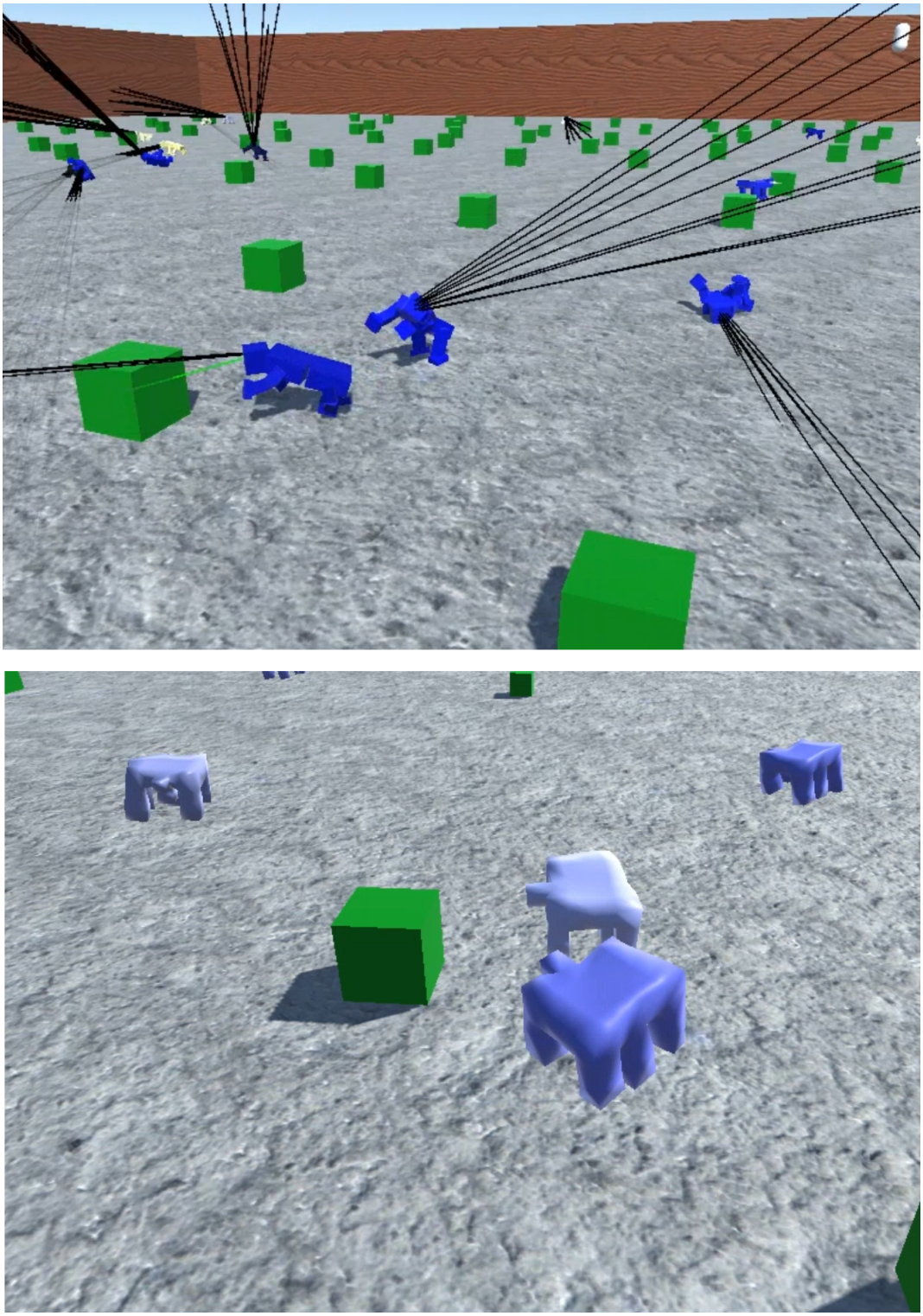


Figure 20: Articulated robot animats (top) and soft voxel robot animats (bottom), seeking and eating food.

For the body, 3 different configurations were tested: wheeled robot, articulated robot, and soft voxel robot (see *Figures 20,24,25*). The mind was Sum and Squash with no learning, except Articulated Robot which used Sum and Squash Hebb, because they did not move much at all without Hebb. The environment was the static flat plane.

Depending on the robot, certain initial connections were manually placed to give evolution a "hint" or "head start". For the wheeled robot, no initial connections were placed. For the articulated robot, in each segment, the 10 sensory neurons were fully connected to the 3 motor neurons. For the soft voxel robot, in each voxel, the 3 sensory neurons were fully connected to the 3 motor neurons. This provided a built-in reflex mechanism, by giving each body part a direct sensorymotor path.

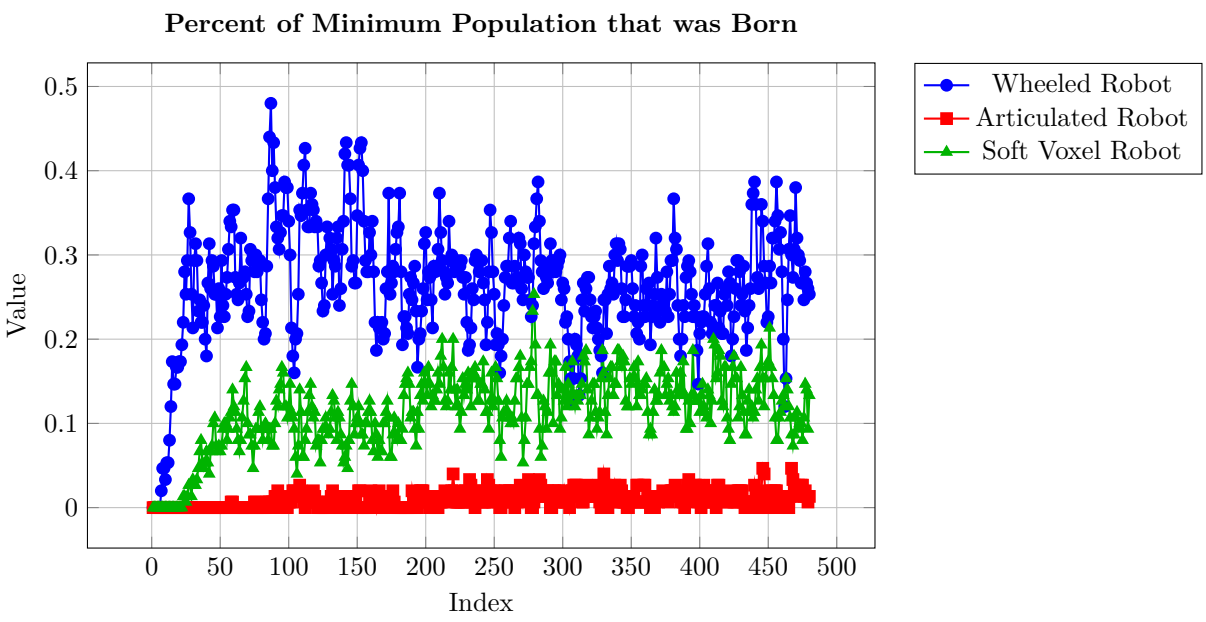


Figure 21: Ratio of birthed animats to total population in the *Body* experiments.

In the "percent born" graph, *Figure 21*, we can see a striking difference between the performance of the three robotic body types. The wheeled robot achieved about a 48% ratio at most, the soft voxel robot achieved about a 26% ratio at most, and the articulated robot achieved about a 5% ratio at most.

This drastic difference in performance is not surprising when we consider the properties of each robot type. Obviously, the wheeled robot as able to far outperform the other two methods, because the wheeled robot does not have to move its limbs in a special complicated way to move its body forward; it merely has to activate one neuron that directly locomotes its body forward. On the other hand, the soft voxel robot had to pump its voxels rhythmically, pushing off the ground, and the articulated robot also had to push off the ground, by rotating and pushing with its limbs. It is also not surprising to see that the soft voxel robot outperformed the articulated robot. Being soft and deformable, the soft voxel robot has a wide and flat smushed base that is hard to knock over. The robot can literally "bounce back", and absorb the force from, disruptions to its overall movement. Compare this to the rigid articulated robot, where a body segment does not deform to grip the ground but instead often has only one or a few unstable points of contact with the ground, making it more unstable and finicky.

What is somewhat surprising to see is the very poor performance of the articulated robots. Though they were, in fact, able to seek out food, the numbers show that they still did not succeed very often at reproduction. This result is unlikely to be caused by the use of Hebbian learning, since as we saw in *Section 4.2* Hebbian learning did not much impact performance, and even improved it for Sum and Squash. Therefore, it seems we can conclude that the articulated robots are rather difficult to control and maneuver successfully. This result is significant, since the robots used in all sorts of robotics research are often of the articulated type, and it may be that robustness and performance can be improved by using soft robots.

We can examine the fitness scores in *Figure 22* and the food eaten in *Figure 23* for further analysis. Of

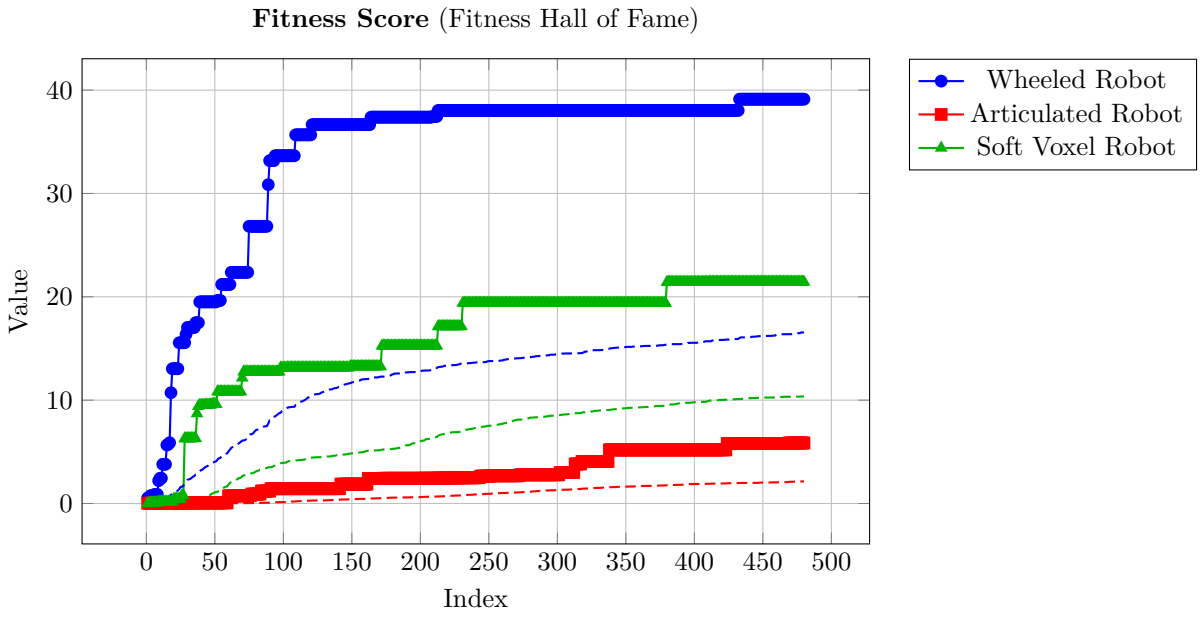


Figure 22: The fitness scores in the Fitness Hall of Fame animat table, in the *Body* experiments. The bold line is the maximum score achieved, the dashed line is the mean score of the table ($n = 100$).

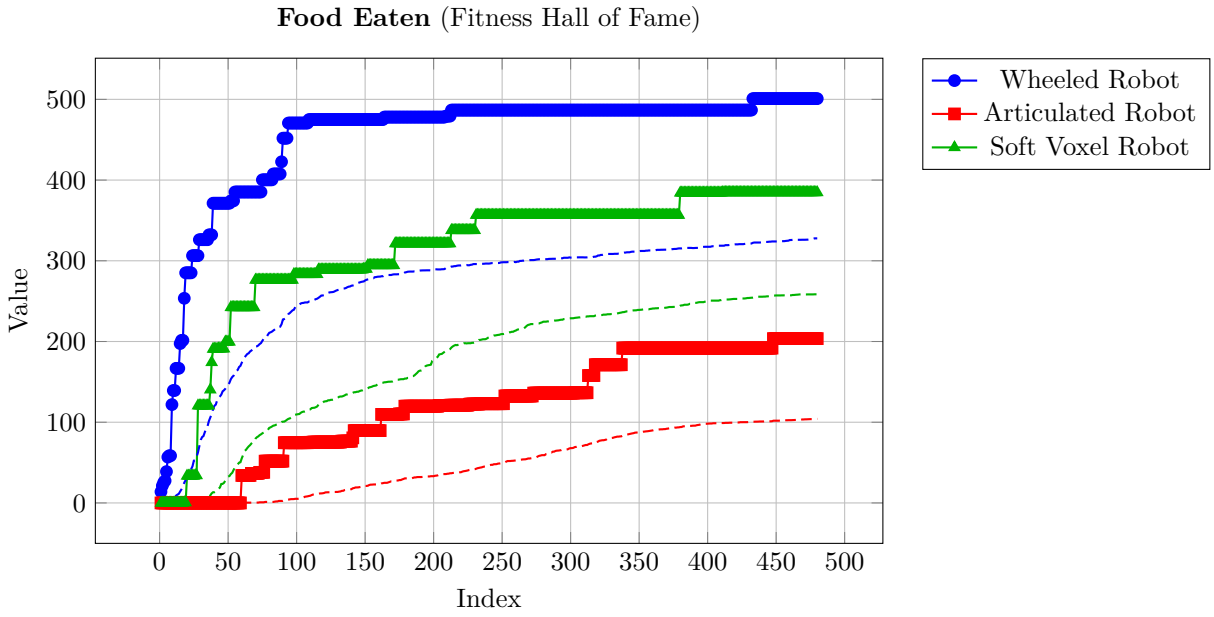


Figure 23: The food eaten scores in the Fitness Hall of Fame animat table, in the *Body* experiments. The bold line is the maximum food eaten achieved, the dashed line is the mean food eaten of the table ($n = 100$).

course, the relative ordering of performance between body types remains the same, with wheeled robot on top, soft voxel robot in the middle, and articulated robot on bottom. The best wheeled robots scored about $\mathcal{F}_{obj} = 38$ and ate about $E_{\text{food}} = 500$ food, the best soft voxel robots scored about $\mathcal{F}_{obj} = 22$ and ate about $E_{\text{food}} = 400$ food, and the best articulated robots scored about $\mathcal{F}_{obj} = 7$ and ate about $E_{\text{food}} = 200$ food.

This data shows that the soft voxel robots were able to compete quite well with the wheeled robots when it came to eating food, losing out by only about 100 food. This close performance surprising, since the

procedure for moving a soft voxel robot is more complicated than moving the wheeled robot. On the other hand, the articulated robots were far behind the wheeled robots, by 200-300 food. Again, the reason for this major discrepancy may be due to physical rigidity and general instability of the moving articulated robots, which caused them to topple over sometimes.

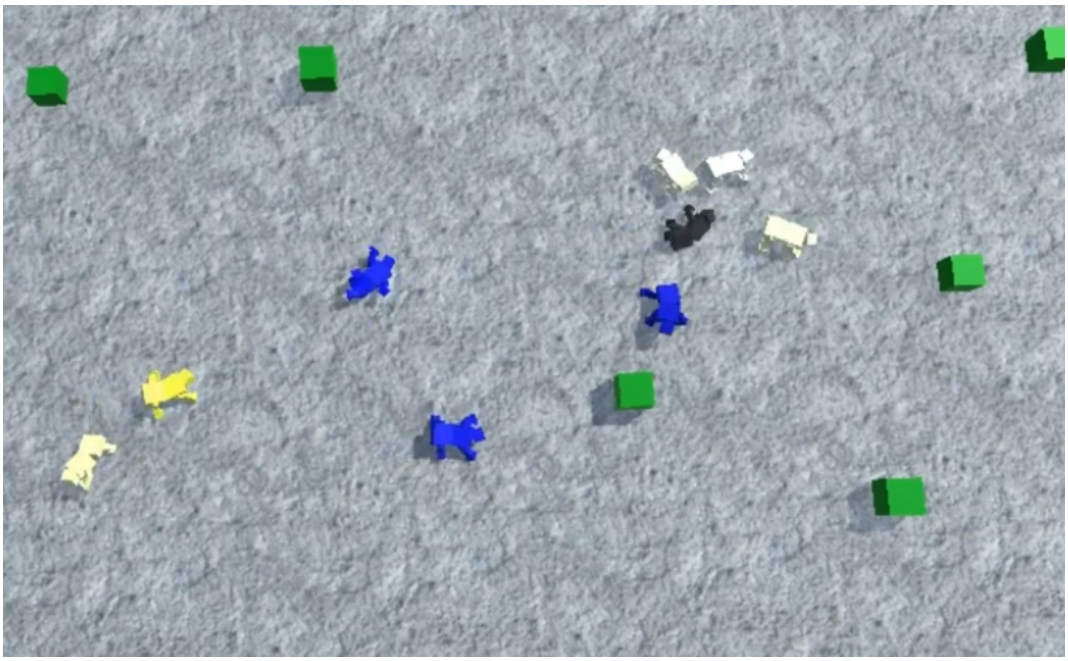


Figure 24: Articulated robot animats from an aerial view.

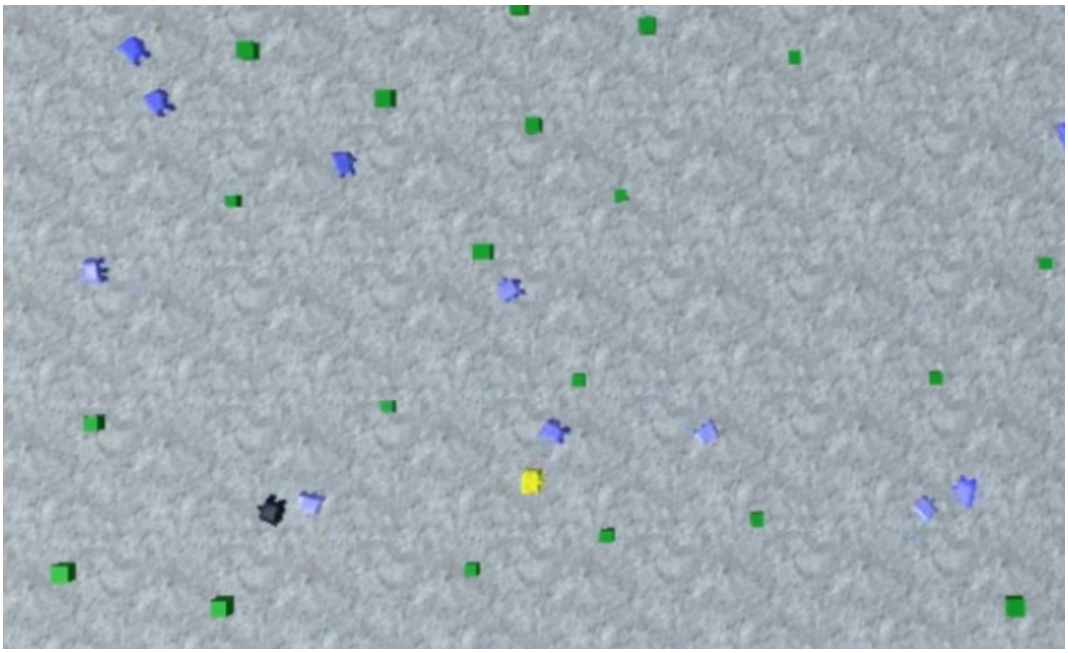


Figure 25: Soft voxel robot animats from an aerial view.

Examining *Figure 26* and *Figure 27*, we can see that though the wheeled robots did not much differentiate between asexually and sexually reproducing, the more complex articulated robots and soft voxel robots did

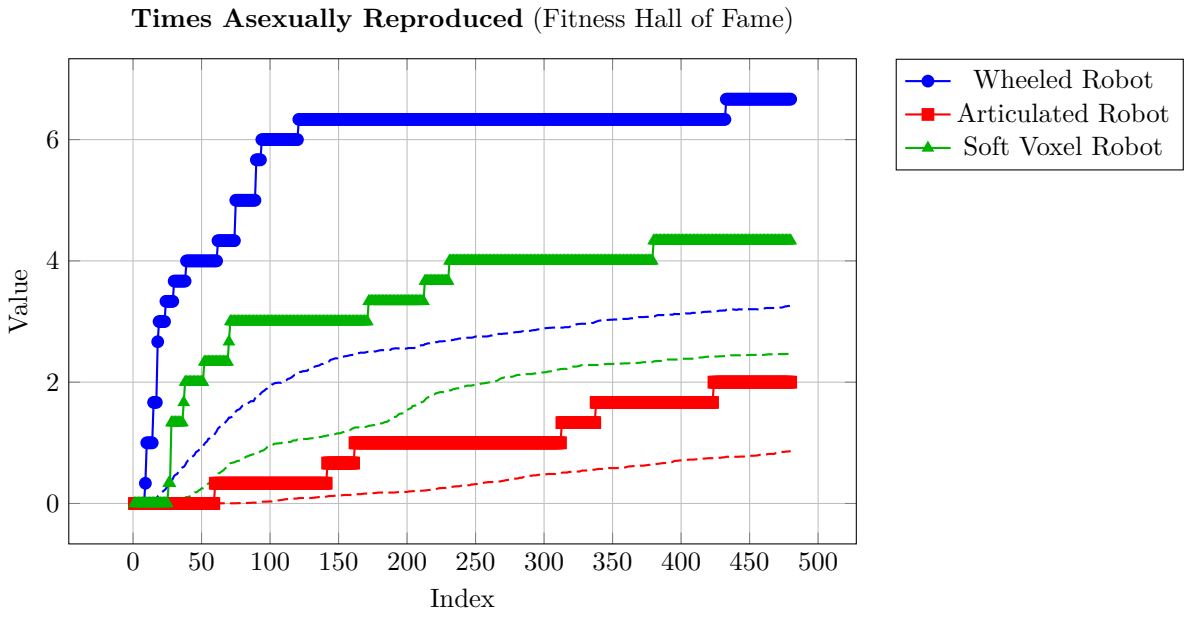


Figure 26: The times asexually reproduced in the Fitness Hall of Fame animat table, in the *Body* experiments. The bold line is the maximum times asexually reproduced, the dashed line is the mean times asexually reproduced of the table ($n = 100$).

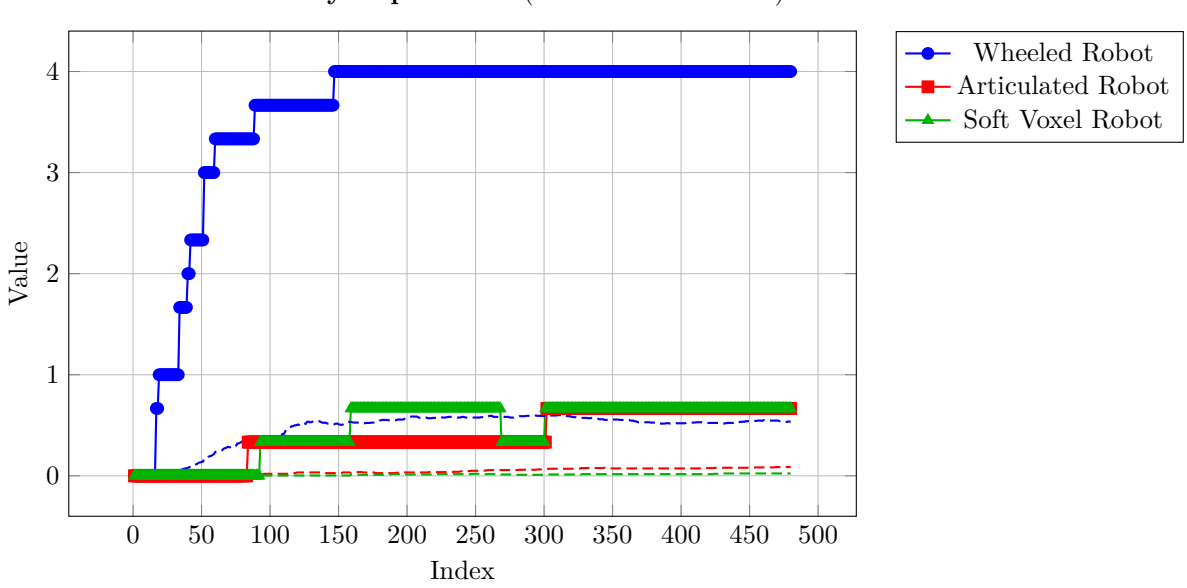


Figure 27: The times sexually reproduced in the Fitness Hall of Fame animat table, in the *Body* experiments. The bold line is the maximum times sexually reproduced, the dashed line is the mean times sexually reproduced of the table ($n = 100$).

take full advantage of the efficiency of asexual reproduction, mostly foregoing sexual reproduction. Though, surprisingy, there were some instances of sexual reproduction in the complex robots, and the behavior apparently occurred slightly more frequently over the course of the simulation. For asexual reproduction, the best wheeled robots asexually reproduced 6 times, the best soft voxel robots asexually reproduced 4 times, and the best articulated robots asexually reproduced 2 times. For sexual reproduction, the best

wheeled robots sexually reproduced 4 times, while the best soft voxel robots and articulated robots barely sexually reproduced one time. The reproduction chain results in *Figure 28* also show the wheeled robot with major success, maxing out at a chain of 8, while the soft voxel robots had minimal success, with a maximum chain of 2, and the articulated robots having little to no success in achieving a reproduction chain.

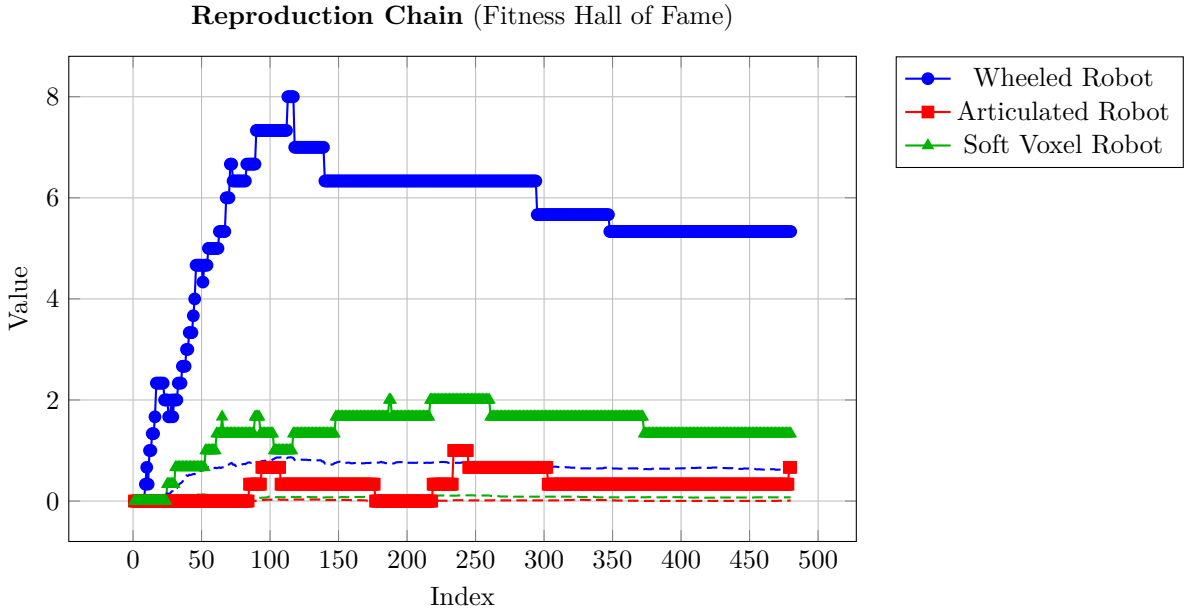


Figure 28: The reproduction chain in the Fitness Hall of Fame animat table, in the *Body* experiments. The bold line is the maximum reproduction chain, the dashed line is the mean reproduction chain of the table ($n = 100$).

4.4 Experiments in *Environment*

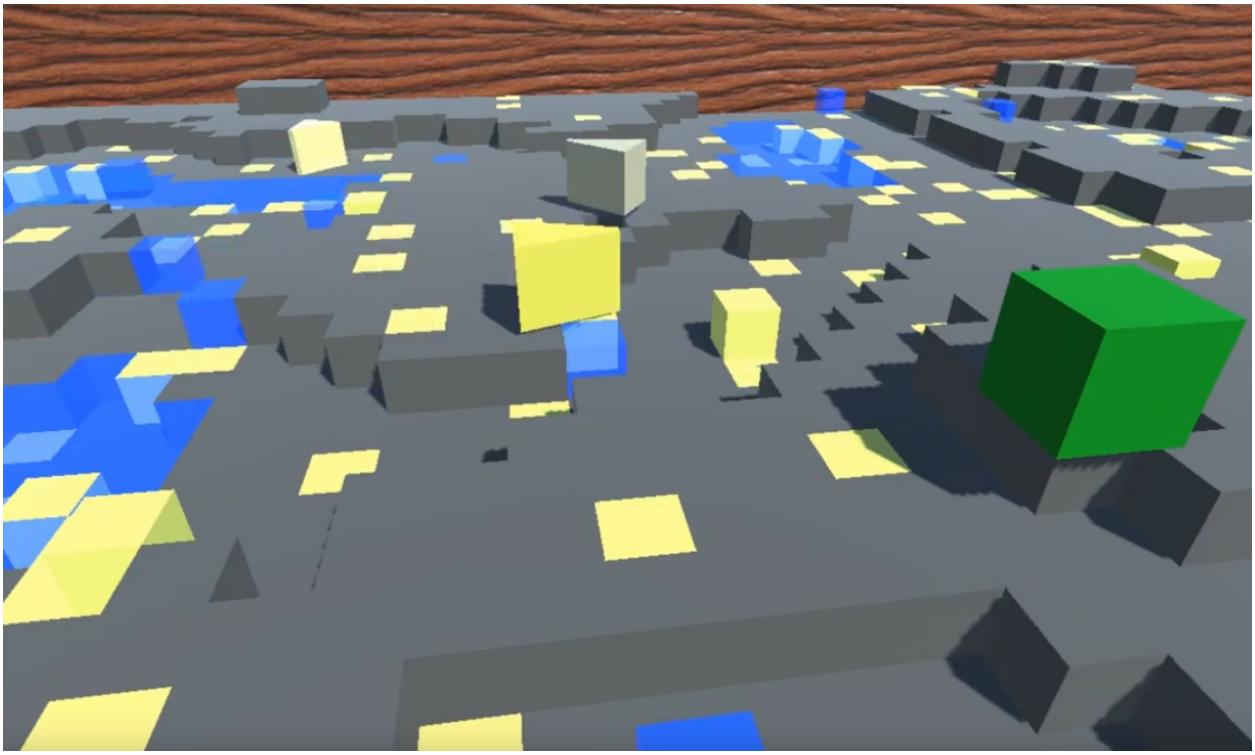


Figure 29: Animats in the interactive voxel world environment.

For the environment, an interactive voxel world of the "falling-sand simulator" variety was tested. The purpose of testing a voxel environment in place of the flat environment is two-fold. Firstly, a voxel world opens up more actions and activities for the animats to perform,such as moving voxels, building structures, and digging tunnels. This makes the simulation more interesting for us, the animats more useful in general by acquiring more activities, and provides opportunities for the animats to evolve and use behaviors related to "intelligence". Secondly, a voxel world is more difficult than a flat world, both to navigate and interact with. The boxy and mountainous terrain is difficult to navigate, while the interactions and moving voxels make the world less predictable. The difficulty of the environment is an important concept related to the evolutionary pressure of intelligence, as described by Wilson [Wilson, 1986, Wilson, 1991]. The idea is to make the environmental pressures increasingly more difficult, to evolve increasingly greater intelligence. The voxel world is a step in this direction.

Figure 30 showcases the increased difficulty of the voxel environment over the flat world. Indeed, the environment was significantly more difficult, such that, compared to the 48% maximum birth ratio achieved in the flat plane, the animats could only achieve up to a 10% maximum birth ratio in the voxel world. In both cases, the performance seemed to increase fairly rapidly then plateau for the rest of the simulation, with the plateau starting around index 50 for the flat world animats, and starting around index 150 for the voxel world animats. Qualitatively, the animats' behaviors in the voxel world were much less directed and focused than those in the flat world. Unlike the animats in the flat world, the animats in the voxel world tended to bounce around and spin, not locking in on food. It seemed they were much less able to make sense of, and navigate, the voxel environment.

Despite the fact that the animat performance was relatively poor in the voxel world compared to the flat world, the voxel world animats' fitness (*Figure 31*) and ability to eat food (*Figure 32*) plateaued around index 200, at around 12 fitness and 300 food, compared to the flat world's animats which plateaued at 38 fitness and 500 food. The movement of the animats was not necessarily slowed down by navigating the voxel world, and in fact the animats could move around quite quickly. So, the limiting problem is not the environment,

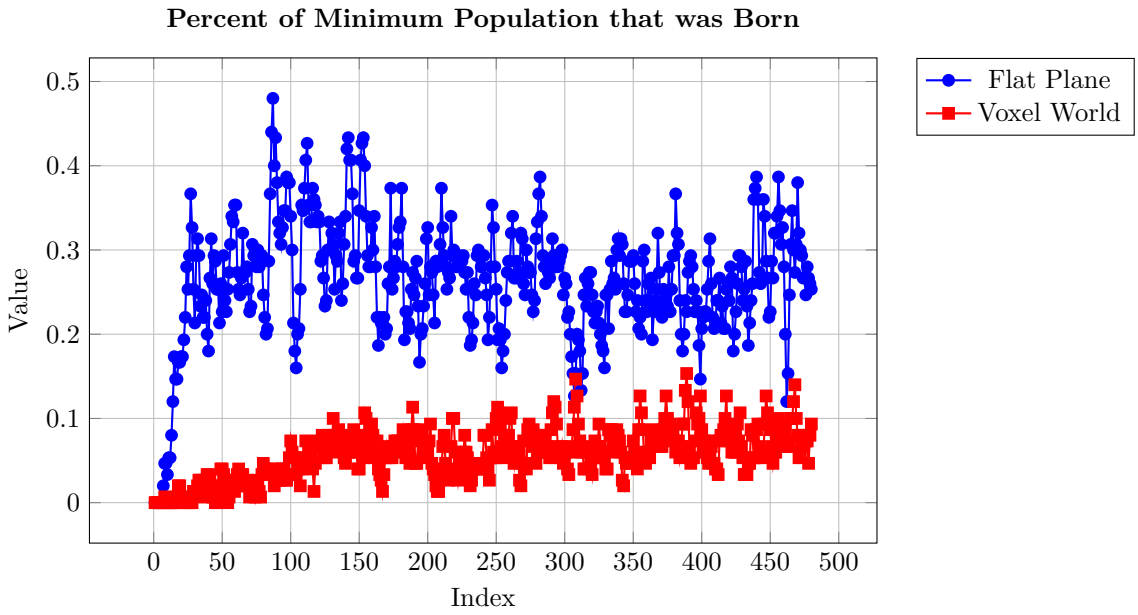


Figure 30: Ratio of birthed animats to total population in the *Environment* experiments.

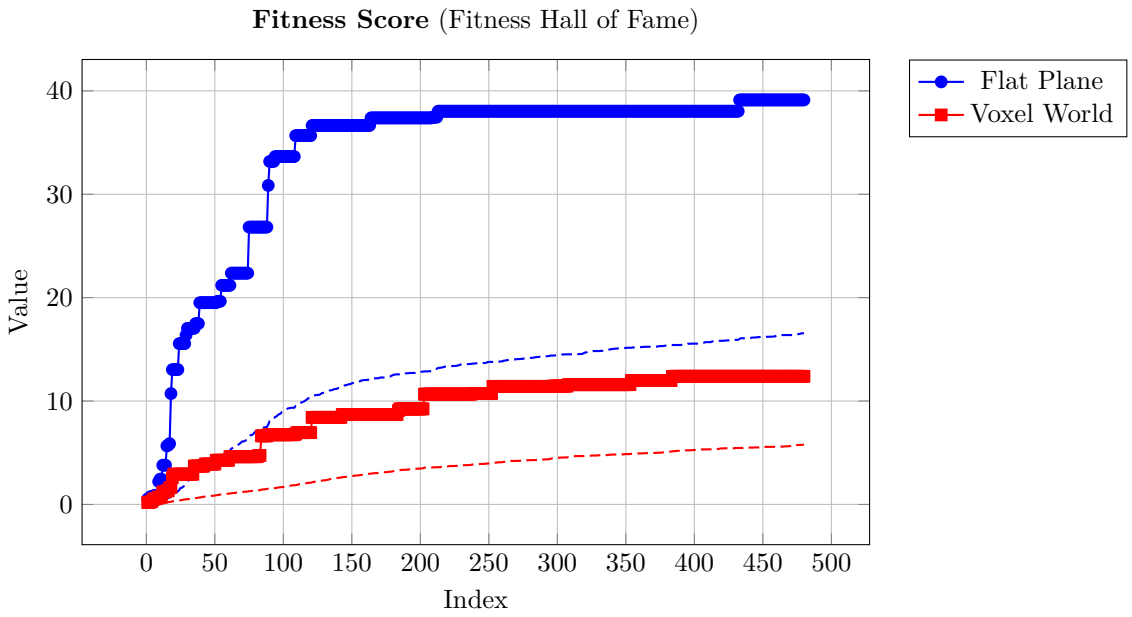


Figure 31: The fitness scores in the Fitness Hall of Fame animat table, in the *Environment* experiments. The bold line is the maximum score achieved, the dashed line is the mean score of the table ($n = 100$).

it is not impossible to navigate. It is also not likely to be a limitation in the evolutionary algorithm, since it was effective in the flat world, and the fitness did moderately improve over time, such that the best animat ate up to 300 food. This data suggests some limitation in either the cognitive algorithm, in that it cannot handle the necessary computations to navigate the voxel world effectively, or a limitation in the robotic body, specifically an impoverished sensory experience, and the body may need a richer sensory apparatus to effectively perceive the Voxel World. Most likely, the robotic body is the limitation, and a richer sensory experience would improve performance, especially directional touch sensors and a moveable vision sensor; in that case, we may simply want to use the articulated robots, or the soft voxel robots (if one can figure out

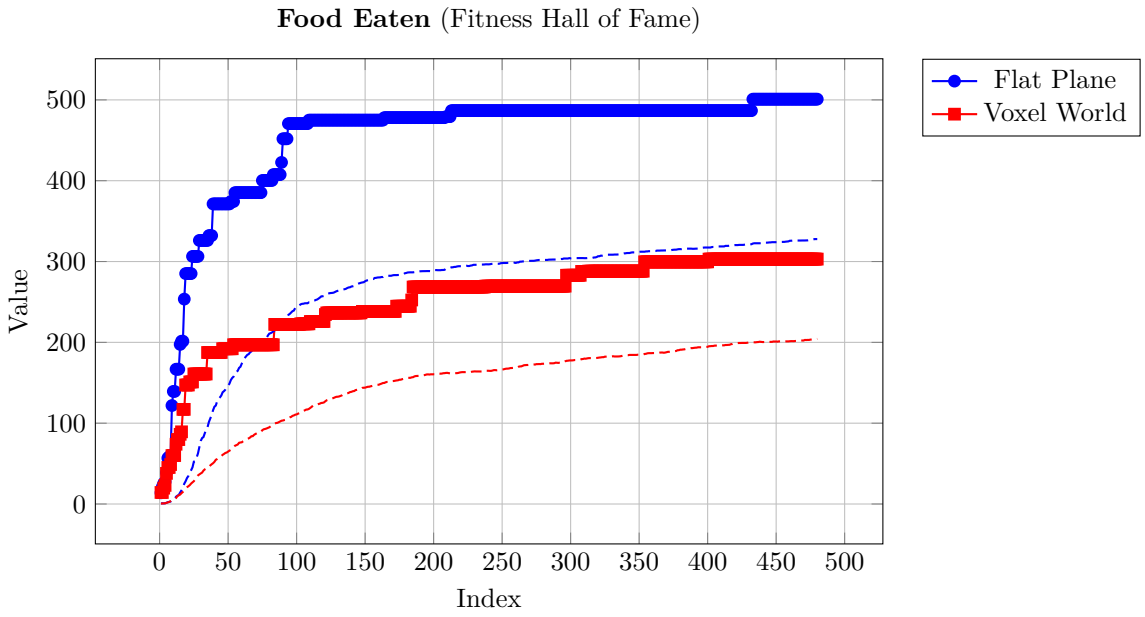


Figure 32: The food eaten in the Fitness Hall of Fame animat table, in the *Environment* experiments. The bold line is the maximum food eaten, the dashed line is the mean food eaten of the table ($n = 100$).

how to translate the voxel world structure to generate the proper forces in the *Voxelyze* engine).

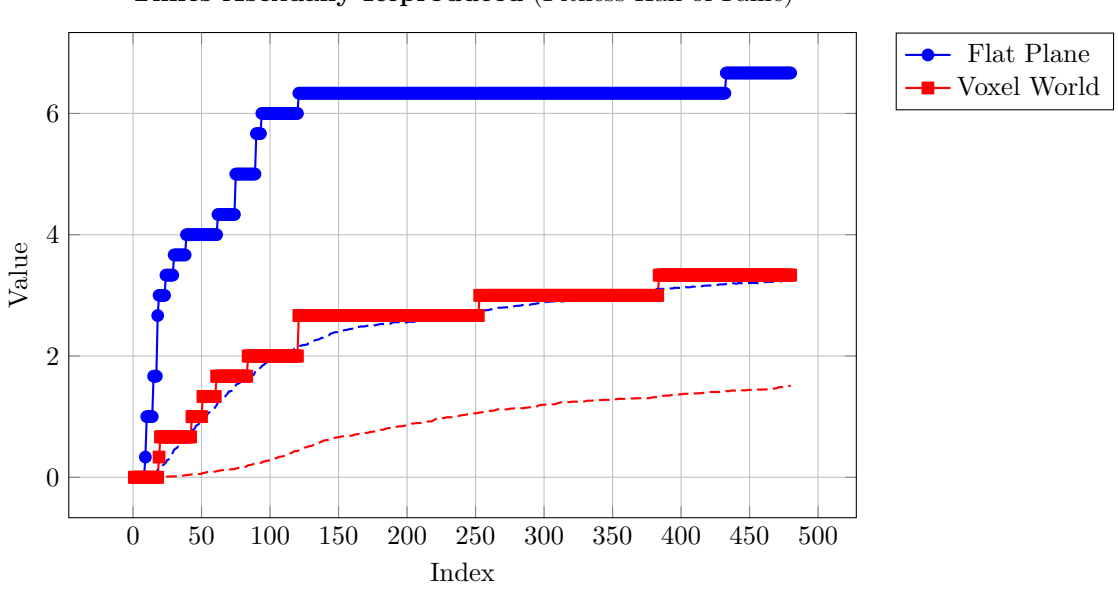


Figure 33: The times asexually reproduced in the Fitness Hall of Fame animat table, in the *Environment* experiments. The bold line is the maximum times asexually reproduced, the dashed line is the mean times asexually reproduced of the table ($n = 100$).

The reproduction performance of animats in the voxel world was about half that of the animats in the flat world. The maximum asexual reproduction score *Figure 33* plateaued around 3 for the voxel world animats, compared to 6 for the flat world animats. Similarly, the maximum sexual reproduction score *Figure 34* plateaued around 3 for the voxel world animats, compared to 6 for the flat world animats. On average,

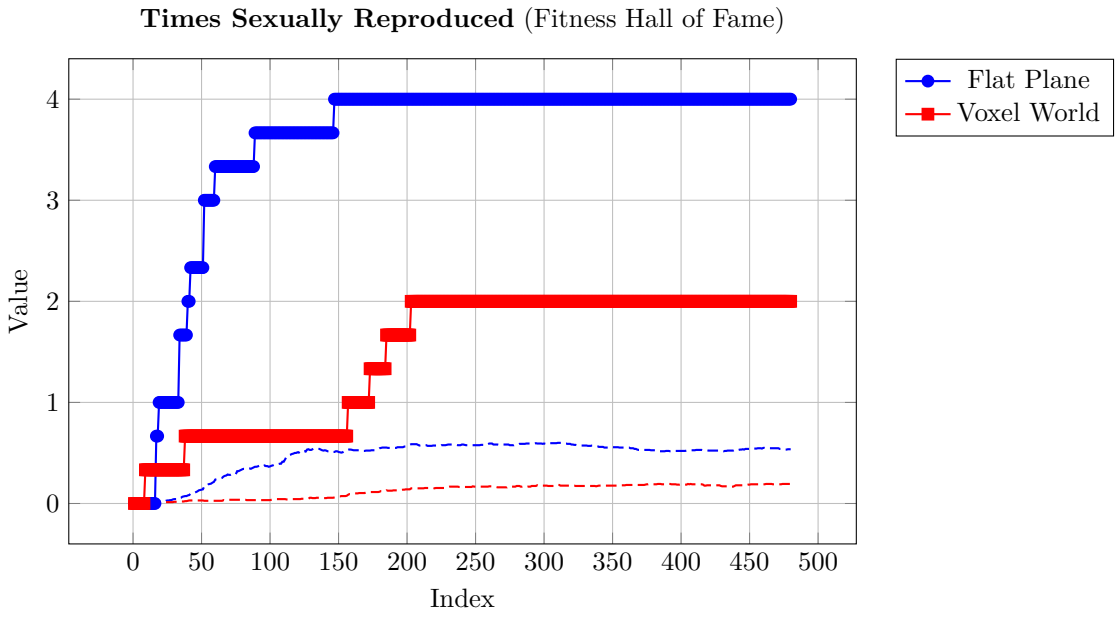


Figure 34: The times sexually reproduced in the Fitness Hall of Fame animat table, in the *Environment* experiments. The bold line is the maximum times sexually reproduced, the dashed line is the mean times sexually reproduced of the table ($n = 100$).

the animats in both worlds tended to asexually reproduce rather than sexually reproduce, which is logical considering the ease of asexual reproduction compared to sexual reproduction.

In comparing the reproduction chains of the two methods using *Figure 35*, we can see that the performance of the best voxel world animats was one-fourth that of the best flat world animats, with the voxel world animats achieving a maximum reproduction chain of 2, and the flat world animats achieving a maximum reproduction chain of 8. On average, even the best voxel world animats had 0 reproduction chain, and the flat world animats had 1 reproduction chain. The performance on this metric could be improved by improving the reproduction score in general.

The animats were able to pick up and place down the sand voxels in the world, with a carrying capacity of up to 5 at a time. The robots did in fact pick up and place down sand voxels, though in a random-like manner, without any apparent rhyme or reason. They frequently would pick up a voxel and immediately place it down again, though often in a different spot. The animats did not exhibit a clear purpose in interacting with voxels, such as building structures, burrowing and tunneling, or interacting with other animats.

Figure 36 shows the impact of the animats on the environment, how they changed the landscape. Most of the environment was made of stone voxels, which were not made movable to the animats, therefore the major structure of the environment remains the same. However, the sand was redistributed by the animats. Though it may be difficult to compare the entire maps visually in *Figure 36*, compare one or two spots across the images to see how the sand voxels changed. Arguably, the sand voxels were more clumped together at the end of the simulation than at the beginning, though they are still mostly evenly/randomly distributed in both cases.

One environmental impact is clear, the animats made something of a mess out of their water features. At the beginning of the simulation, there was no sand in any of the bodies of water, except perhaps submerged at the bottom. By the end of the simulation, various bodies of water were littered with sand blocks placed there by the animats. This has the effect of displacing water voxels, which may cause them to flow into different areas of the map, dynamically changing the landscape, though in this case only in a minor fashion. It is interesting to imagine how the landscape might change if more of it were interactable, and if there were other types of voxels.

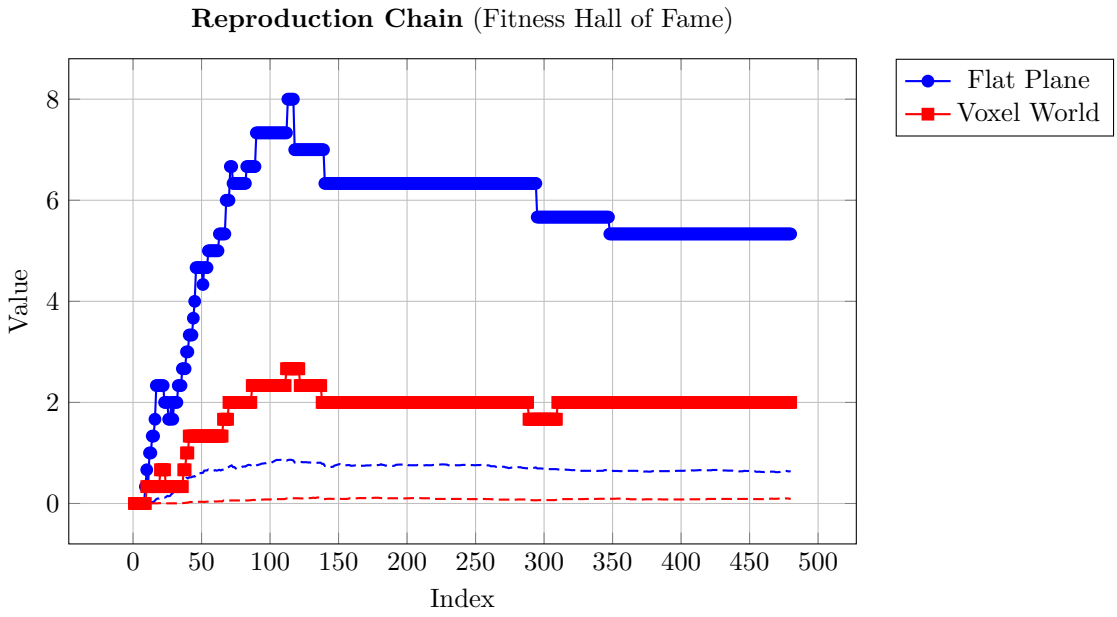


Figure 35: The reproduction chain in the Fitness Hall of Fame animat table, in the *Environment* experiments. The bold line is the maximum reproduction chain, the dashed line is the mean reproduction chain of the table ($n = 100$).

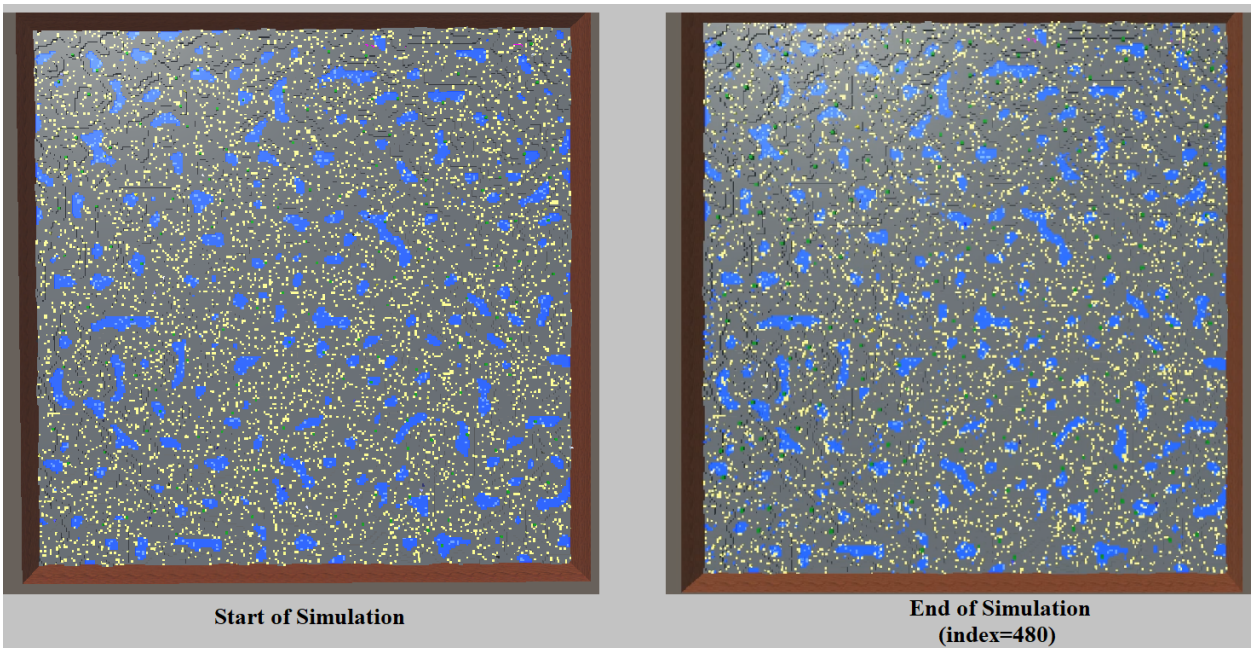


Figure 36: The terrain at the start of the simulation, before the animats modified it (left) and at the end of the simulation, after the animats modified it (right).

5 Conclusion

This work accomplishes the goal set out in [Hahm, 2022], to evolve a simulated ecosystem of robotic AI organisms that exhibit the minimum desired behaviors of self-sustaining their energy and reproducing themselves sexually and asexually while operating with limited resources.

This work tested and compared various cognitive algorithms for artificial animals. Various abstractions of biological neural networks were tested, including a less realistic abstraction (Sum and Squash), a more realistic abstraction (CTNN), real-time learning (Hebb ABCD). Furthermore, NARS, a cognitive algorithm that does not use neural networks at all, but instead uses symbolic logical reasoning, was tested. All cognitive methods were able to accomplish the task to an acceptable degree, demonstrating a clearly improved ability to seek food and reproduce.

This work also tested various frameworks of robotic bodies for artificial animals. The first framework, the Wheeled Robot, is less realistic than the other robots when compared to real animals, but was excellent for testing the other categories such as the mind and evolutionary algorithm. The second framework, the Articulated Robot, is the popular depiction of robots and is often used in evolutionary robotics. They are quite physically rigid and so difficult to maneuever, but they are incredibly interesting to watch, and they are quite complex, having many body segments and joints to control, which gives them a somewhat animalistic movement. The third framework, the Soft Voxel Robot, is a newer type of robot composed of many soft cubes, where each cube can deform and have unique material properties. Since they are floppy and droopy, it is somewhat hard to get this type of robot walking, but once they know how to locomote, they are quite good at it and can maneuver around quickly. Since they are soft and deformable, they are perhaps the most realistic framework when it comes to simulating animalistic forms, which are often somewhat squishy. All 3 robot types were able to accomplish the task to an acceptable degree, demonstrating a clearly improved ability to seek food and reproduce.

Finally, this work tested various environments in which the animats live and interact. The first framework was the Flat Plane, a simple world which provided the animats a guaranteed direct path and line of sight to each other and the food blocks. The second framework was the Interactive Voxel World, a world made of granular particles (stone, sand, and water) that move according to an artificial physics computed using a cellular automaton. Animats were able to pick up and place down sand voxels in the voxel world, giving them a form of environment interaction and an activity that could benefit from complex cognition. Animats in the voxel world performed less well than those in the flat world, due to the increased difficulties introduce by the environment, though even in the voxel world the animats were able to accomplish the task to an acceptable degree, demonstrating a clearly improved ability to seek food and reproduce.

The novelty of these experiments is as follows. NARS was evolved for the first time. Soft voxel robots were evolved in an animat ecosystem for the first time. A voxel cellular automaton world was used in the context of an animat ecosystem for the first time. When examining the results, all of these methods show promise for future experiments. NARS evolution was successful, and the same genetic encoding developed here can be used even for tasks outside of the animat domain. The soft voxel robots evolved to better use vision sensors to seek out food objects and each other, and could be tested with a richer sensory apparatus, especially vision and modalities unusual in AI like hearing. Finally, it was very interesting to watch the animats climb around the voxel environment and displace voxels, and it would surely only be more interesting if this aspect was enriched, though also surely more difficult to evolve proficient creatures.

Using only the results found here, I would make the following recommendations for future experiments in this domain. Firstly, use Sum and Squash with Hebbian learning, because the performance is superior to the other methods, and the Hebbian learning allows the combination of evolution and real-time learning. Secondly, when it comes to complex robots, prefer soft voxel robots over articulated robots, since soft voxel robots appear more robust and perform better, though test both, just do not expect the articulated robot to perform as well. Thirdly, make the environment composed with a higher density of interactable compared to non-interactable voxels, to allow more drastic shifts in the terrain and more opportunities for the animats to evolve better voxel interaction abilities such as tunneling.

For the future experiments, I would make a couple suggestions. For the mind, to try other abstractions of the mind, especially spiking networks, since they are closer to biological neurons. For the robotic body, it would be interesting to try controlling the robot with other mind methods such as learning, to try different robot morphologies, and to test the complex robots in the voxel world. It would also be interesting to test the evolution of NARS for complex robots, like soft voxel robots. For the environment, I would suggest adding dangerous and lethal obstacles to the environment, so that the animats cannot mindlessly mill around to get food, but instead must pay close attention to their movement path to prevent death. For the evolutionary algorithm, it is certainly necessary to test different hyperparameter values like mutation rates, since the ones in these experiments may have been suboptimal. Finally, there should be increased incentive for sexual

reproduction, and perhaps asexual reproduction should be removed altogether. This seems plausible since all methods did achieve sexual reproduction at least once, even with asexual reproduction available. The reason to focus on sexual reproduction is because finding mates is an important environment challenge, if not the ultimate environmental challenge, for all intelligent animals.

Overall, the experiments resulted in various types of animats and animat ecosystems, each with its own unique and interesting properties, strengths and weaknesses. Some insights were made about each type of animat-relevant framework which can be referenced in future research. Hopefully, this work is a step towards advancing the field of evolving artificial animals.

Acknowledgements

Built with Unity [Juliani et al., 2018]. Soft voxel robots were simulated using the *Voxelyze* engine [Hiller and Lipson, 2011, Hiller and Lipson, 2014].

Videos

Videos of the simulation can be found at the following links:
Sum and Squash: https://www.youtube.com/watch?v=eo7shmgIb78
NARS: https://www.youtube.com/watch?v=w4K79vpSdHo
Soft Voxel Robots: https://www.youtube.com/watch?v=O6QJGfm5qsI
Articulated Robots: https://www.youtube.com/watch?v=fzvVEnZkjFw
Voxel World: https://www.youtube.com/watch?v=378ee97m28s

References

[Adams and Burbeck, 2012] Adams, S. S. and Burbeck, S. (2012). Beyond the octopus: From general intelligence toward a human-like mind. In *Theoretical foundations of artificial general intelligence*, pages 49–65. Springer.

[Arkoudas, 2023] Arkoudas, K. (2023). Gpt-4 can't reason. *arXiv preprint arXiv:2308.03762*.

[Beer, 2006] Beer, R. D. (2006). Parameter space structure of continuous-time recurrent neural networks. *Neural computation*, 18(12):3009–3051.

[Bongard, 2025] Bongard, J. (2025). Lecture 11. continuous time recurrent neural networks (ctrnns). https://www.youtube.com/watch?v=EBxCpm9DnXM&t=3840s.

[Bongard et al., 2008] Bongard, J. C. et al. (2008). Behavior chaining-incremental behavior integration for evolutionary robotics. In *ALIFE*, pages 64–71. Citeseer.

[Brownlee, 2025] Brownlee, J. (2025). Genetic algorithms. Accessed: 2025-02-14. https://algorithmafternoon.com/books/genetic_algorithm/chapter04/.

[Bubeck et al., 2023] Bubeck, S., Chadrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., Lee, P., Lee, Y. T., Li, Y., Lundberg, S., et al. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4.

[Cheney et al., 2016] Cheney, N., Bongard, J., SunSpiral, V., and Lipson, H. (2016). On the difficulty of co-optimizing morphology and control in evolved virtual creatures. In *Artificial life conference proceedings*, pages 226–233. MIT Press One Rogers Street, Cambridge, MA 02142-1209, USA journals-info ....

[Cheney et al., 2014] Cheney, N., MacCurdy, R., Clune, J., and Lipson, H. (2014). Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. *ACM SIGEVOlution*, 7(1):11–23.

[Crosby, 2020] Crosby, M. (2020). Building thinking machines by solving animal cognition tasks. *Minds and Machines*, 30(4):589–615.

[Dawkins and Krebs, 1979] Dawkins, R. and Krebs, J. R. (1979). Arms races between and within species. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 205(1161):489–511.

[De Carlo et al., 2023] De Carlo, M., Ferrante, E., Ellers, J., Meynen, G., and Eiben, A. (2023). Interacting robots in an artificial evolutionary ecosystem. In *European Conference on Genetic Programming (Part of EvoStar)*, pages 339–354. Springer.

[Eiben and Smith, 2015] Eiben, A. E. and Smith, J. E. (2015). *Introduction to evolutionary computing*. Springer.

[Gomes et al., 2015] Gomes, J., Mariano, P., and Christensen, A. L. (2015). Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 943–950.

[Gunning and Aha, 2019] Gunning, D. and Aha, D. (2019). Darpa's explainable artificial intelligence (xai) program. *AI magazine*, 40(2):44–58.

[Hahm, 2021] Hahm, C. (2021). Nars-python v0.3 — technical overview. NARS Workshop at AGI-21.

[Hahm, 2022] Hahm, C. (2022). Designing naturalistic simulations for evolving agi species. In *International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 296–303.

[Hahm, 2024a] Hahm, C. (2024a). Considering simple triangle vision perception with eye movements in nars. NARS Workshop at AGI-24.

[Hahm, 2025] Hahm, C. (2025). Neuroevolution for vision-based seeking behavior in 3d soft voxel robots. *Artificial Life and Robotics*, pages 1–10.

[Hahm and Suereth, 2025] Hahm, C. and Suereth, R. (2025). Resolving human concerns about ai and technology with non-axiomatic reasoning systems. *European Journal of Science, Innovation and Technology*, 5(1).

[Hahm and Wang, 2025] Hahm, C. and Wang, P. (2025). Nars genetic encoding. Technical Report 23, *Temple University, Department of Computer and Information Sciences, AGI Team.*

[Hahm et al., 2021] Hahm, C., Xu, B., and Wang, P. (2021). Goal generation and management in nars. In *14th International Conference on Artificial General Intelligence (AGI 2021)*, volume 14, pages 96–105. Springer.

[Hahm, 2023] Hahm, C. G. (2023). A framework of artificial matter. Technical Report 18, *Temple University, Department of Computer and Information Sciences, AGI Team.*

[Hahm, 2024b] Hahm, C. G. (2024b). Evolving hebbian neural networks for articulated robot locomotion. Technical Report 20, *Temple University, Department of Computer and Information Sciences, AGI Team.*

[Hammer, 2021] Hammer, P. (2021). *Autonomy through real-time learning and OpenNARS for Applications.* PhD thesis, Temple University.

[Hammer et al., 2016] Hammer, P., Lofthouse, T., and Wang, P. (2016). The opennars implementation of the non-axiomatic reasoning system. In *Artificial General Intelligence: 9th International Conference, AGI 2016, New York, NY, USA, July 16-19, 2016, Proceedings 9*, pages 160–170. Springer.

[Hiller and Lipson, 2011] Hiller, J. and Lipson, H. (2011). Automatic design and manufacture of soft robots. *IEEE Transactions on Robotics*, 28(2):457–466.

[Hiller and Lipson, 2014] Hiller, J. and Lipson, H. (2014). Dynamic simulation of soft multimaterial 3d-printed objects. *Soft robotics*, 1(1):88–101.

[Hornik et al., 1989] Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

[Ireland, 2024] Ireland, D. (2024). Mirabile dictu: Language acquisition in the non-axiomatic reasoning system. In *International Conference on Artificial General Intelligence*, pages 99–108. Springer.

[Juliani et al., 2018] Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627.*

[Kilic, 2015] Kilic, O. (2015). Intelligent reasoning on natural language data: a non-axiomatic reasoning system approach. Master's thesis, Temple University.

[Lehman and Stanley, 2011] Lehman, J. and Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223.

[Li, 2021] Li, X. (2021). *Functionalist Emotion Model in Artificial General Intelligence.* PhD thesis, Temple University.

[Li et al., 2018] Li, X., Hammer, P., Wang, P., and Xie, H. (2018). Functionalist emotion model in nars. In *Artificial General Intelligence: 11th International Conference, AGI 2018, Prague, Czech Republic, August 22-25, 2018, Proceedings 11*, pages 119–129. Springer.

[Minsky, 1991] Minsky, M. L. (1991). Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI magazine*, 12(2):34–34.

[Mirzadeh et al., 2024] Mirzadeh, I., Alizadeh, K., Shahrokhi, H., Tuzel, O., Bengio, S., and Farajtabar, M. (2024). Gsm-symbolic: Understanding the limitations of mathematical reasoning in large language models. *arXiv preprint arXiv:2410.05229*.

[Najarro and Risi, 2020] Najarro, E. and Risi, S. (2020). Meta-learning through hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33:20719–20731.

[Nezhurina et al., 2024] Nezhurina, M., Cipolina-Kun, L., Cherti, M., and Jitsev, J. (2024). Alice in wonderland: Simple tasks showing complete reasoning breakdown in state-of-the-art large language models. *arXiv preprint arXiv:2406.02061*.

[Niv et al., 2001] Niv, Y., Joel, D., Meilijson, I., and Ruppin, E. (2001). Evolution of reinforcement learning in uncertain environments: Emergence of risk-aversion and matching. In *Advances in Artificial Life: 6th European Conference, ECAL 2001 Prague, Czech Republic, September 10–14, 2001 Proceedings 6*, pages 252–261. Springer.

[Russell and Norvig, 2016] Russell, S. J. and Norvig, P. (2016). *Artificial intelligence: a modern approach*. Pearson.

[Sims, 1994] Sims, K. (1994). Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22.

[Stanley and Miikkulainen, 2002] Stanley, K. O. and Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127.

[Strannegård et al., 2021] Strannegård, C., Engsner, N., Ferrari, P., Glimmerfors, H., Södergren, M. H., Karlsson, T., Kleve, B., and Skoglund, V. (2021). The ecosystem path to agi. In *Artificial General Intelligence: 14th International Conference, AGI 2021, Palo Alto, CA, USA, October 15–18, 2021, Proceedings 14*, pages 269–278. Springer.

[Strannegård et al., 2023] Strannegård, C., Engsner, N., Ulfsbäcker, S., Andreasson, S., Endler, J., and Nordgren, A. (2023). Survival games for humans and machines. *Available at SSRN 4587702*.

[Strannegård et al., 2017] Strannegård, C., Svangård, N., Bach, J., and Steunebrink, B. (2017). Generic animats. In *Artificial General Intelligence: 10th International Conference, AGI 2017, Melbourne, VIC, Australia, August 15-18, 2017, Proceedings 10*, pages 23–32. Springer.

[Strannegård et al., 2020] Strannegård, C., Xu, W., Engsner, N., and Endler, J. A. (2020). Combining evolution and learning in computational ecosystems. *Journal of Artificial General Intelligence*, 11(1):1–37.

[Unity, 2024] Unity (2024). Articulation body component reference. https://docs.unity3d.com/6000.0/Documentation/Manual/class-ArticulationBody.html.

[van der Sluis, 2023] van der Sluis, D. (2023). Nuts, nars, and speech. In *International Conference on Artificial General Intelligence*, pages 307–316. Springer.

[Wang, 1995] Wang, P. (1995). *Non-Axiomatic Reasoning System: Exploring the Essence of Intelligence*. PhD thesis, Indiana University.

[Wang, 2013a] Wang, P. (2013a). Natural language processing by reasoning and learning. In *Artificial General Intelligence: 6th International Conference, AGI 2013, Beijing, China, July 31–August 3, 2013 Proceedings 6*, pages 160–169. Springer.

[Wang, 2013b] Wang, P. (2013b). *Non-Axiomatic Logic: A Model of Intelligent Reasoning*. World Scientific, Singapore.

[Wang, 2018] Wang, P. (2018). Perception in nars. Technical Report 7, *Temple University, Department of Computer and Information Sciences, AGI Team*.

[Wang et al., 2022] Wang, P., Hahm, C., and Hammer, P. (2022). A model of unified perception and cognition. *Frontiers in Artificial Intelligence*, 5:806403.

[Wang et al., 2019] Wang, P., Li, X., and Hammer, P. (2019). Sensorimotor in nars. Technical Report 8, *Temple University, Department of Computer and Information Sciences, AGI Team.*

[Wang et al., 2016] Wang, P., Talanov, M., and Hammer, P. (2016). The emotional mechanisms in nars. In *Artificial General Intelligence: 9th International Conference, AGI 2016, New York, NY, USA, July 16-19, 2016, Proceedings 9*, pages 150–159. Springer.

[Weel et al., 2014] Weel, B., Crosato, E., Heinerman, J., Haasdijk, E., and Eiben, A. (2014). A robotic ecosystem with evolvable minds and bodies. In *2014 IEEE International Conference on Evolvable Systems*, pages 165–172. IEEE.

[Wilson, 1986] Wilson, S. W. (1986). Knowledge growth in an artificial animal. In *Adaptive and Learning Systems*, pages 255–264. Springer.

[Wilson, 1991] Wilson, S. W. (1991). The animat path to ai. In *From Animals to Animats: Proceedings of the first international conference on simulation of adaptive behavior*, pages 15–21. MIT Press.

[Yaeger et al., 1994] Yaeger, L. et al. (1994). Computational genetics, physiology, metabolism, neural systems, learning, vision, and behavior or poly world: Life in a new context. In *SANTA FE INSTITUTE STUDIES IN THE SCIENCES OF COMPLEXITY-PROCEEDINGS VOLUME-*, volume 17, pages 263–263. Citeseer.

Preliminary 2 Exam:
Experiments in Advancing the Evolution of Artificial Animals

Christian Hahm
christian.hahm@temple.edu

Department of Computer and Information Sciences
Temple University

March 26, 2025

Abstract

This paper details the results of various experimental setups for the evolution of artificial animals.

Keywords: *artificial animal, AI*

APPENDIX

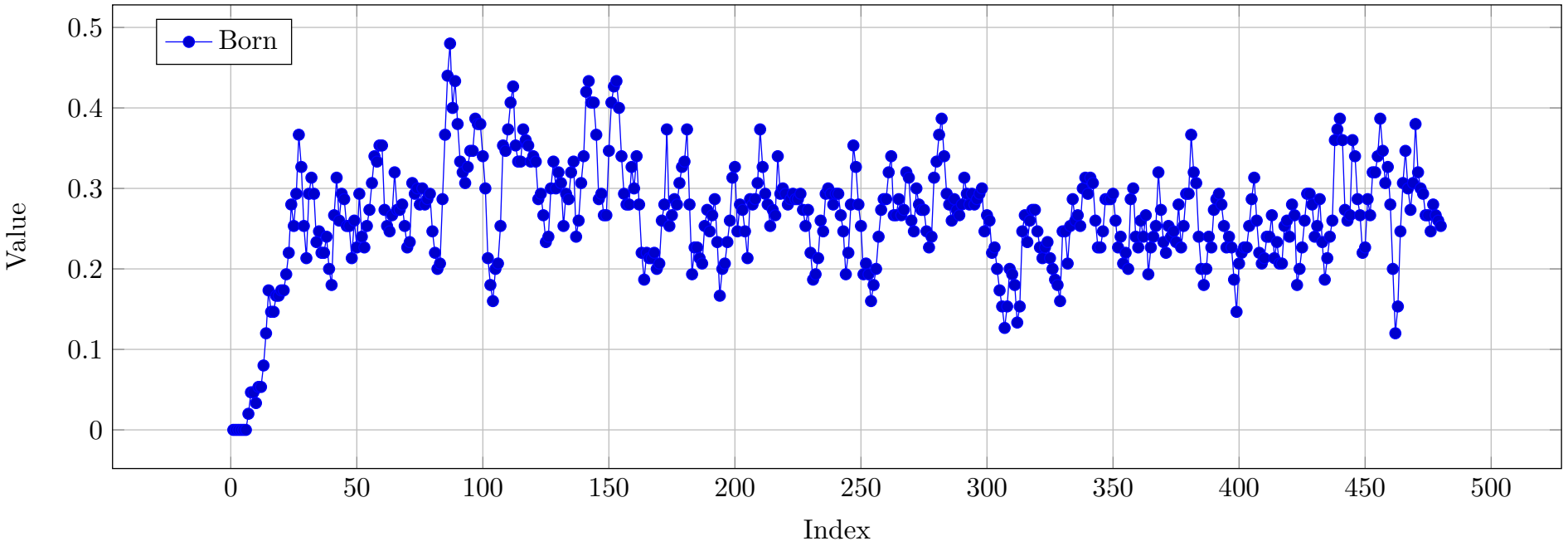
Contents:

- Wheeled Robot - Sum & Squash - No Learning - Flat World
- Wheeled Robot - Sum & Squash - Hebb ABCD - Flat World
- Wheeled Robot - CTRNN - No Learning - Flat World
- Wheeled Robot - CTRNN - Hebb ABCD - Flat World
- Wheeled Robot - NARS - No Learning - Flat World
- Soft Voxel Robot - Sum & Squash - No Learning - Flat World
- Articulated Robot - Sum & Squash - Hebb ABCD - Flat World
- Wheeled Robot - Sum & Squash - No Learning - Interactive Voxel World
- Wheeled Robot - Random - Flat World

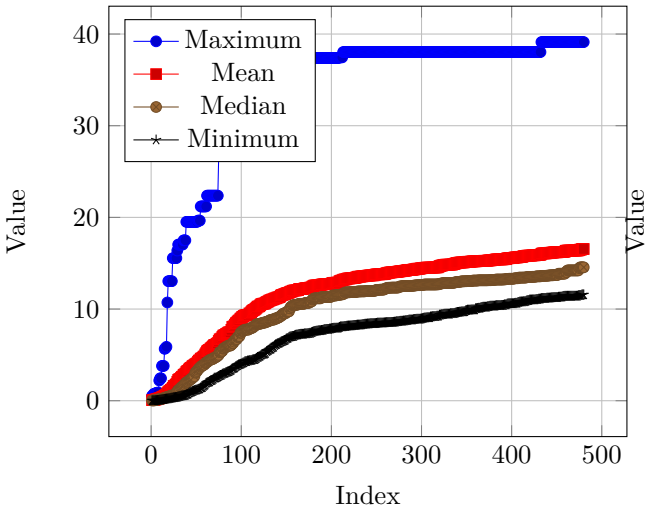
Wheeled Robot - Sum & Squash - No Learning - Flat World

RESULTS

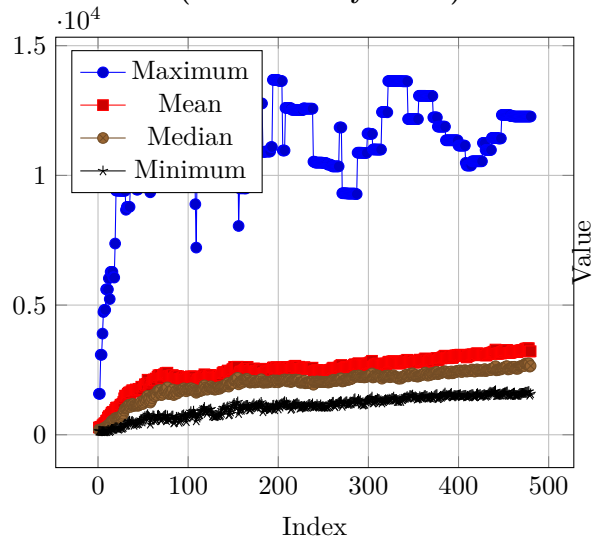
Percent of Minimum Population that was Born



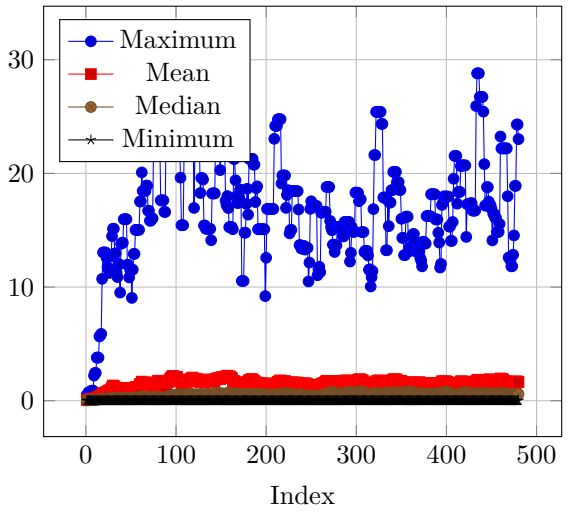
Fitness Score (Elite Fitness Table)



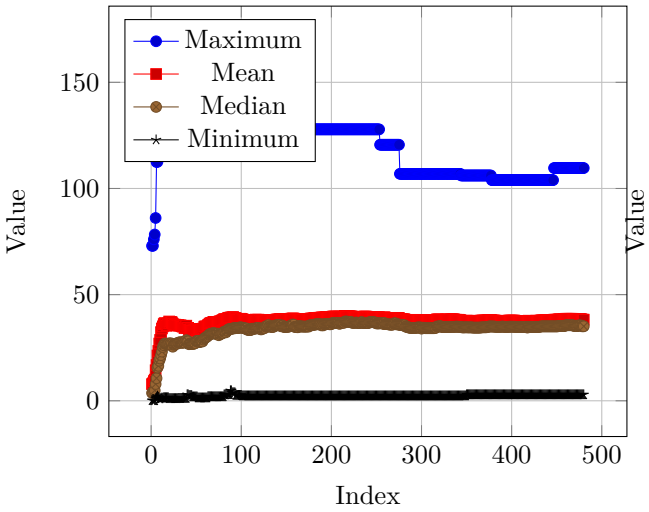
Novelty Score (Elite Novelty Table)



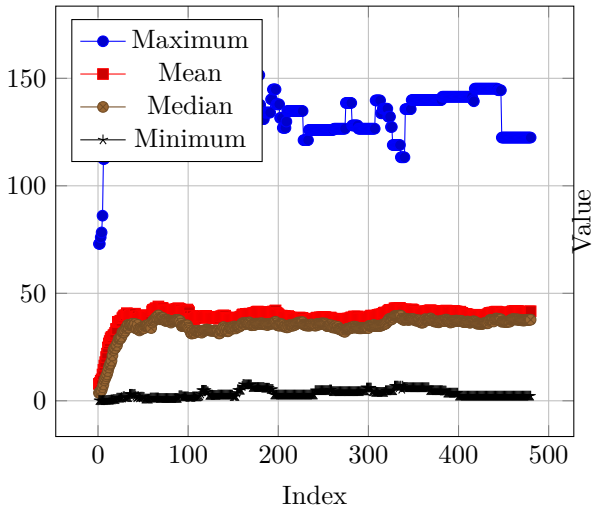
Fitness Score (Recently Dead Table)



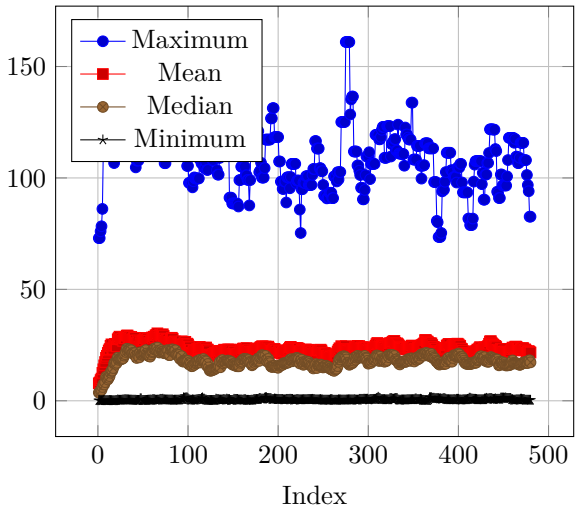
Displacement from Birthplace (Elite Fitness Table)



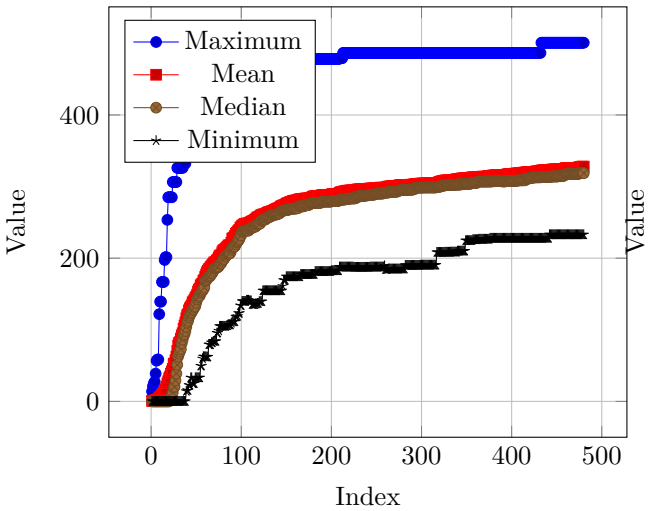
Displacement from Birthplace (Elite Novelty Table)



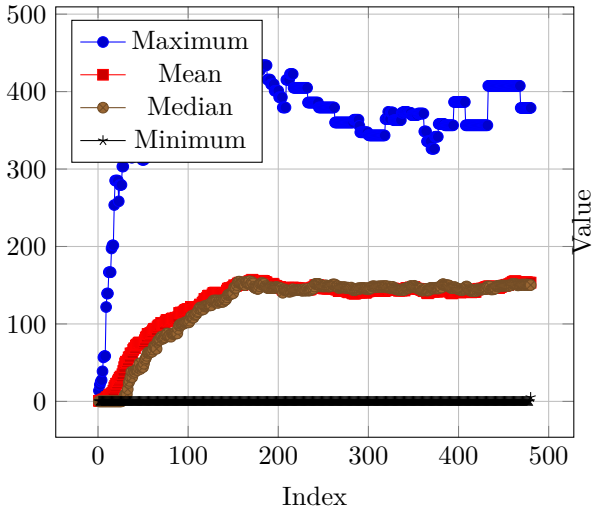
Displacement from Birthplace (Recently Dead Table)



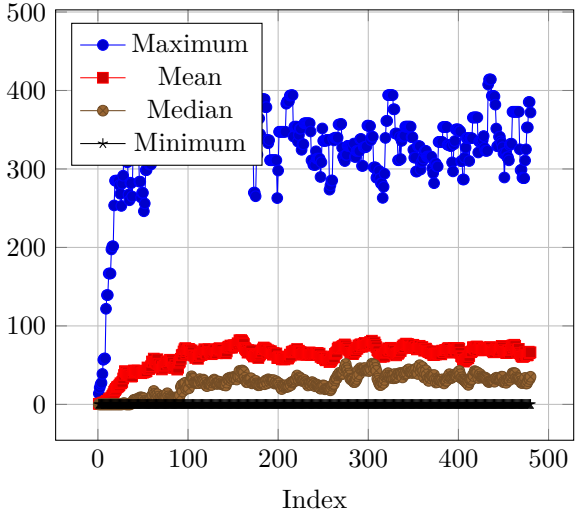
Food Eaten (Elite Fitness Table)

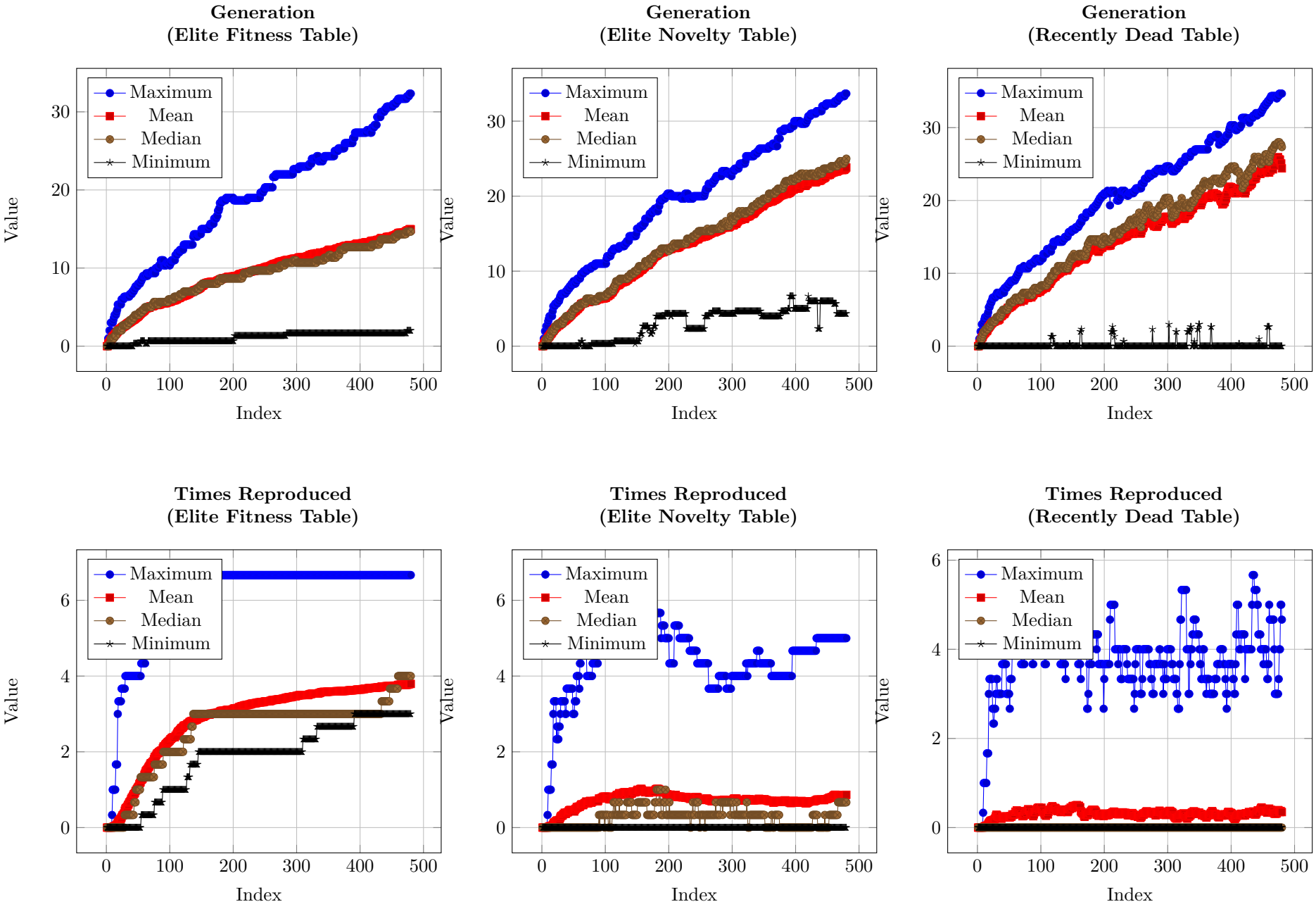


Food Eaten (Elite Novelty Table)

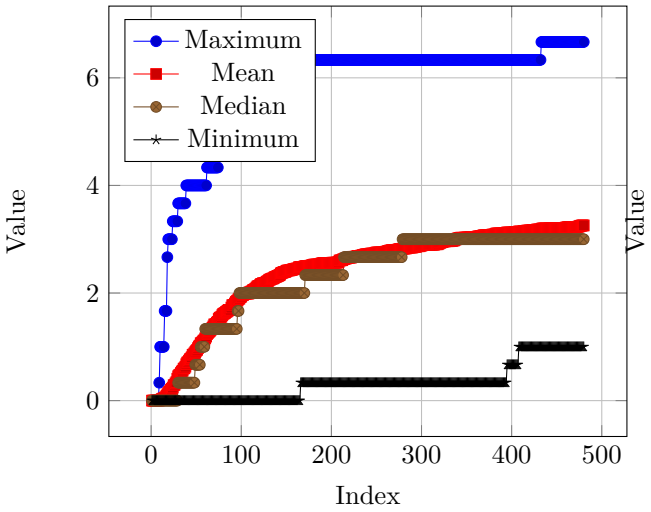


Food Eaten (Recently Dead Table)

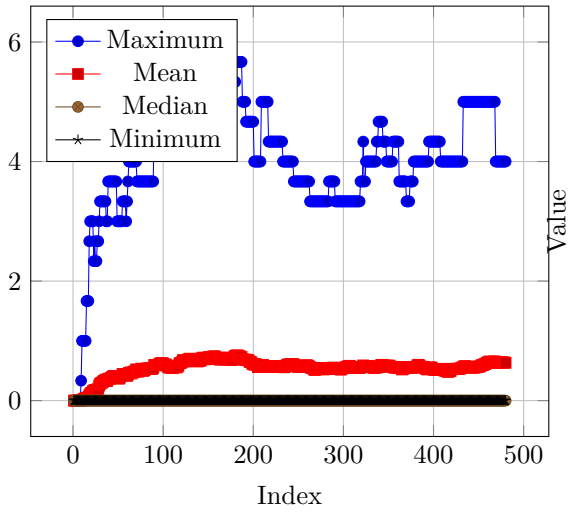



Generation
(Elite Fitness Table)
Generation
(Elite Novelty Table)
Generation
(Recently Dead Table)
Times Reproduced
(Elite Fitness Table)
Times Reproduced
(Elite Novelty Table)
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
0
10
20
30
0
2
4
6

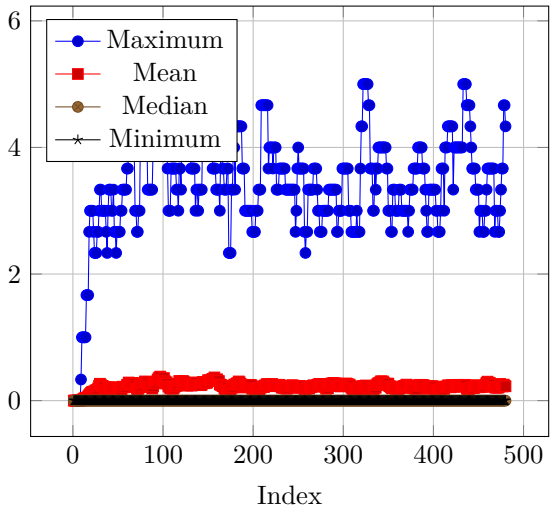
Times Reproduced Asexually (Elite Fitness Table)



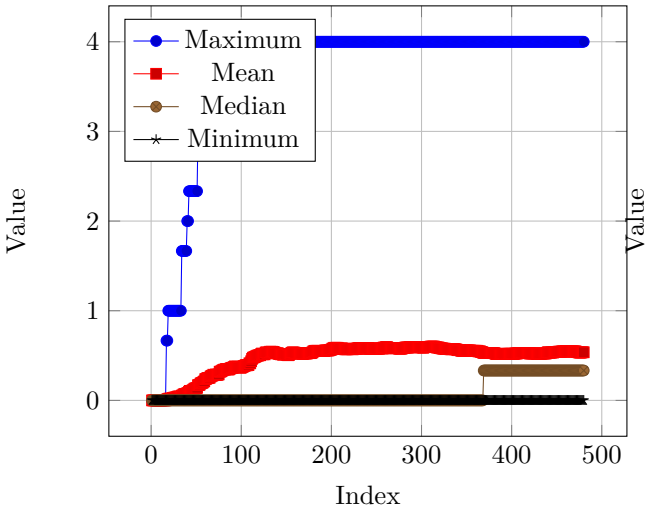
Times Reproduced Asexually (Elite Novelty Table)



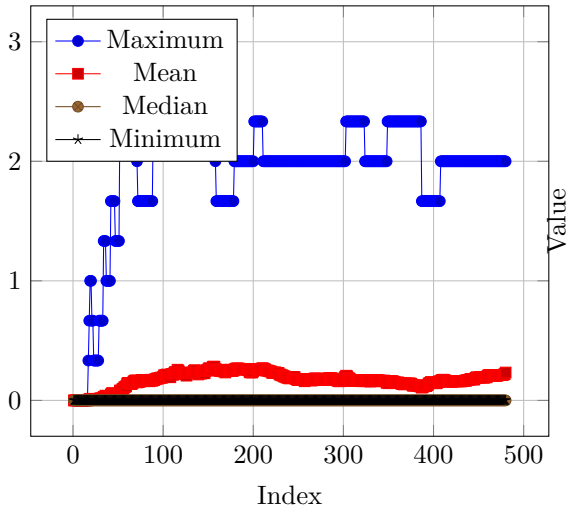
Times Reproduced Asexually (Recently Dead Table)



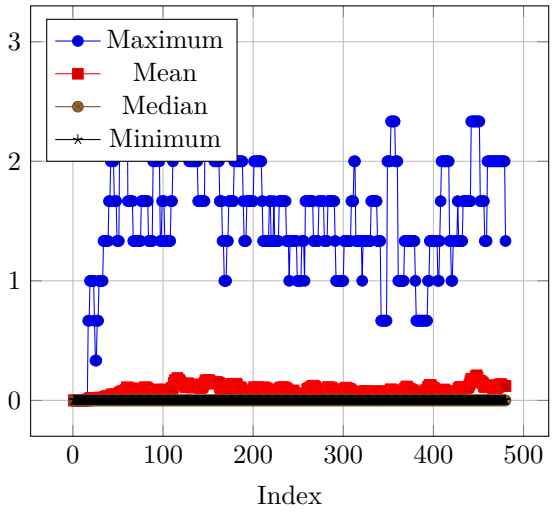
Times Reproduced Sexually (Elite Fitness Table)

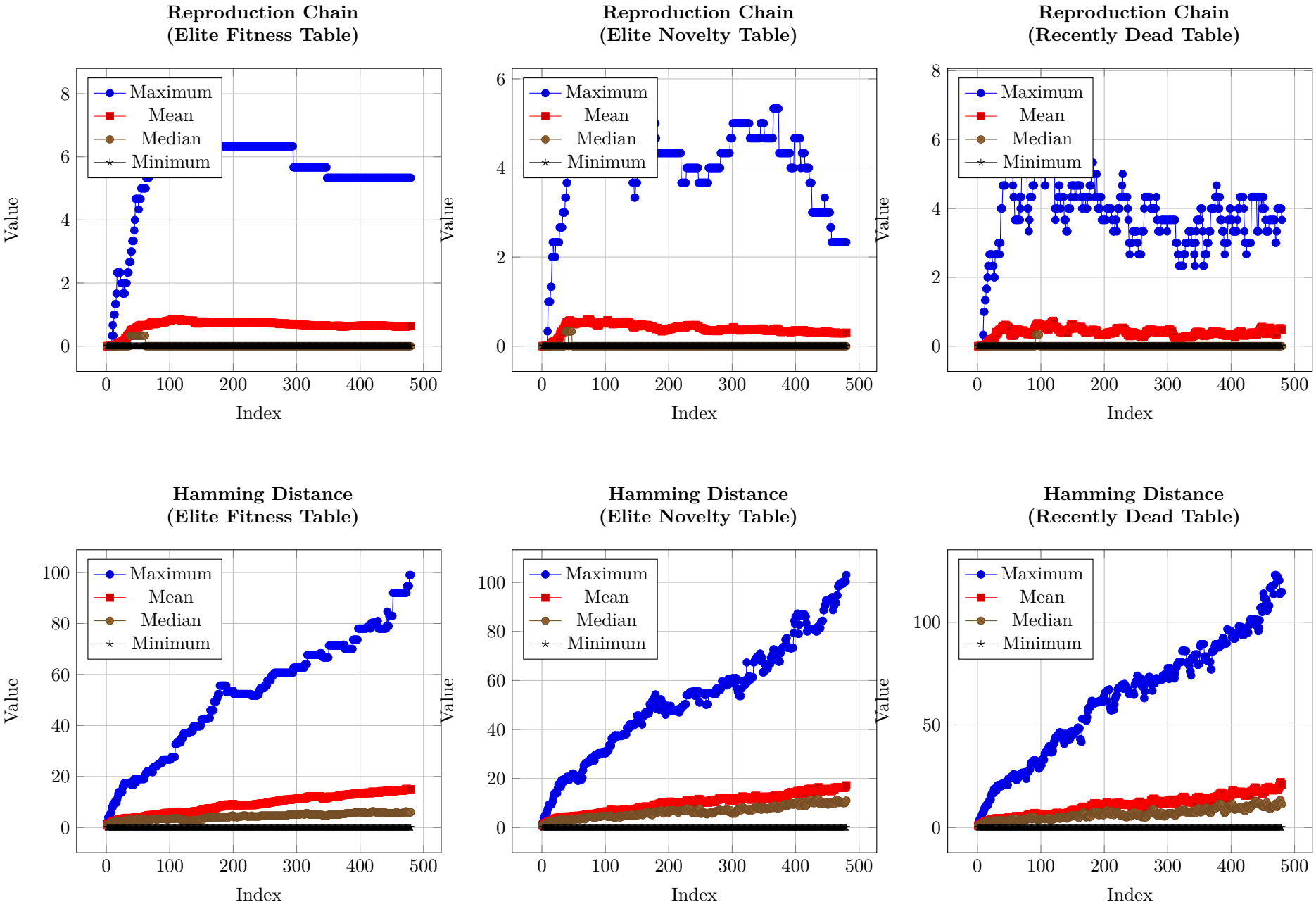


Times Reproduced Sexually (Elite Novelty Table)

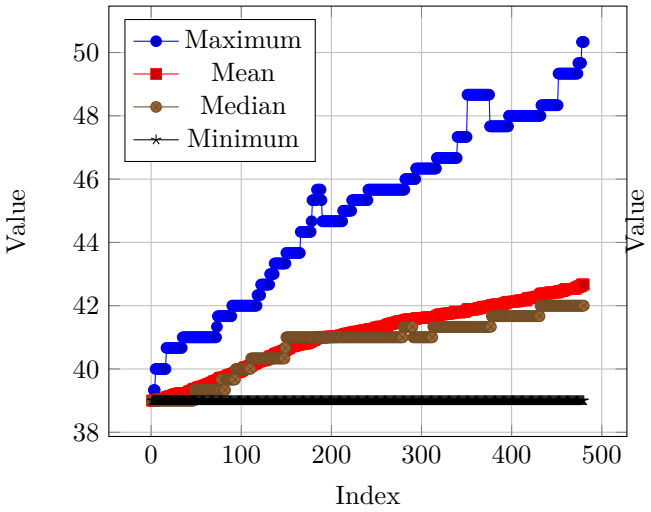


Times Reproduced Sexually (Recently Dead Table)

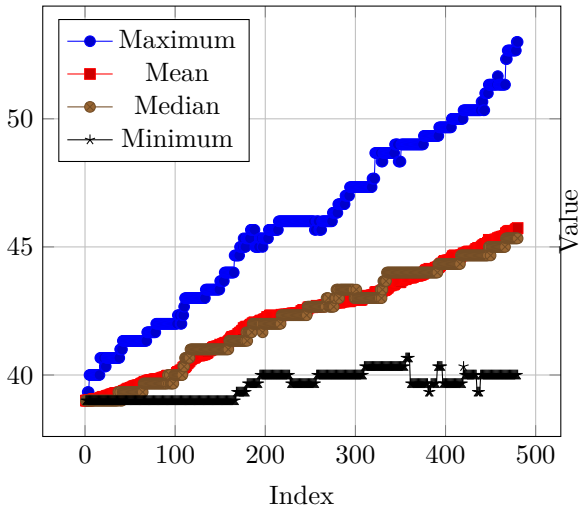



Reproduction Chain
(Elite Fitness Table)
Reproduction Chain
(Elite Novelty Table)
Reproduction Chain
(Recently Dead Table)
Hamming Distance
(Elite Fitness Table)
Hamming Distance
(Elite Novelty Table)
Hamming Distance
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index

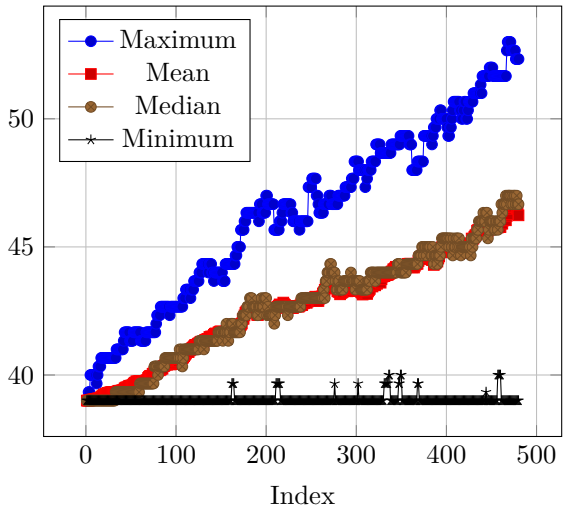
Number of Neurons (Elite Fitness Table)



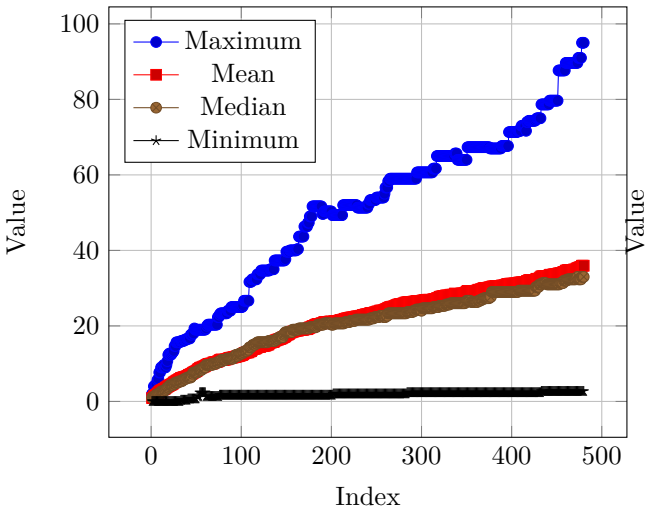
Number of Neurons (Elite Novelty Table)



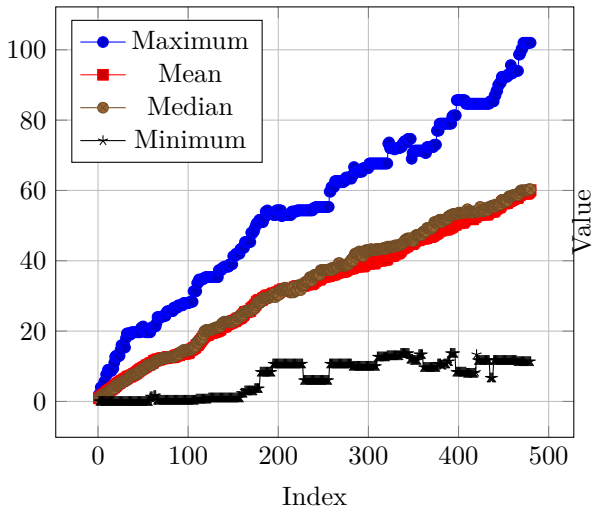
Number of Neurons (Recently Dead Table)



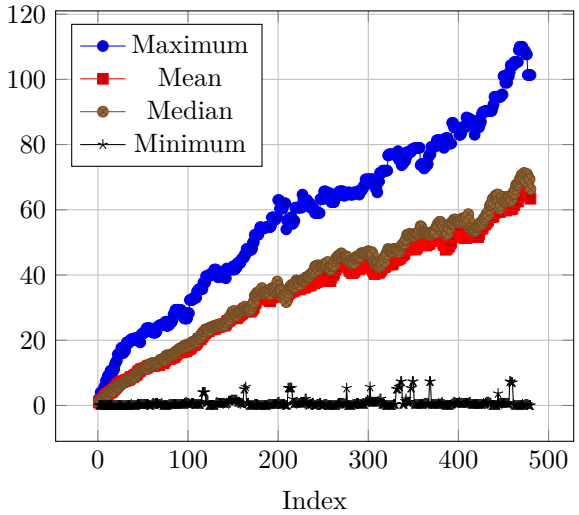
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)



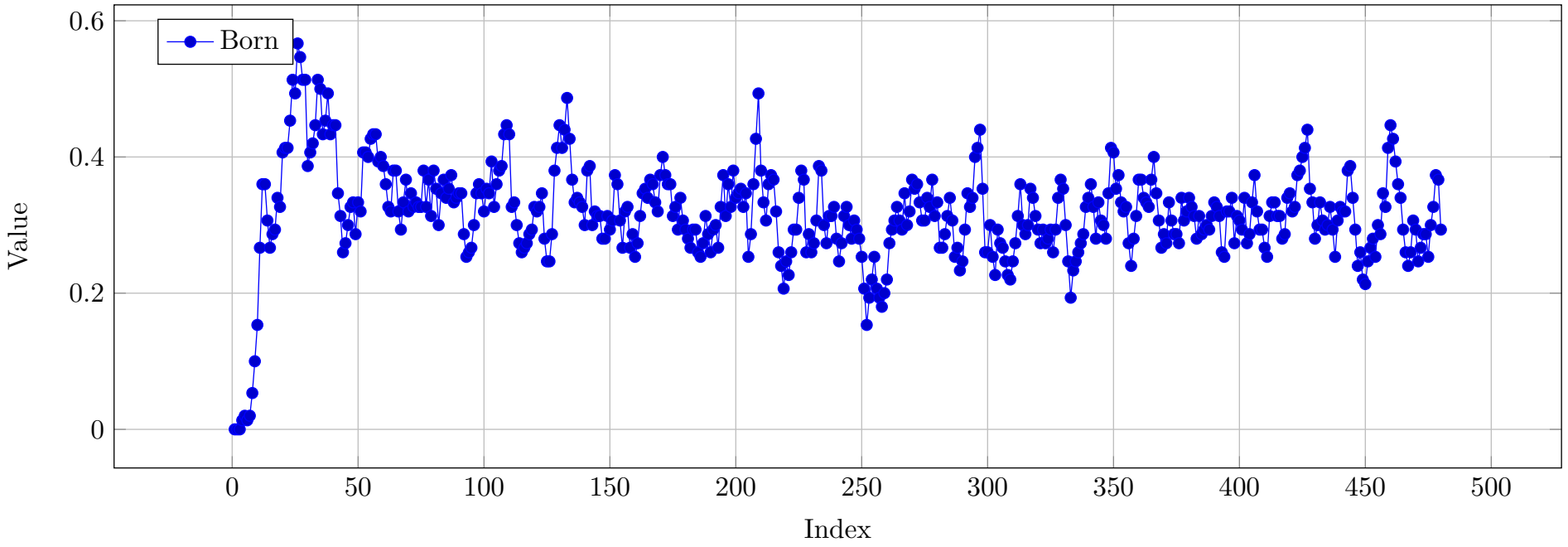
Number of Connections (Recently Dead Table)



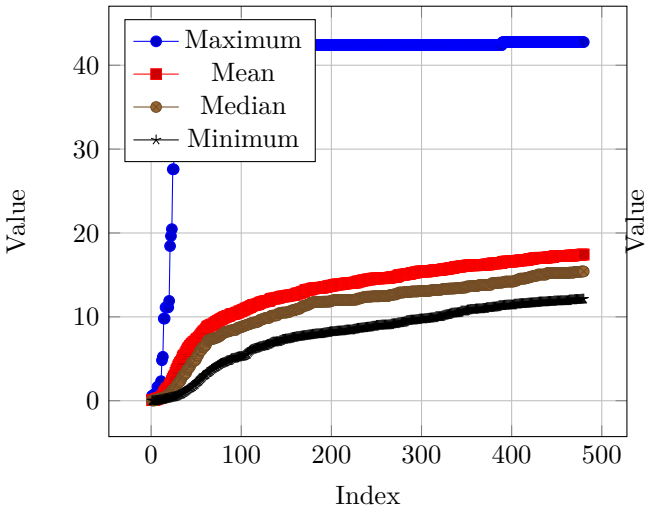
Wheeled Robot - Sum & Squash - Hebb ABCD - Flat World

RESULTS

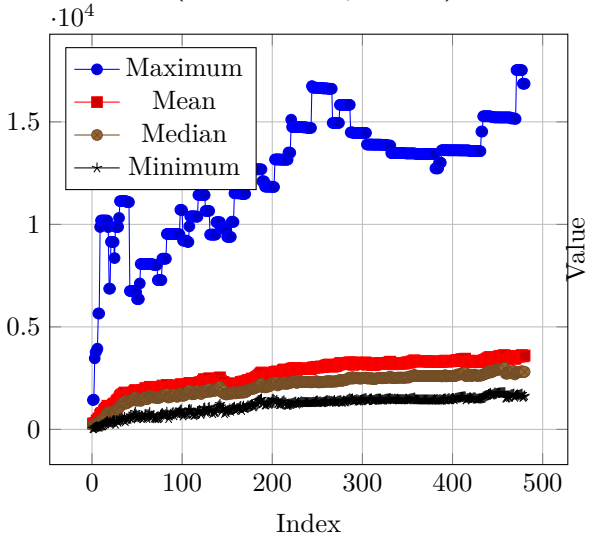
Percent of Minimum Population that was Born



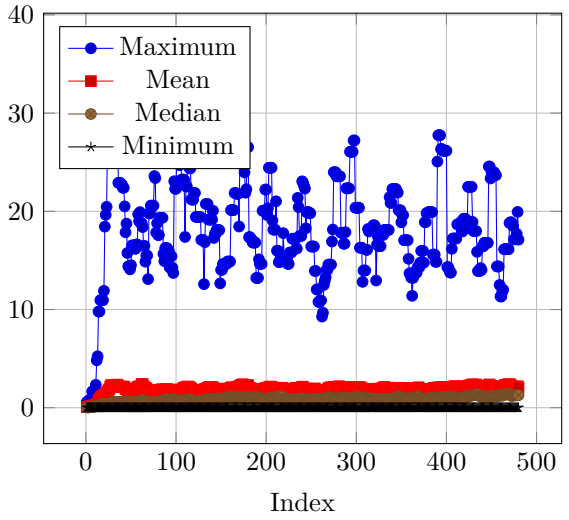
Fitness Score
(Elite Fitness Table)



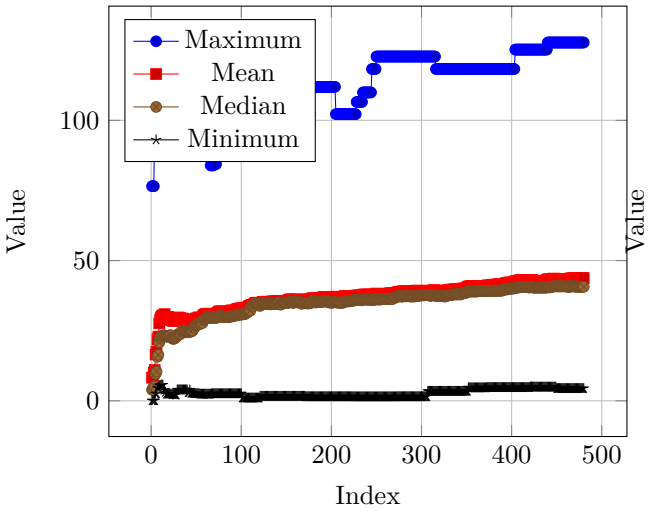
Novelty Score
(Elite Novelty Table)



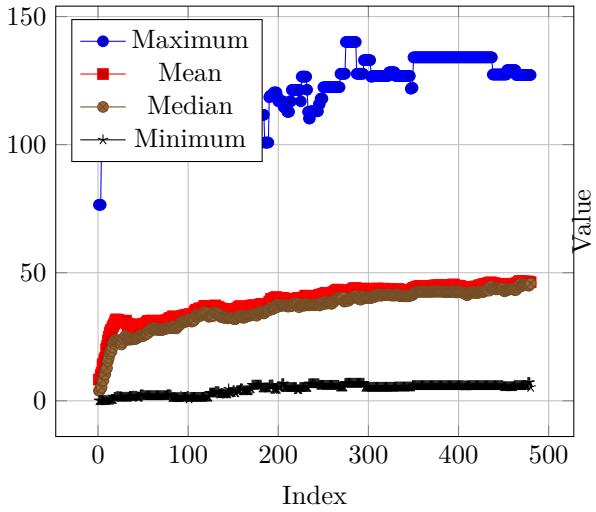
Fitness Score
(Recently Dead Table)



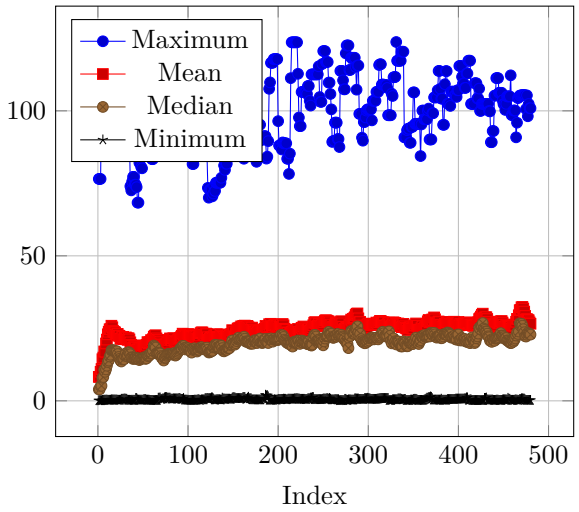
Displacement from Birthplace (Elite Fitness Table)



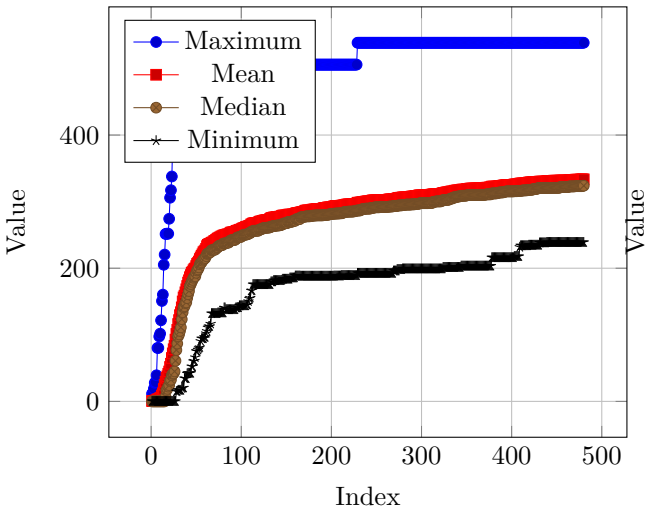
Displacement from Birthplace (Elite Novelty Table)



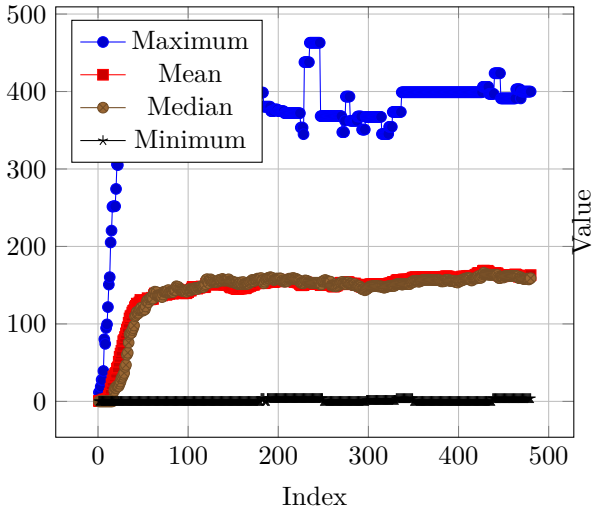
Displacement from Birthplace (Recently Dead Table)



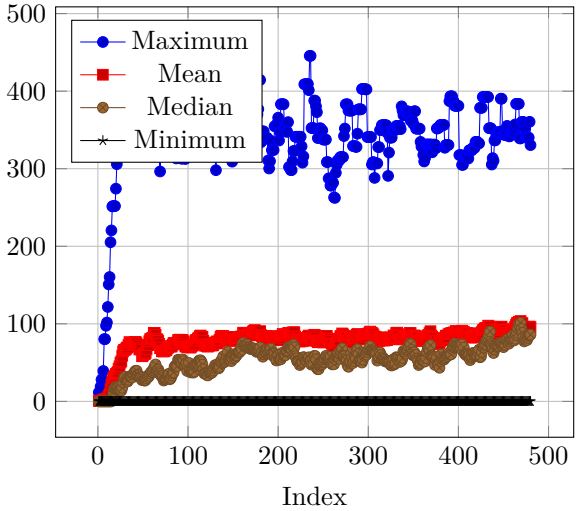
Food Eaten (Elite Fitness Table)

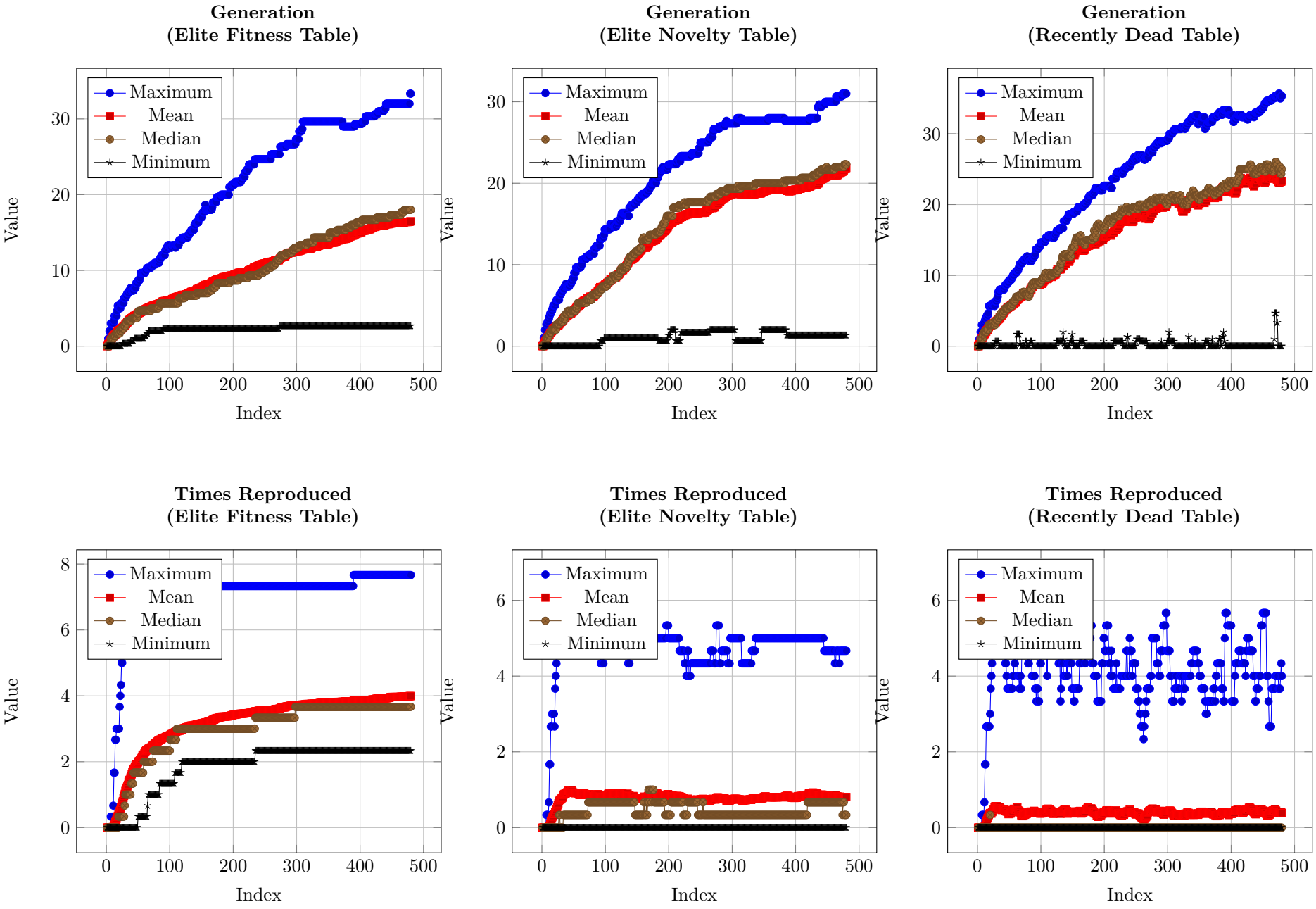


Food Eaten (Elite Novelty Table)

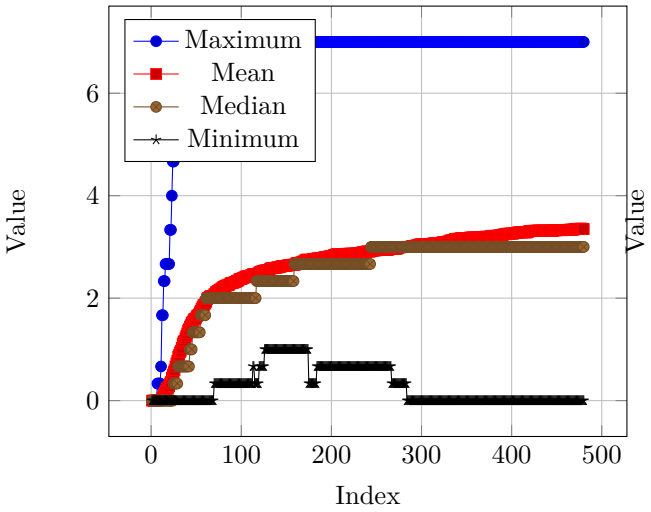


Food Eaten (Recently Dead Table)

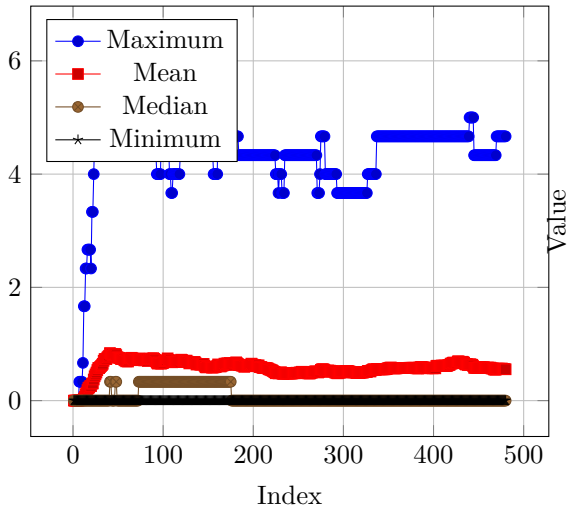



Generation
(Elite Fitness Table)
Generation
(Elite Novelty Table)
Generation
(Recently Dead Table)
Times Reproduced
(Elite Fitness Table)
Times Reproduced
(Elite Novelty Table)
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
0
10
20
30
0
2
4
6
8

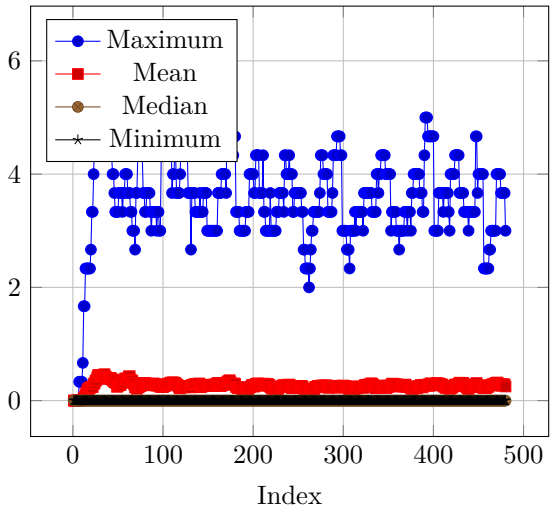
Times Reproduced Asexually (Elite Fitness Table)



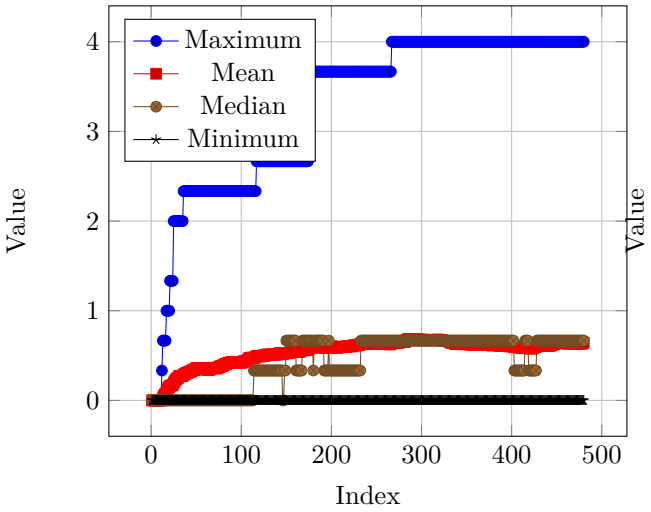
Times Reproduced Asexually (Elite Novelty Table)



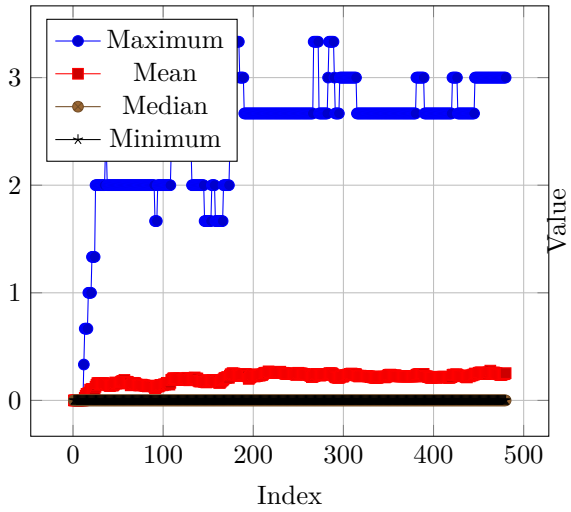
Times Reproduced Asexually (Recently Dead Table)



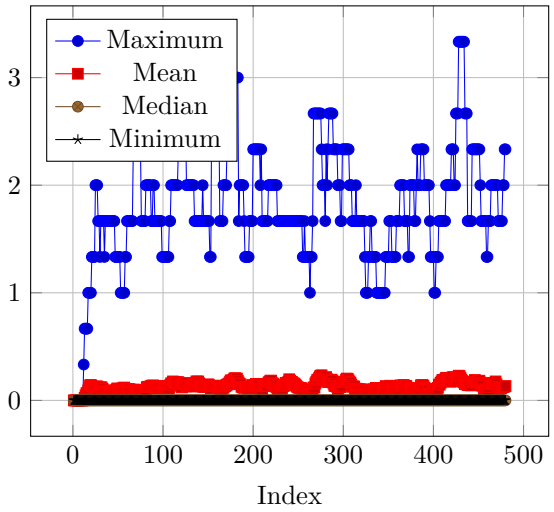
Times Reproduced Sexually (Elite Fitness Table)



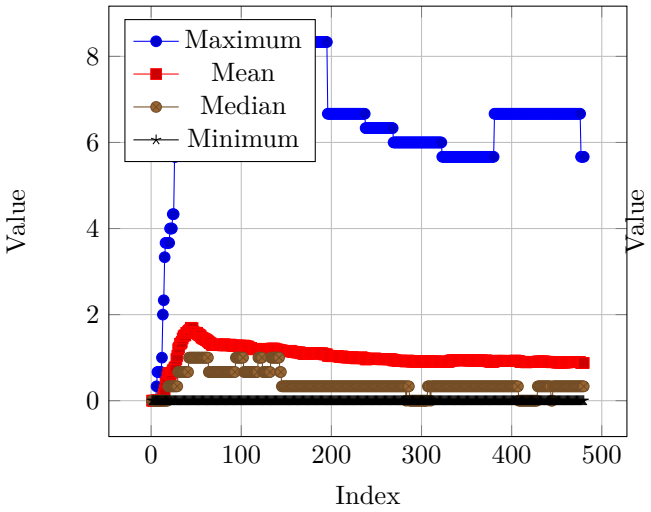
Times Reproduced Sexually (Elite Novelty Table)



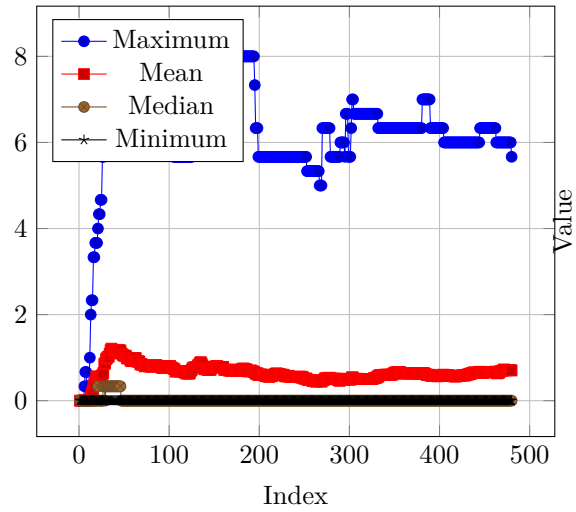
Times Reproduced Sexually (Recently Dead Table)



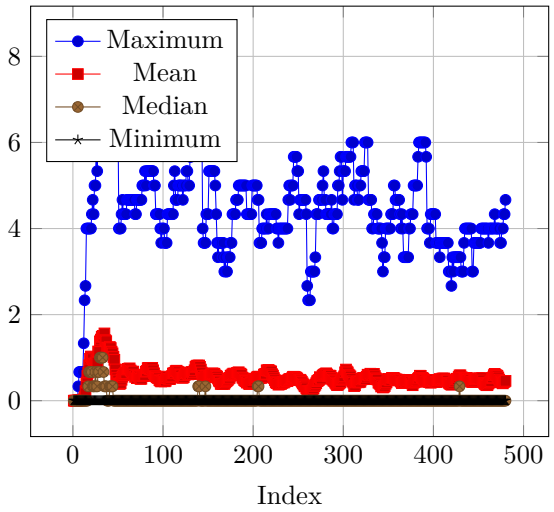
Reproduction Chain (Elite Fitness Table)



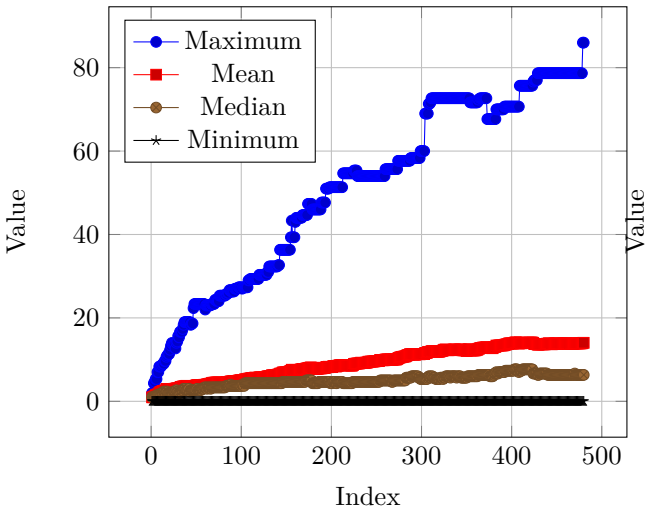
Reproduction Chain (Elite Novelty Table)



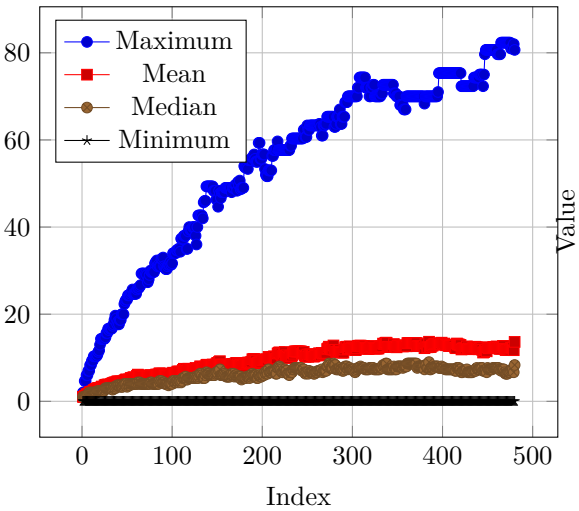
Reproduction Chain (Recently Dead Table)



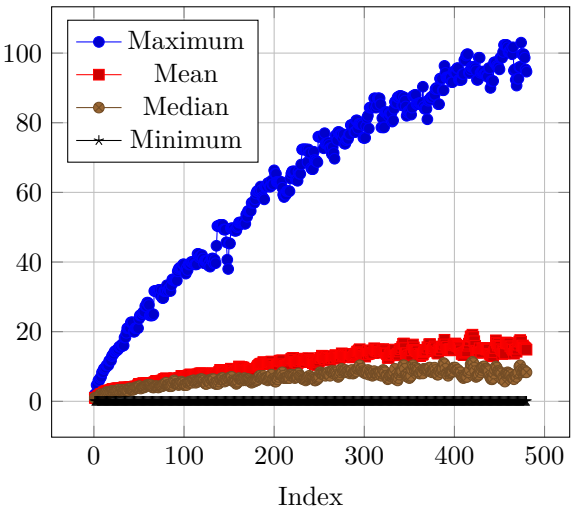
Hamming Distance (Elite Fitness Table)



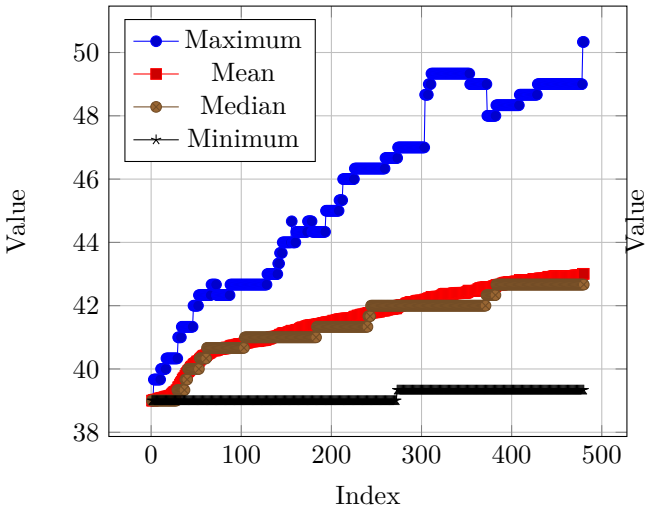
Hamming Distance (Elite Novelty Table)



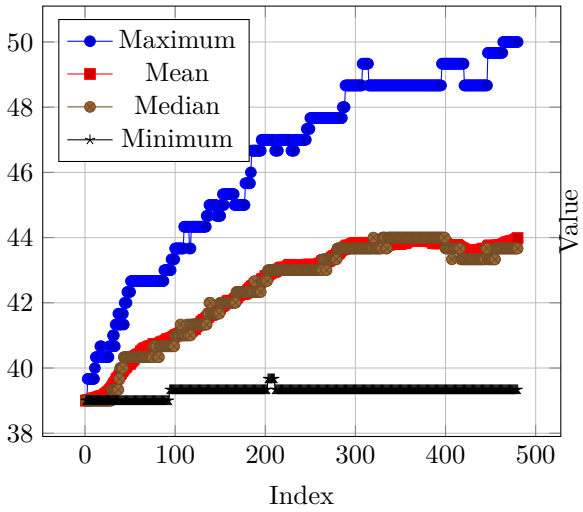
Hamming Distance (Recently Dead Table)



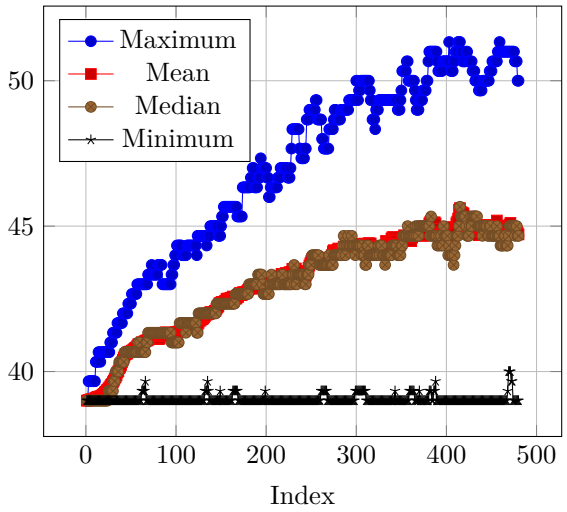
Number of Neurons (Elite Fitness Table)



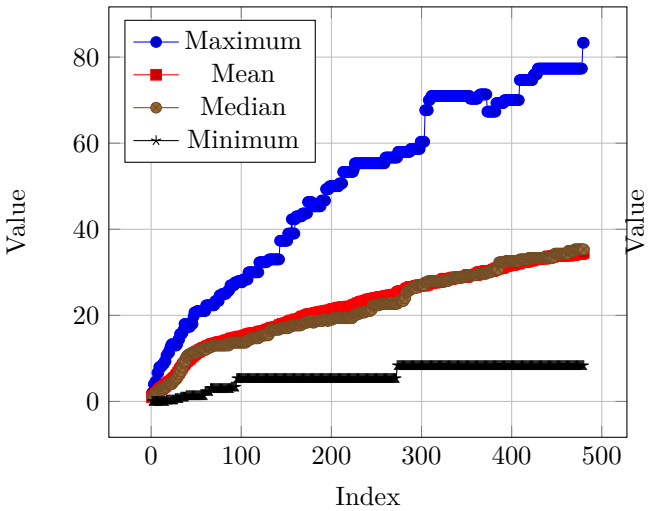
Number of Neurons (Elite Novelty Table)



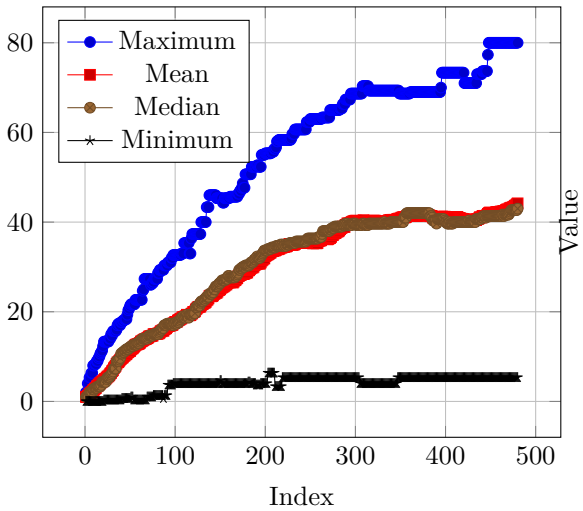
Number of Neurons (Recently Dead Table)



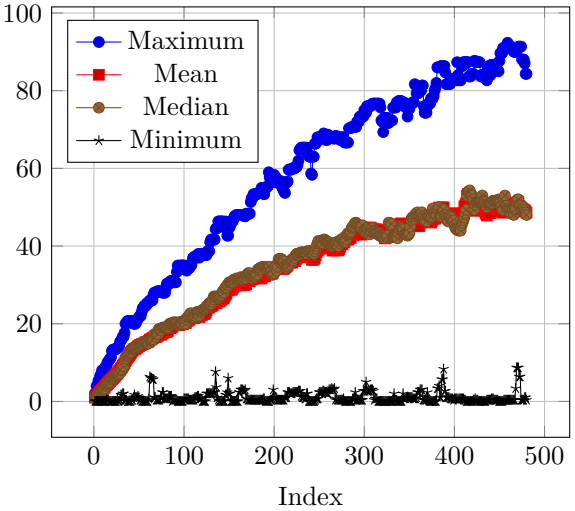
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)



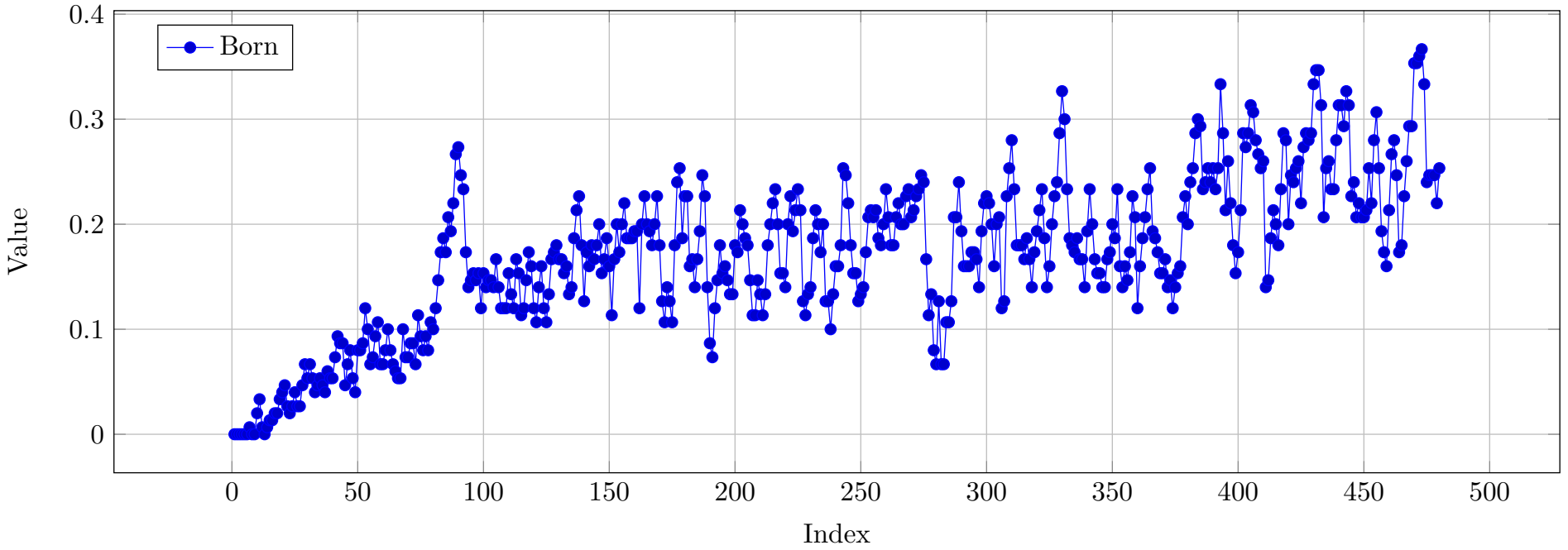
Number of Connections (Recently Dead Table)



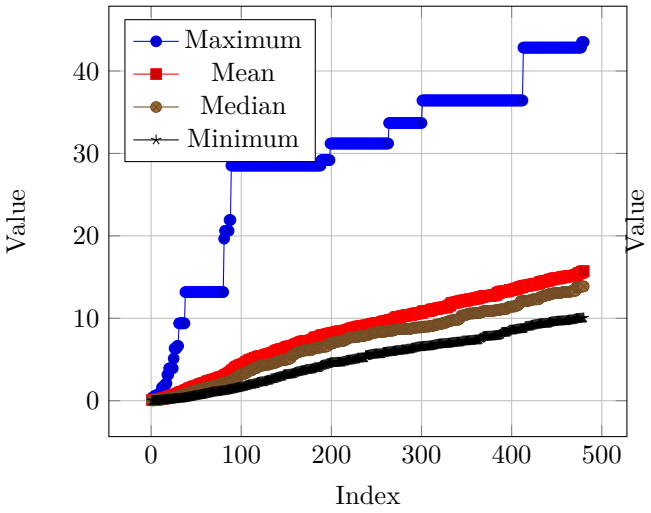
Wheeled Robot - CTRNN - No Learning - Flat World

RESULTS

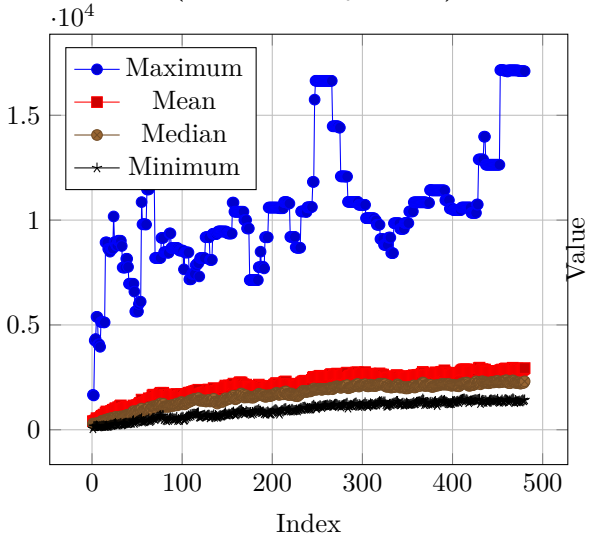
Percent of Minimum Population that was Born



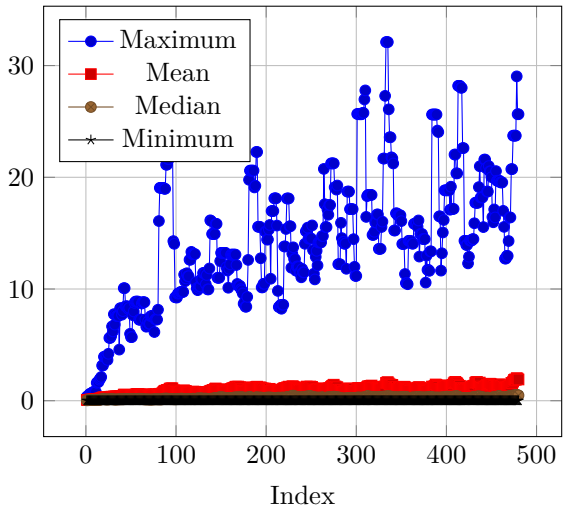
Fitness Score
(Elite Fitness Table)



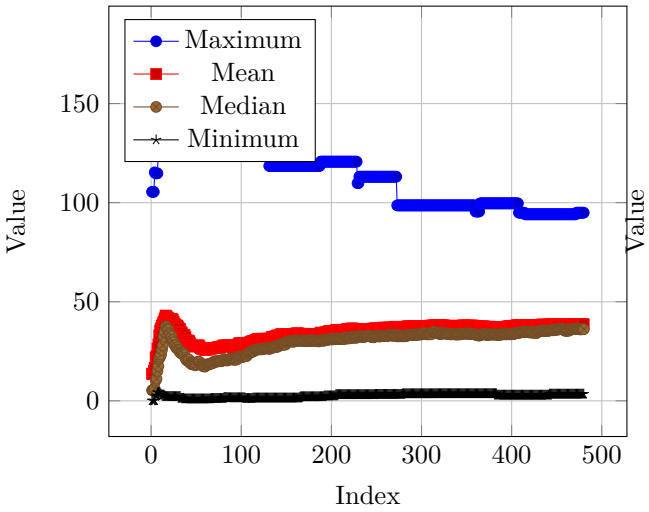
Novelty Score
(Elite Novelty Table)



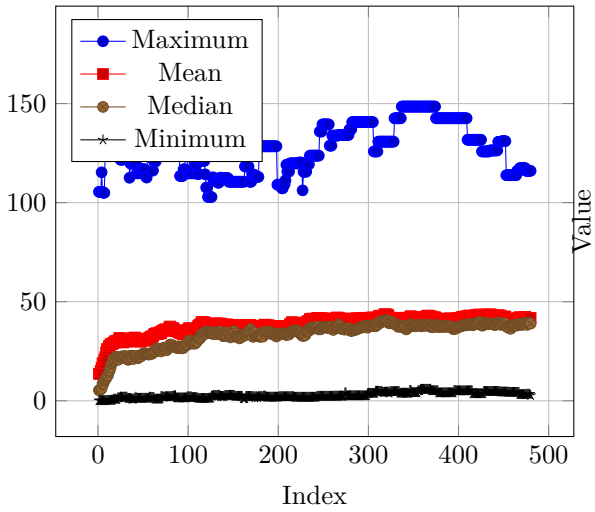
Fitness Score
(Recently Dead Table)



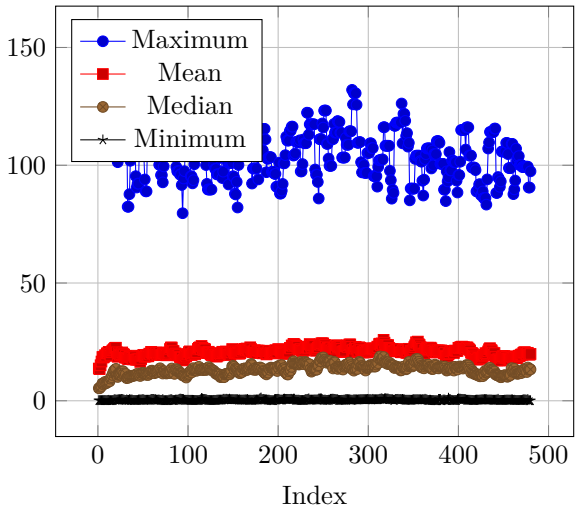
Displacement from Birthplace (Elite Fitness Table)



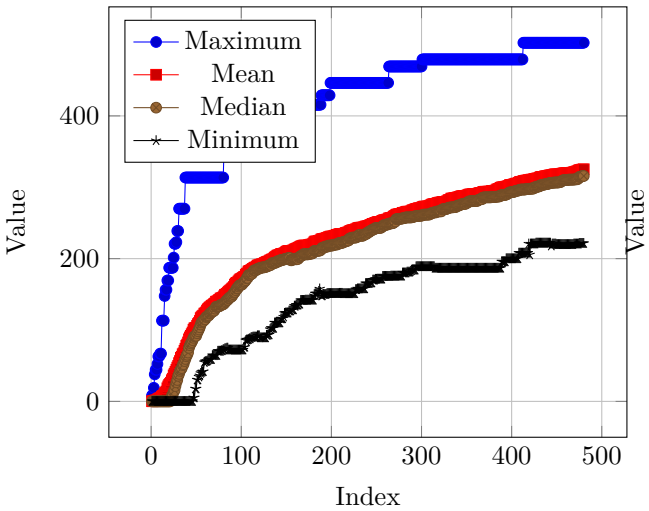
Displacement from Birthplace (Elite Novelty Table)



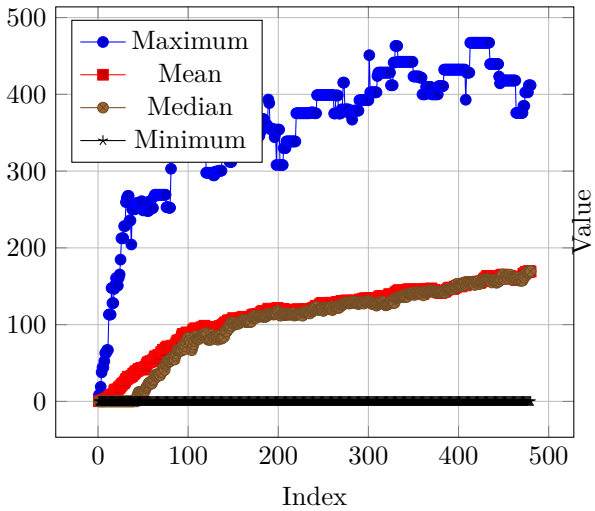
Displacement from Birthplace (Recently Dead Table)



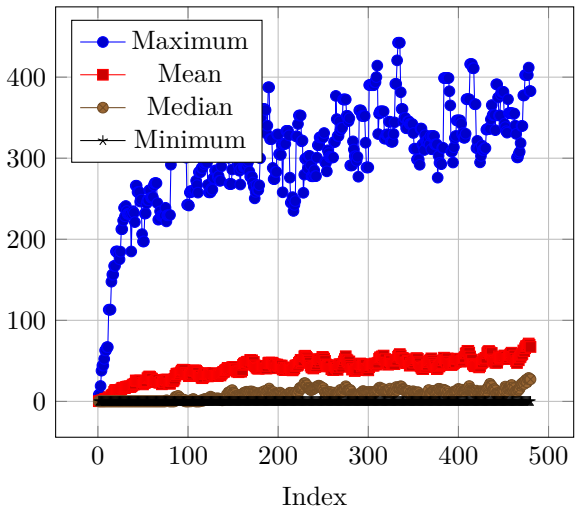
Food Eaten (Elite Fitness Table)

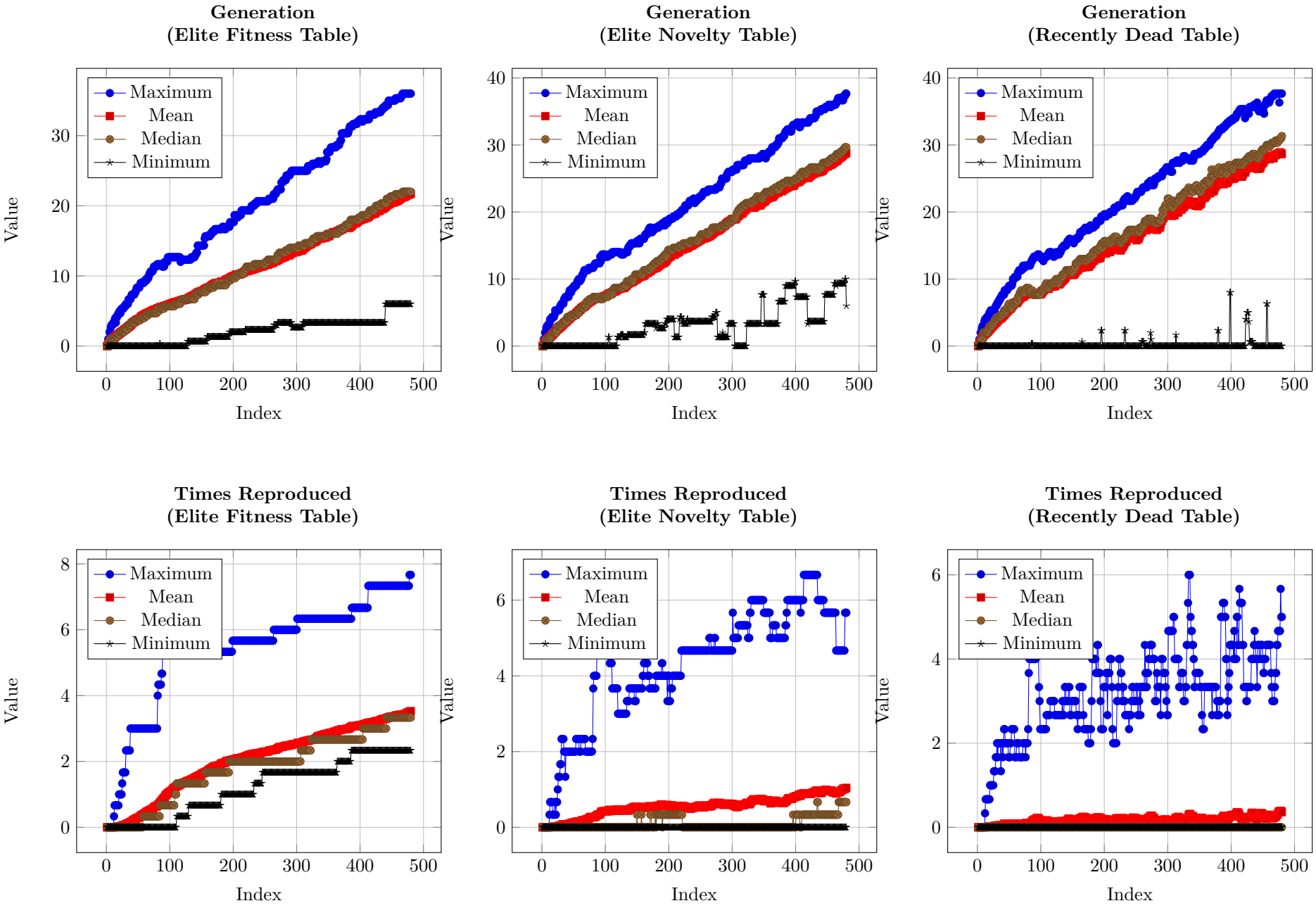


Food Eaten (Elite Novelty Table)

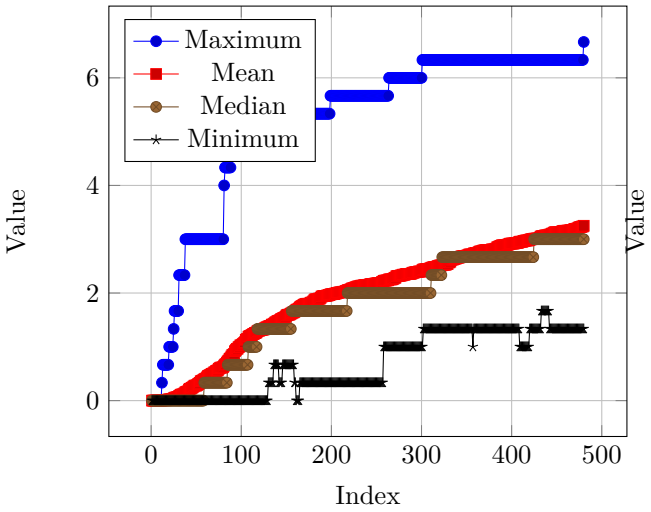


Food Eaten (Recently Dead Table)

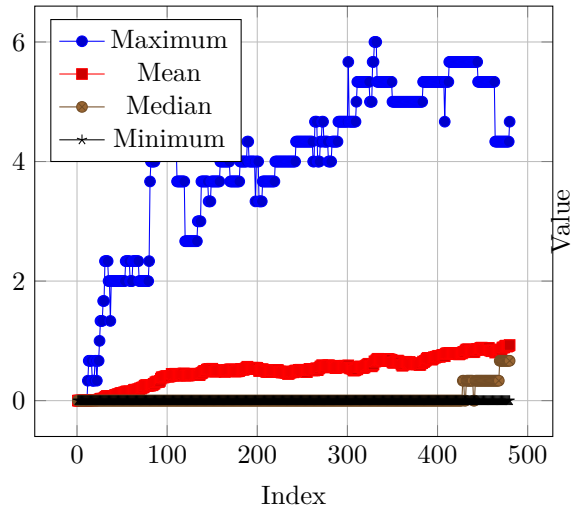



Generation
(Elite Fitness Table)
Generation
(Elite Novelty Table)
Generation
(Recently Dead Table)
Times Reproduced
(Elite Fitness Table)
Times Reproduced
(Elite Novelty Table)
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
0
10
20
30
40
2
4
6
8

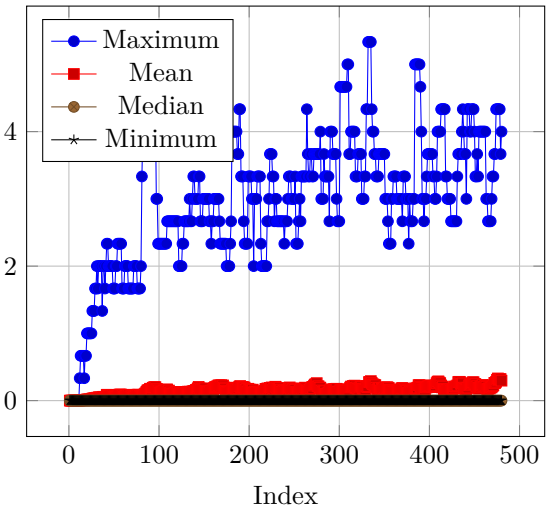
Times Reproduced Asexually (Elite Fitness Table)



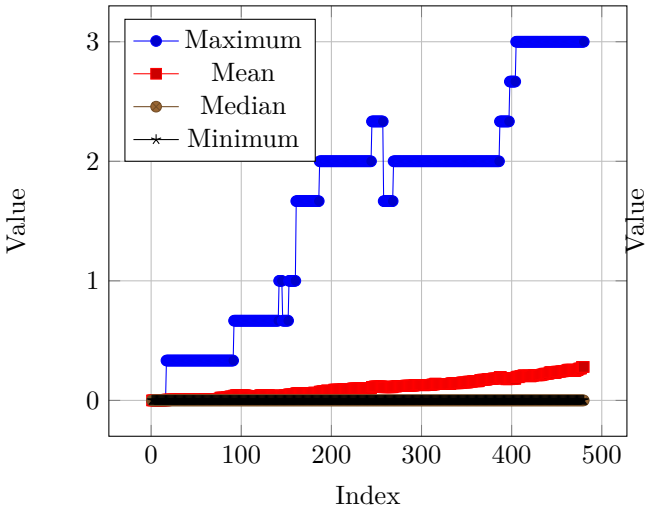
Times Reproduced Asexually (Elite Novelty Table)



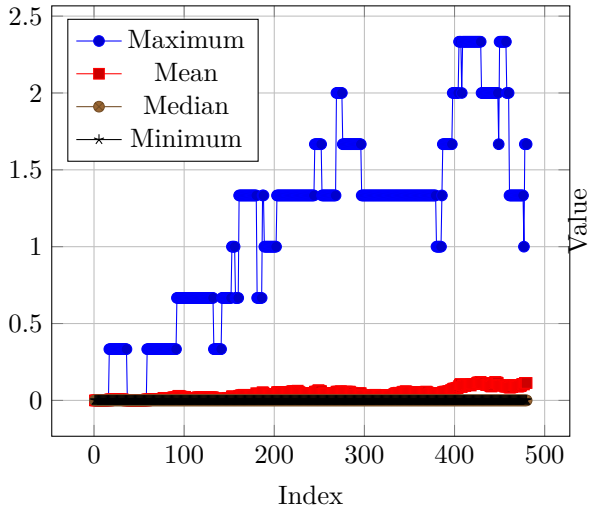
Times Reproduced Asexually (Recently Dead Table)



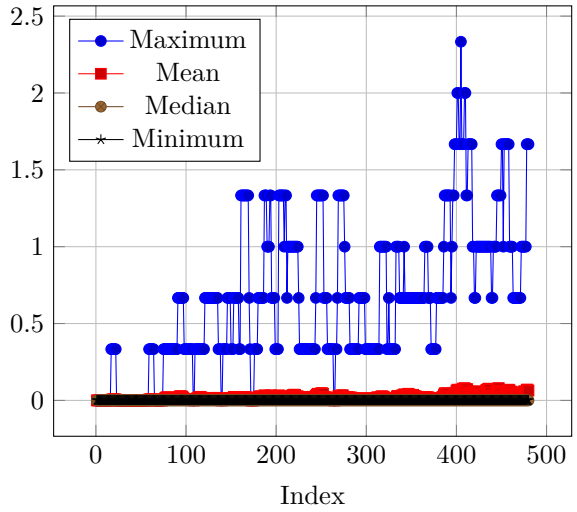
Times Reproduced Sexually (Elite Fitness Table)



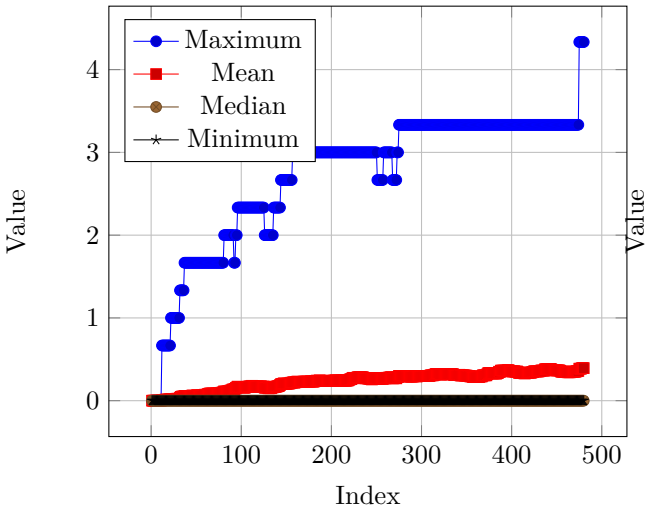
Times Reproduced Sexually (Elite Novelty Table)



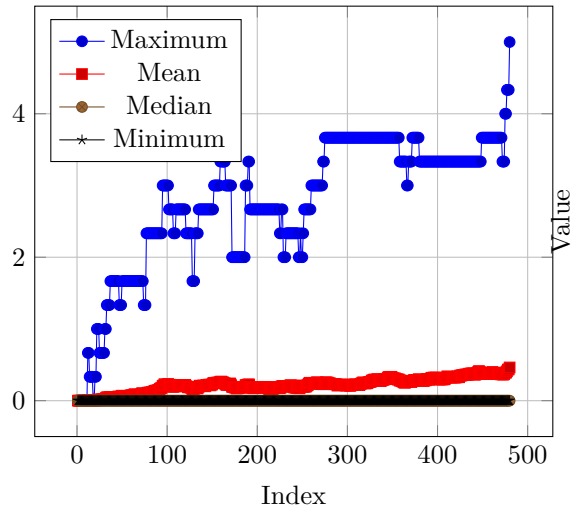
Times Reproduced Sexually (Recently Dead Table)



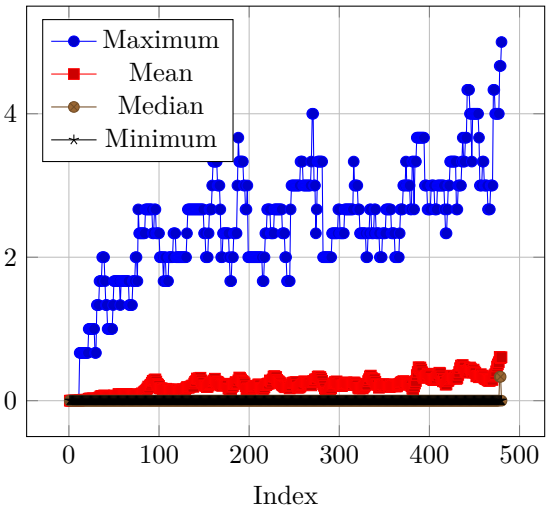
Reproduction Chain (Elite Fitness Table)



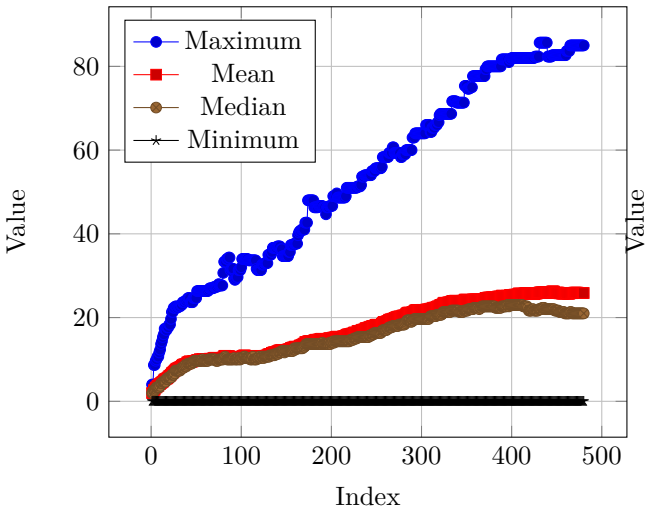
Reproduction Chain (Elite Novelty Table)



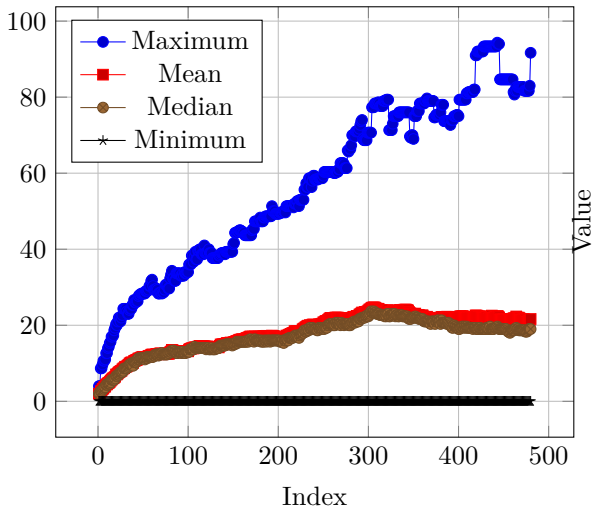
Reproduction Chain (Recently Dead Table)



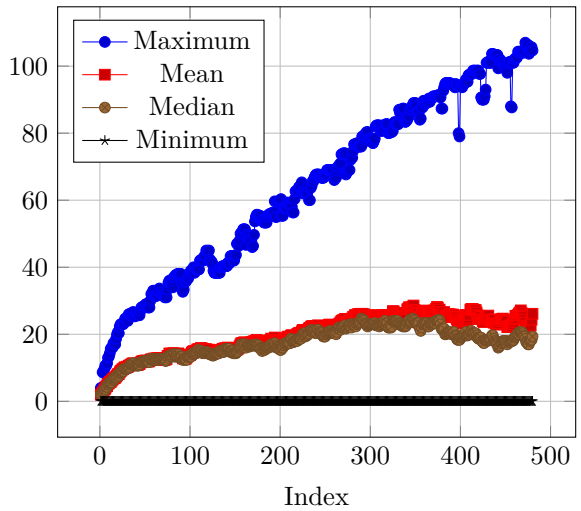
Hamming Distance (Elite Fitness Table)



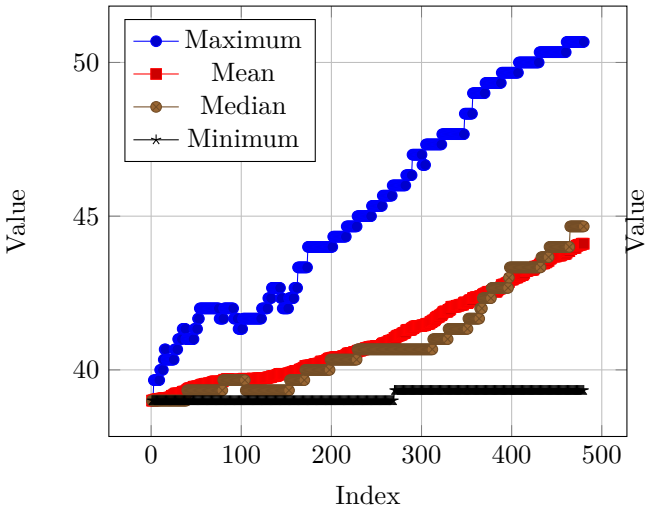
Hamming Distance (Elite Novelty Table)



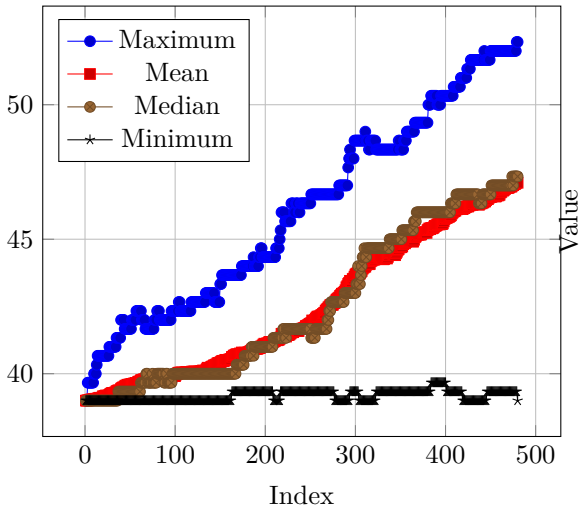
Hamming Distance (Recently Dead Table)



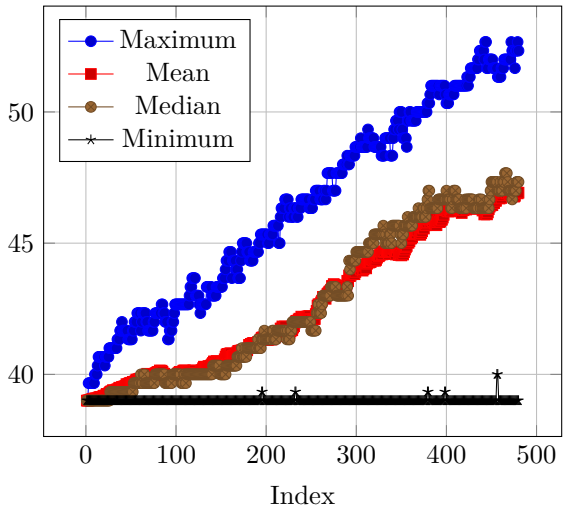
Number of Neurons (Elite Fitness Table)



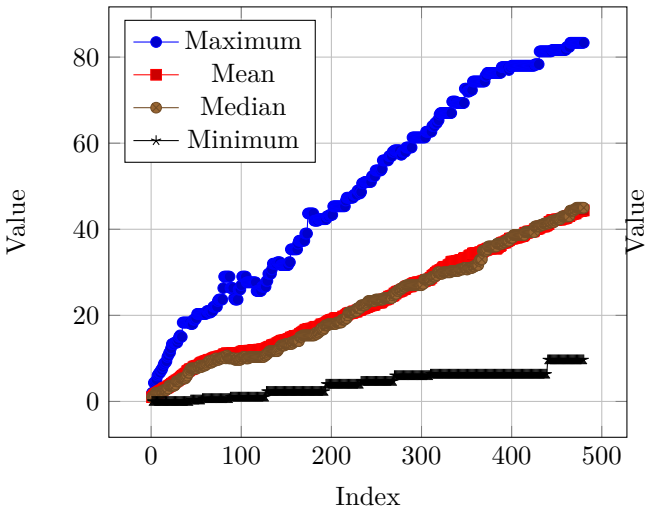
Number of Neurons (Elite Novelty Table)



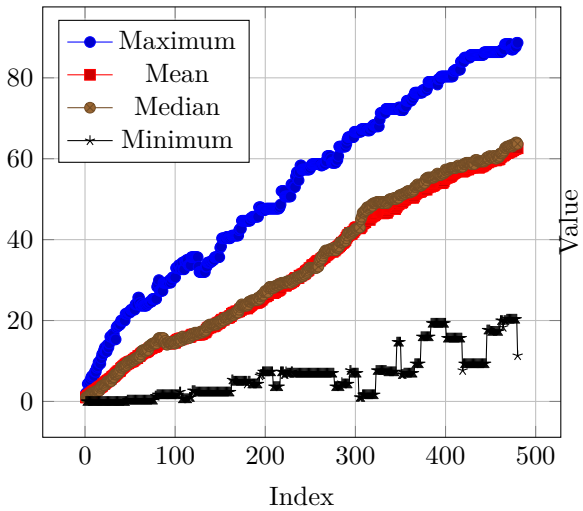
Number of Neurons (Recently Dead Table)



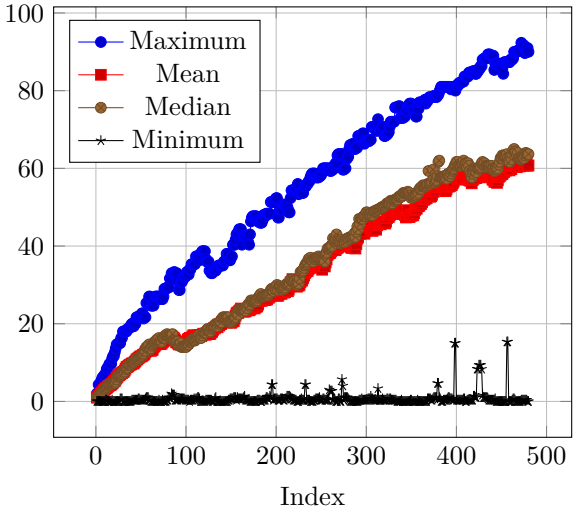
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)



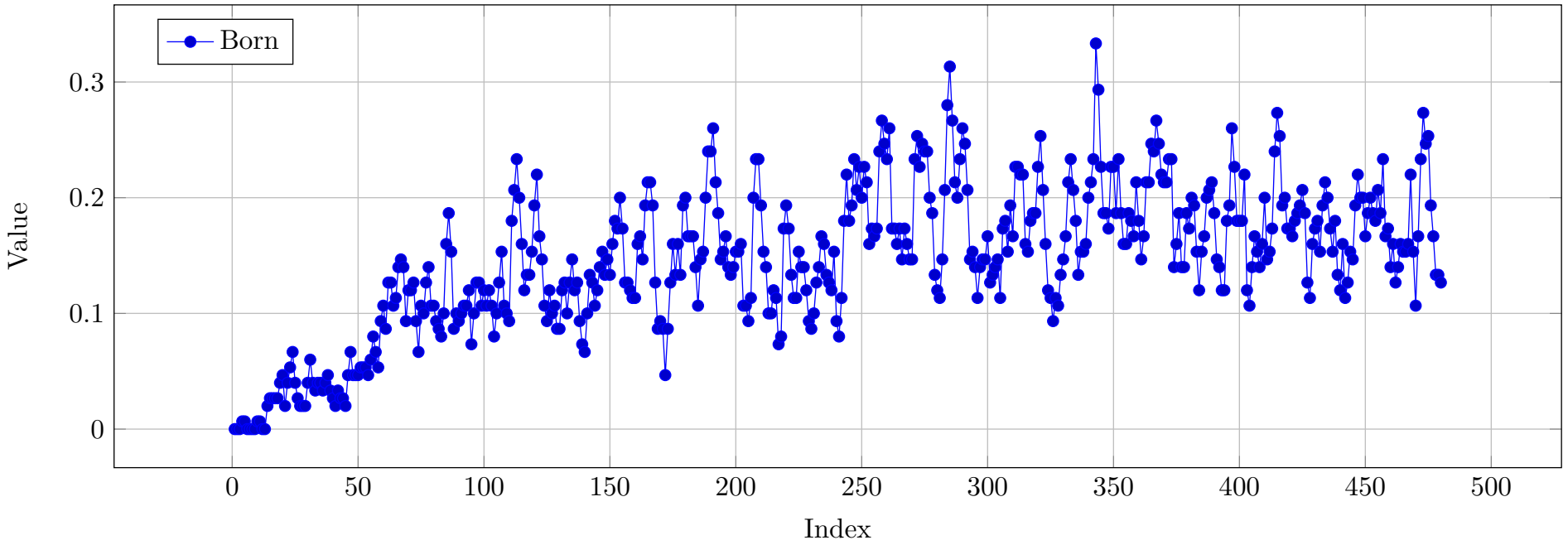
Number of Connections (Recently Dead Table)



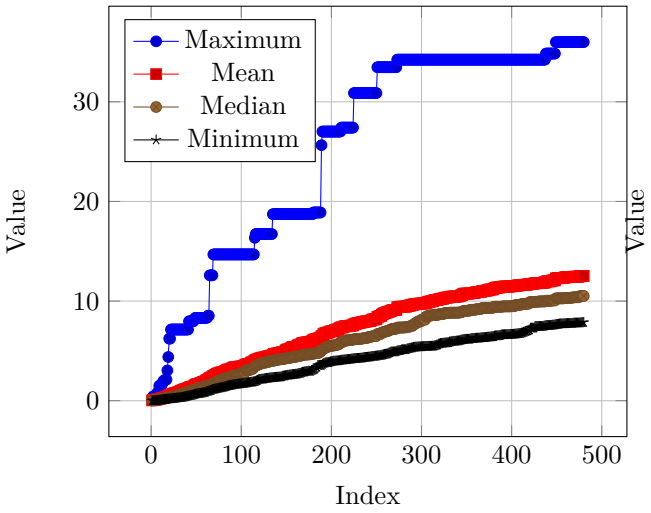
Wheeled Robot - CTRNN - Hebb ABCD - Flat World

RESULTS

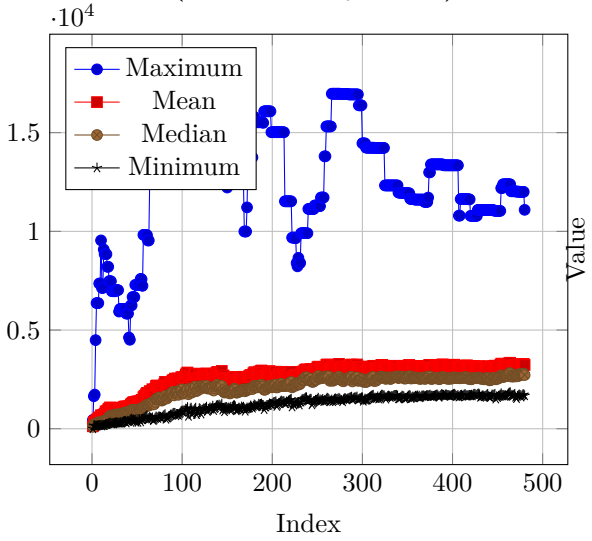
Percent of Minimum Population that was Born



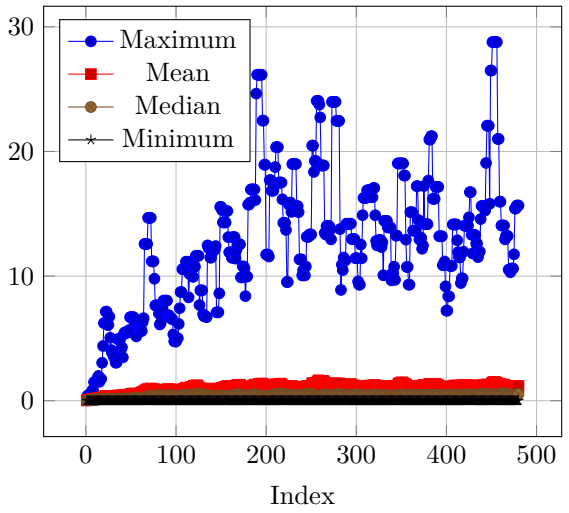
Fitness Score
(Elite Fitness Table)



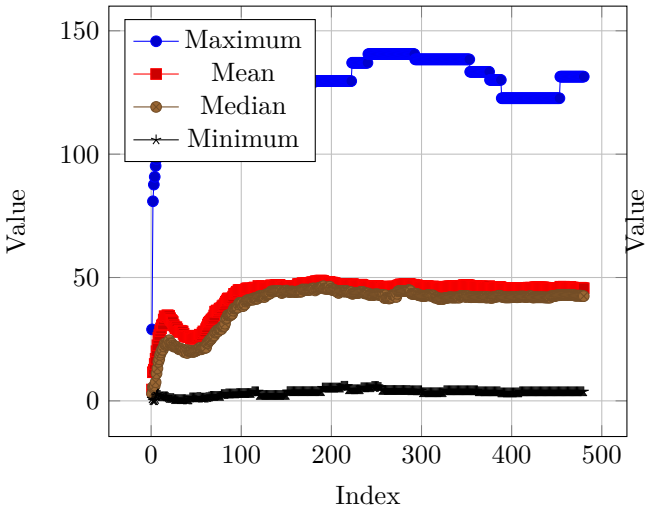
Novelty Score
(Elite Novelty Table)



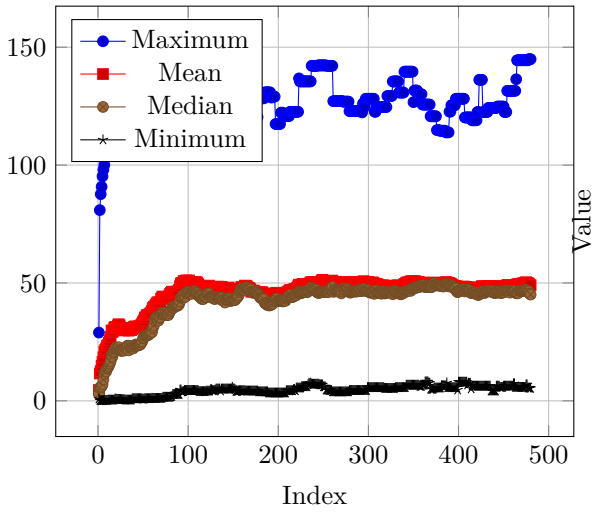
Fitness Score
(Recently Dead Table)



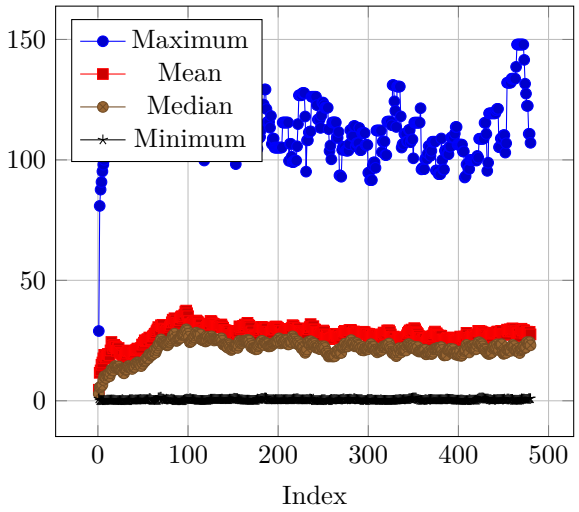
Displacement from Birthplace (Elite Fitness Table)



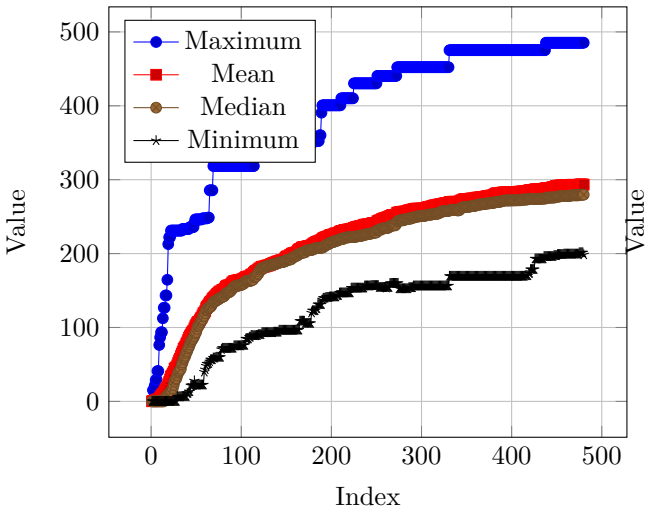
Displacement from Birthplace (Elite Novelty Table)



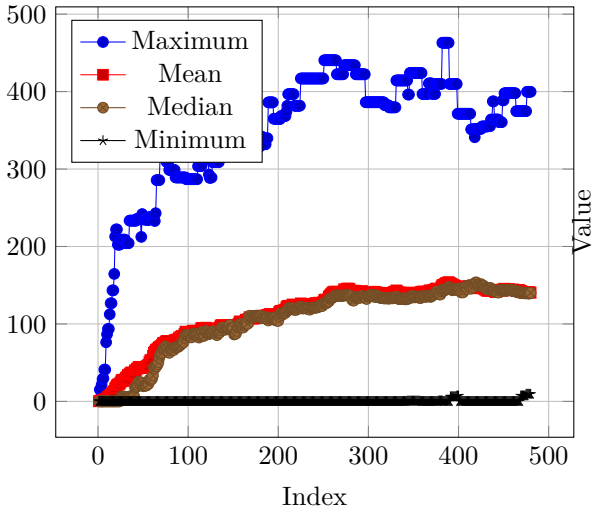
Displacement from Birthplace (Recently Dead Table)



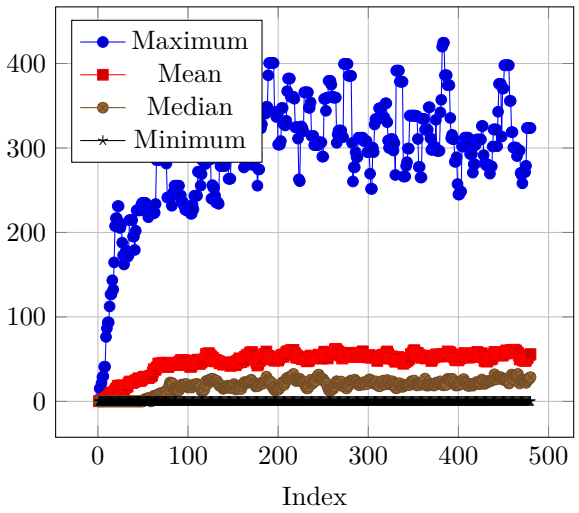
Food Eaten (Elite Fitness Table)

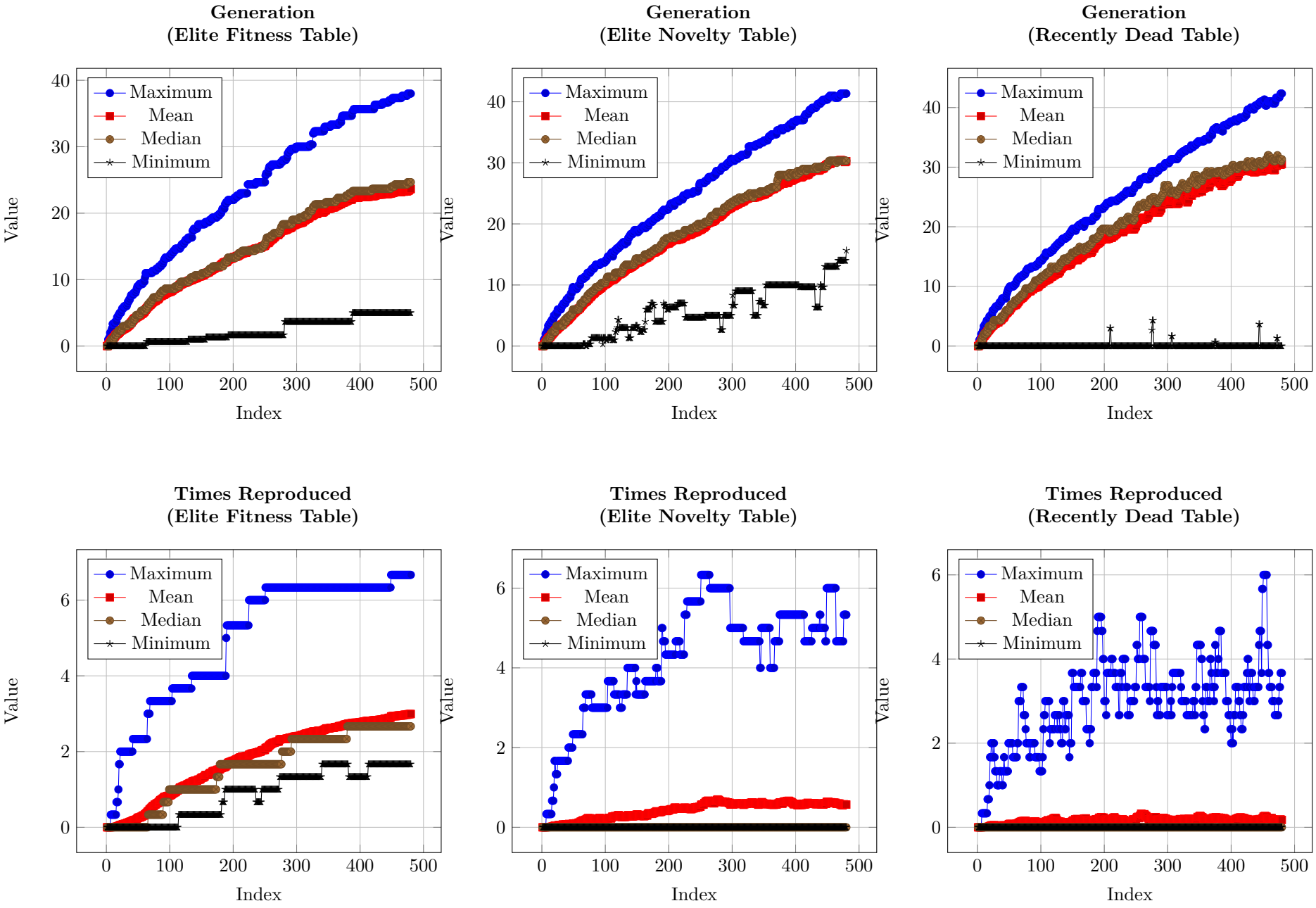


Food Eaten (Elite Novelty Table)

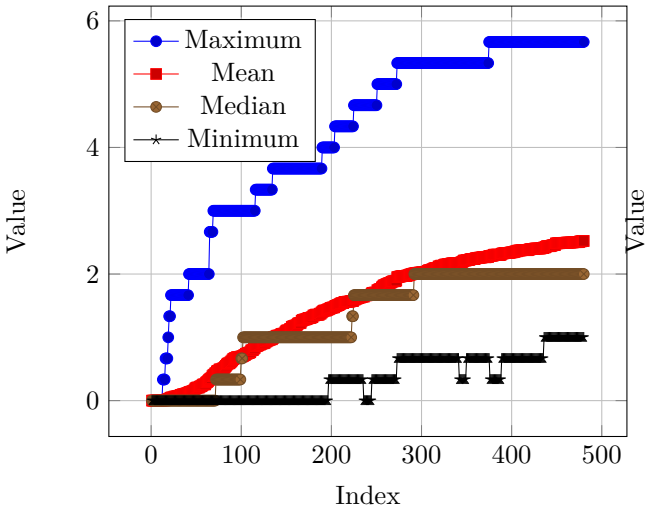


Food Eaten (Recently Dead Table)

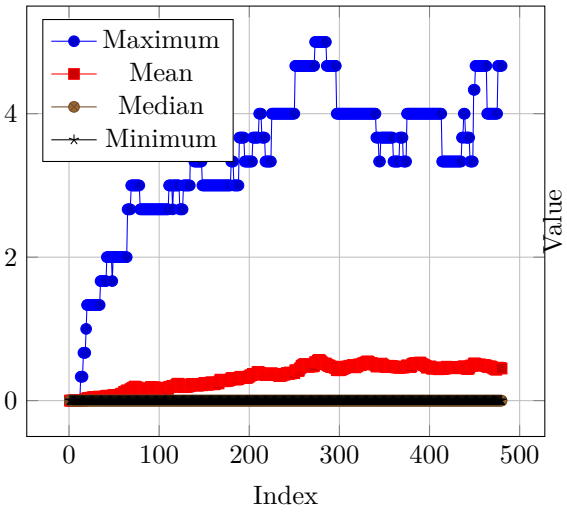



Generation
(Elite Fitness Table)
Generation
(Elite Novelty Table)
Generation
(Recently Dead Table)
Times Reproduced
(Elite Fitness Table)
Times Reproduced
(Elite Novelty Table)
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
0
10
20
30
40
0
2
4
6

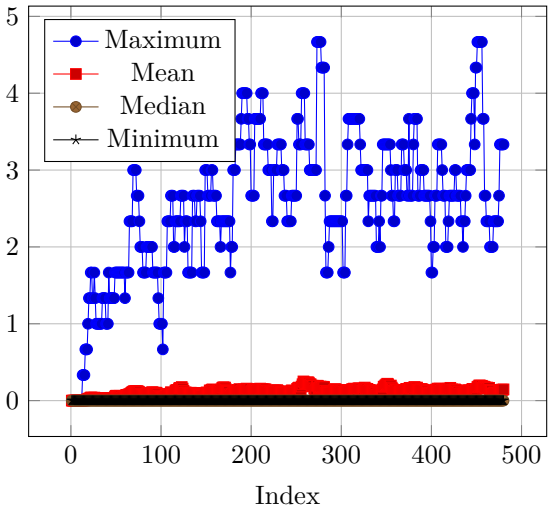
Times Reproduced Asexually (Elite Fitness Table)



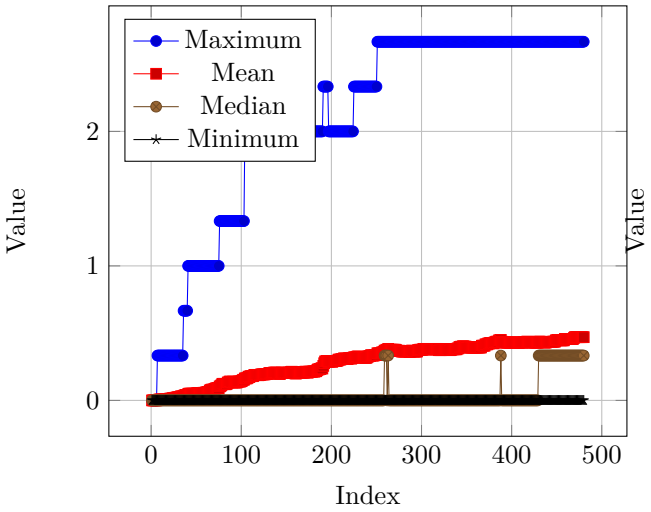
Times Reproduced Asexually (Elite Novelty Table)



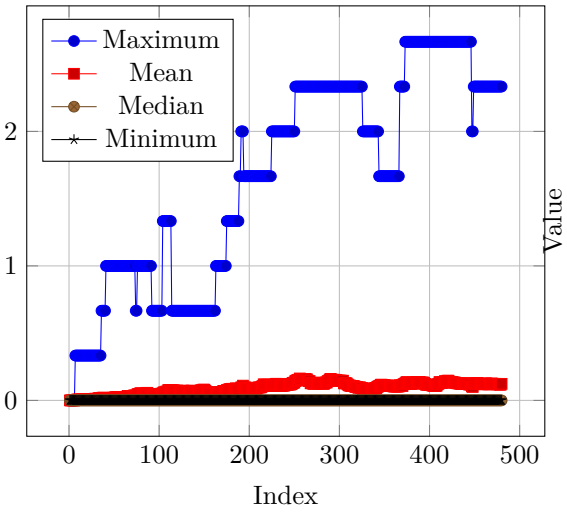
Times Reproduced Asexually (Recently Dead Table)



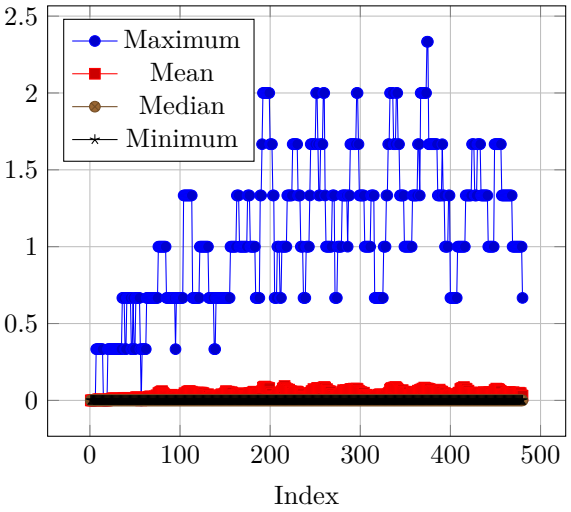
Times Reproduced Sexually (Elite Fitness Table)

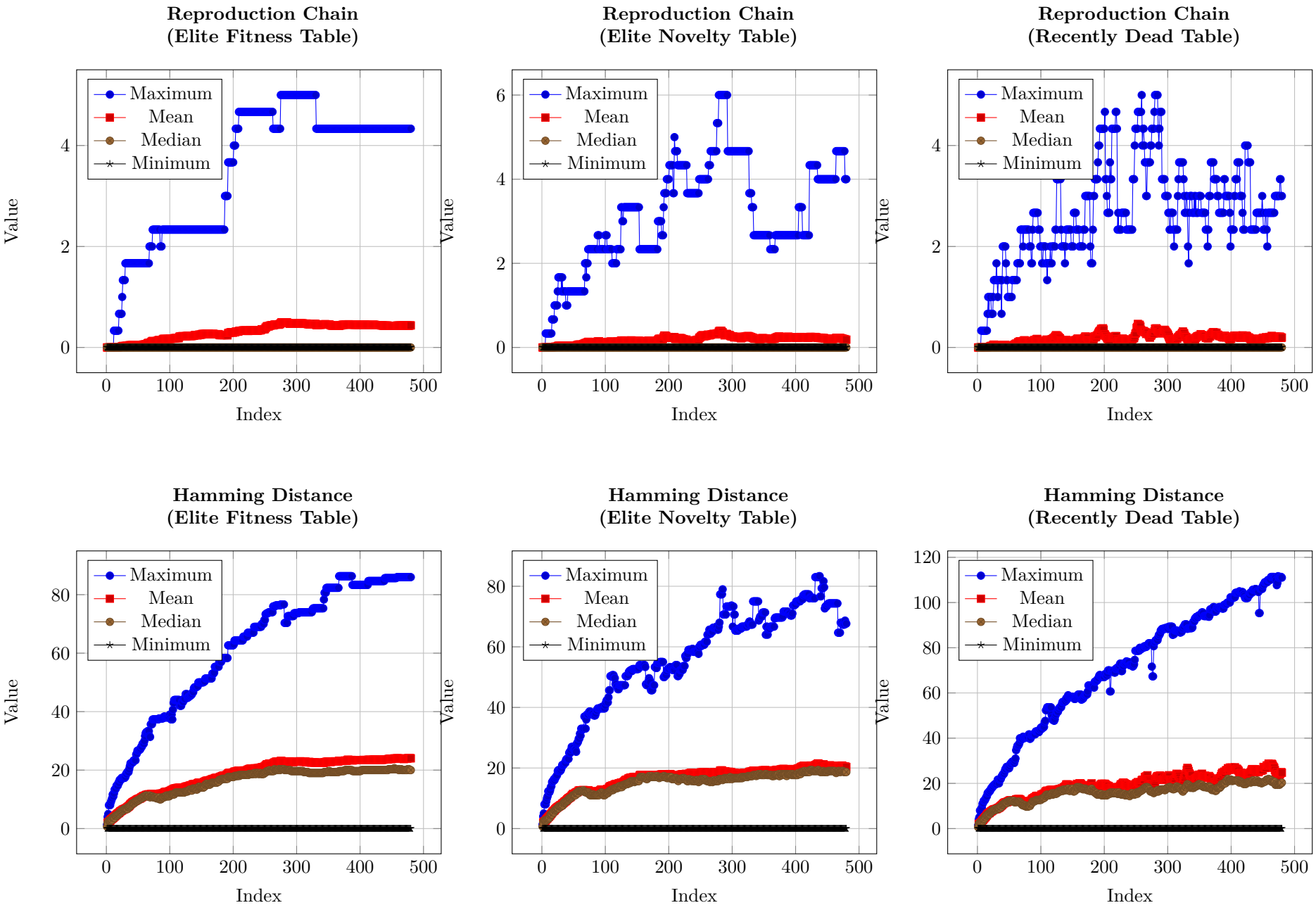


Times Reproduced Sexually (Elite Novelty Table)

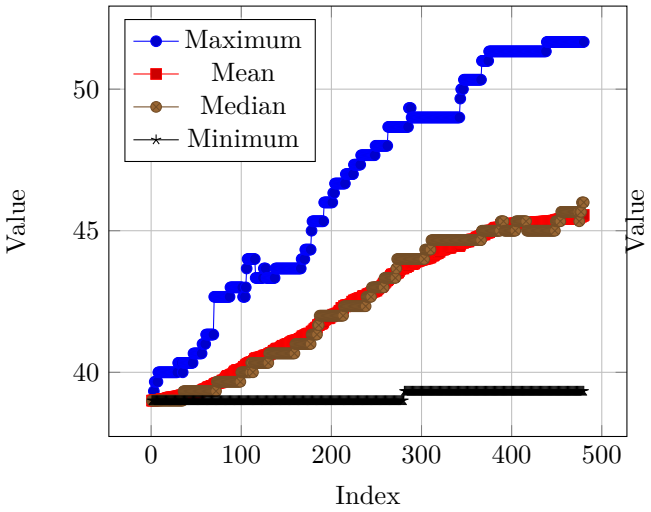


Times Reproduced Sexually (Recently Dead Table)

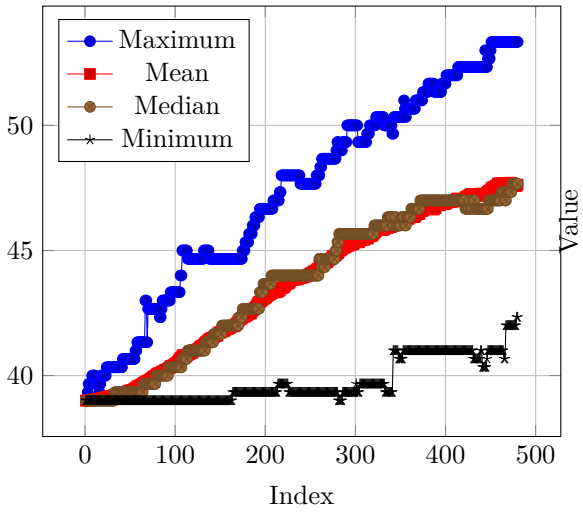



Reproduction Chain
(Elite Fitness Table)
Reproduction Chain
(Elite Novelty Table)
Reproduction Chain
(Recently Dead Table)
Hamming Distance
(Elite Fitness Table)
Hamming Distance
(Elite Novelty Table)
Hamming Distance
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
2
4
6
20
40
60
80
120

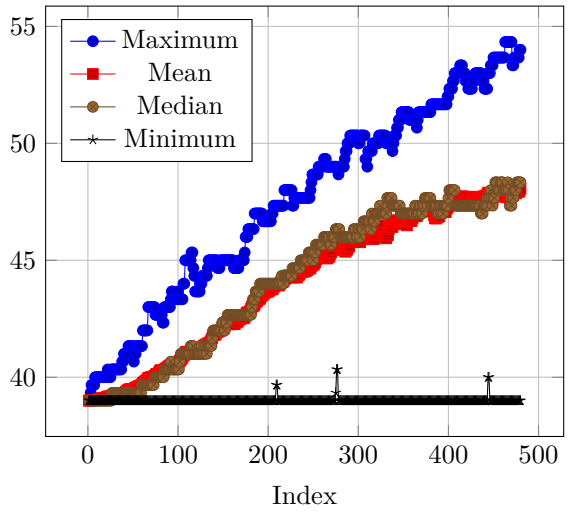
Number of Neurons (Elite Fitness Table)



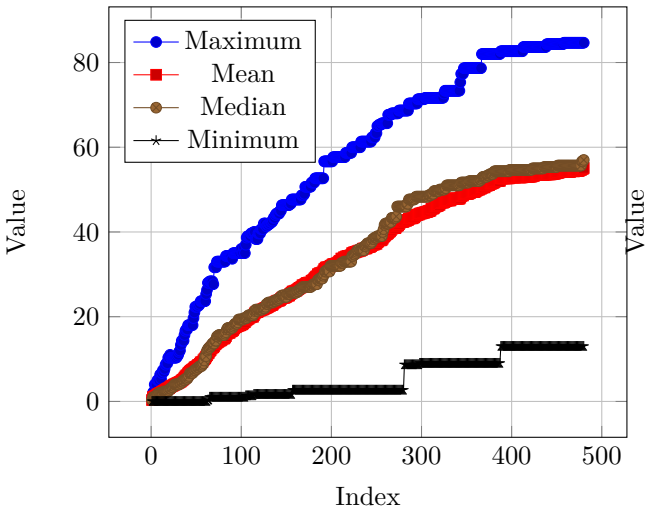
Number of Neurons (Elite Novelty Table)



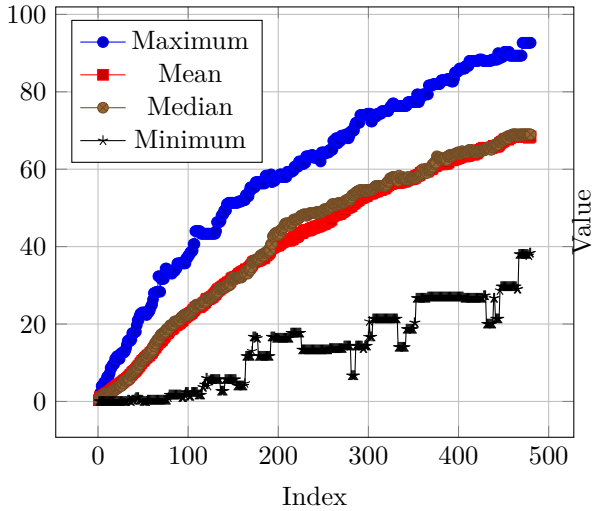
Number of Neurons (Recently Dead Table)



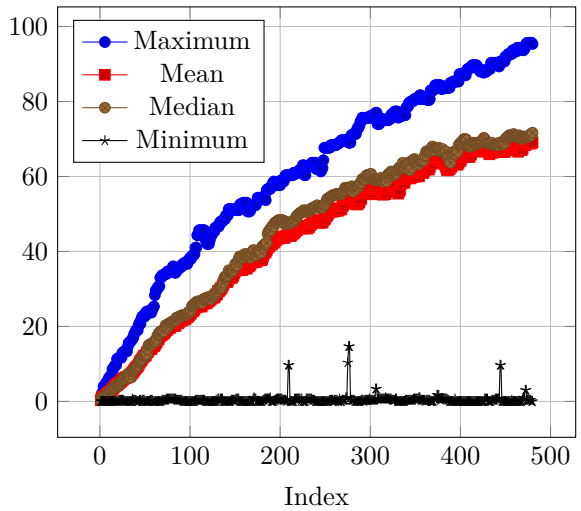
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)



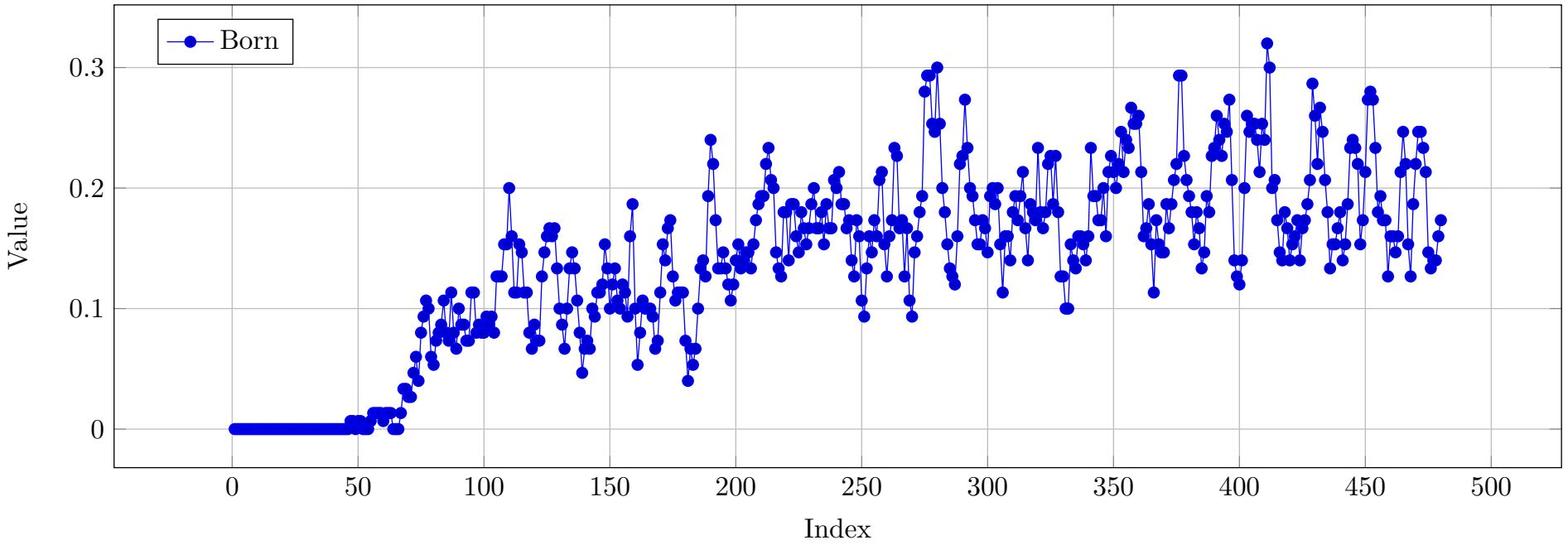
Number of Connections (Recently Dead Table)



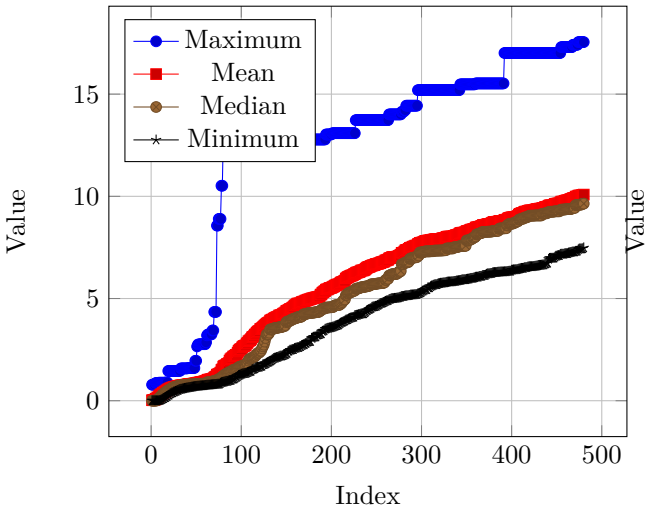
Wheeled Robot - NARS - No Learning - Flat World

RESULTS

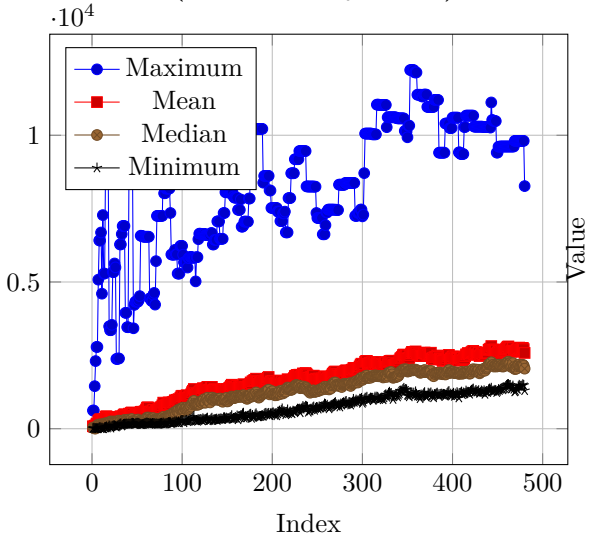
Percent of Minimum Population that was Born



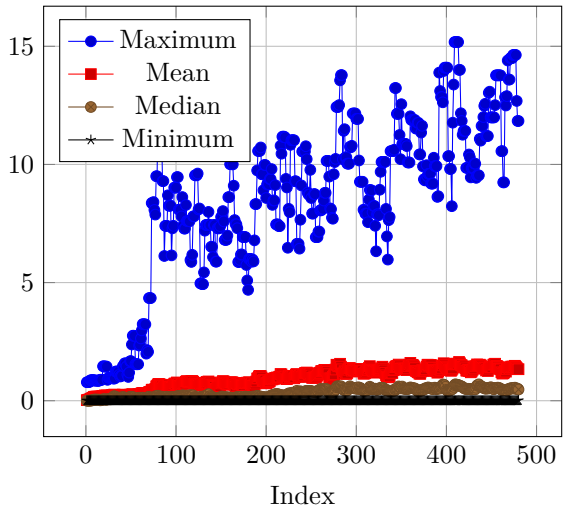
Fitness Score
(Elite Fitness Table)



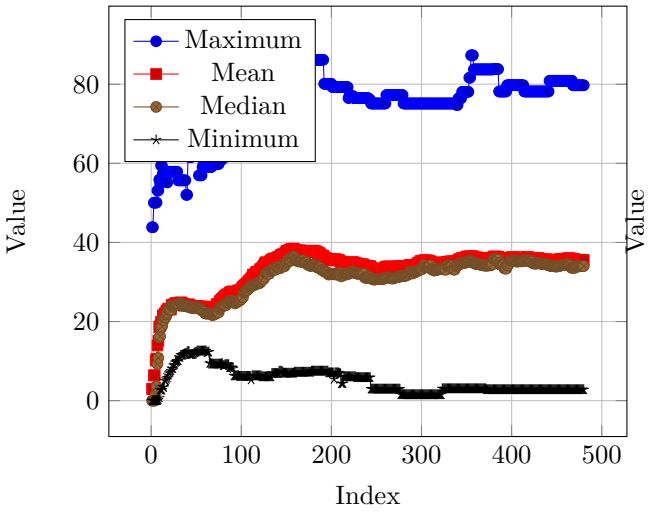
Novelty Score
(Elite Novelty Table)



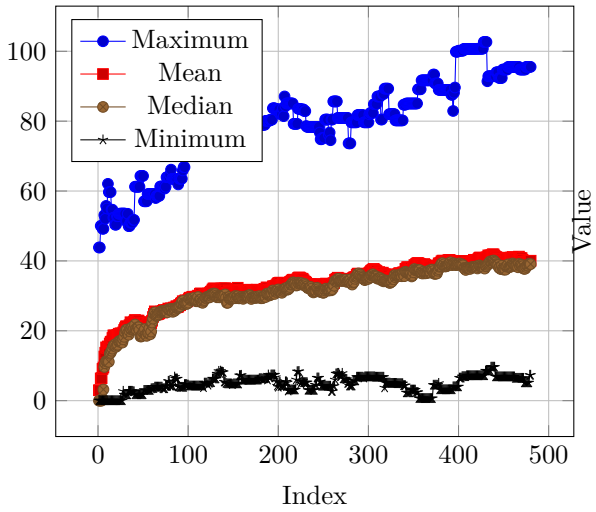
Fitness Score
(Recently Dead Table)



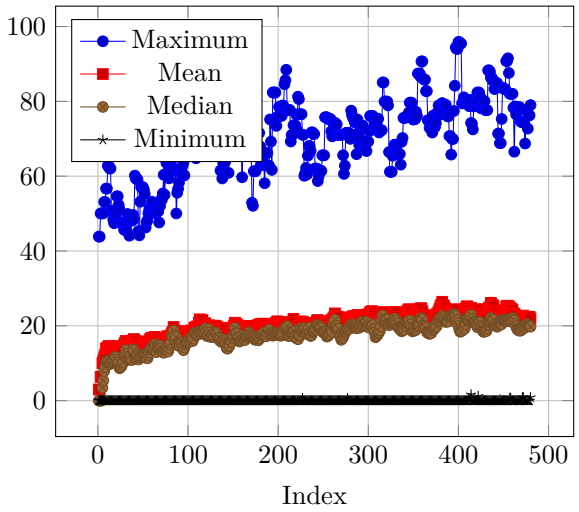
Displacement from Birthplace (Elite Fitness Table)



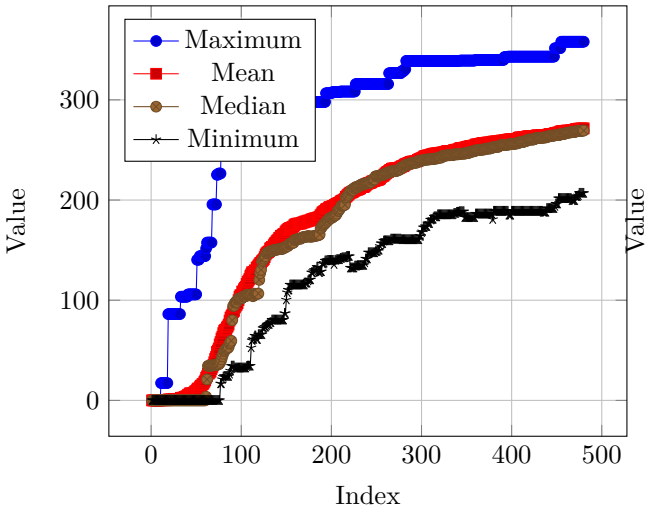
Displacement from Birthplace (Elite Novelty Table)



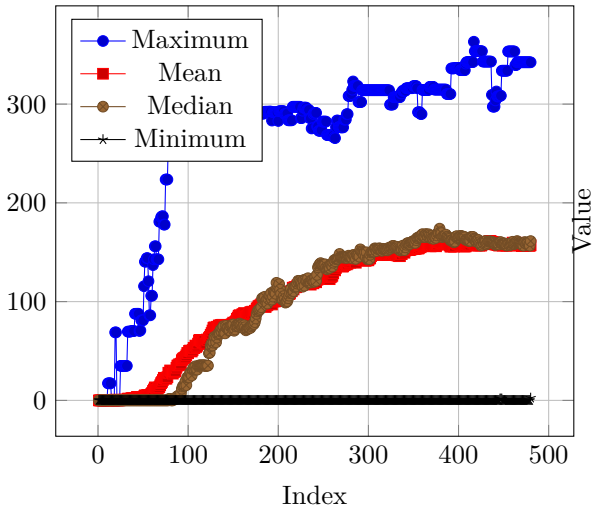
Displacement from Birthplace (Recently Dead Table)



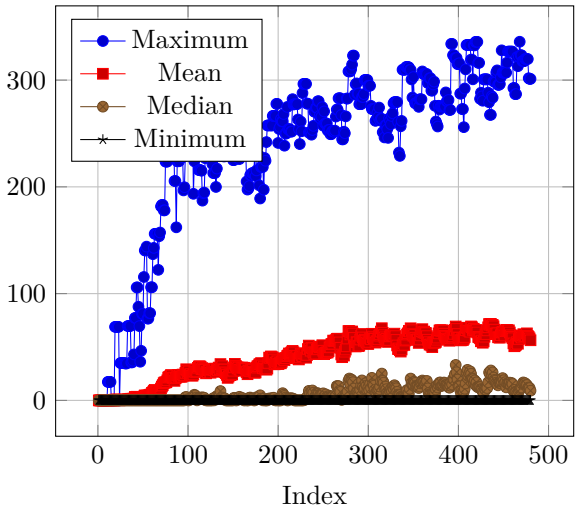
Food Eaten (Elite Fitness Table)

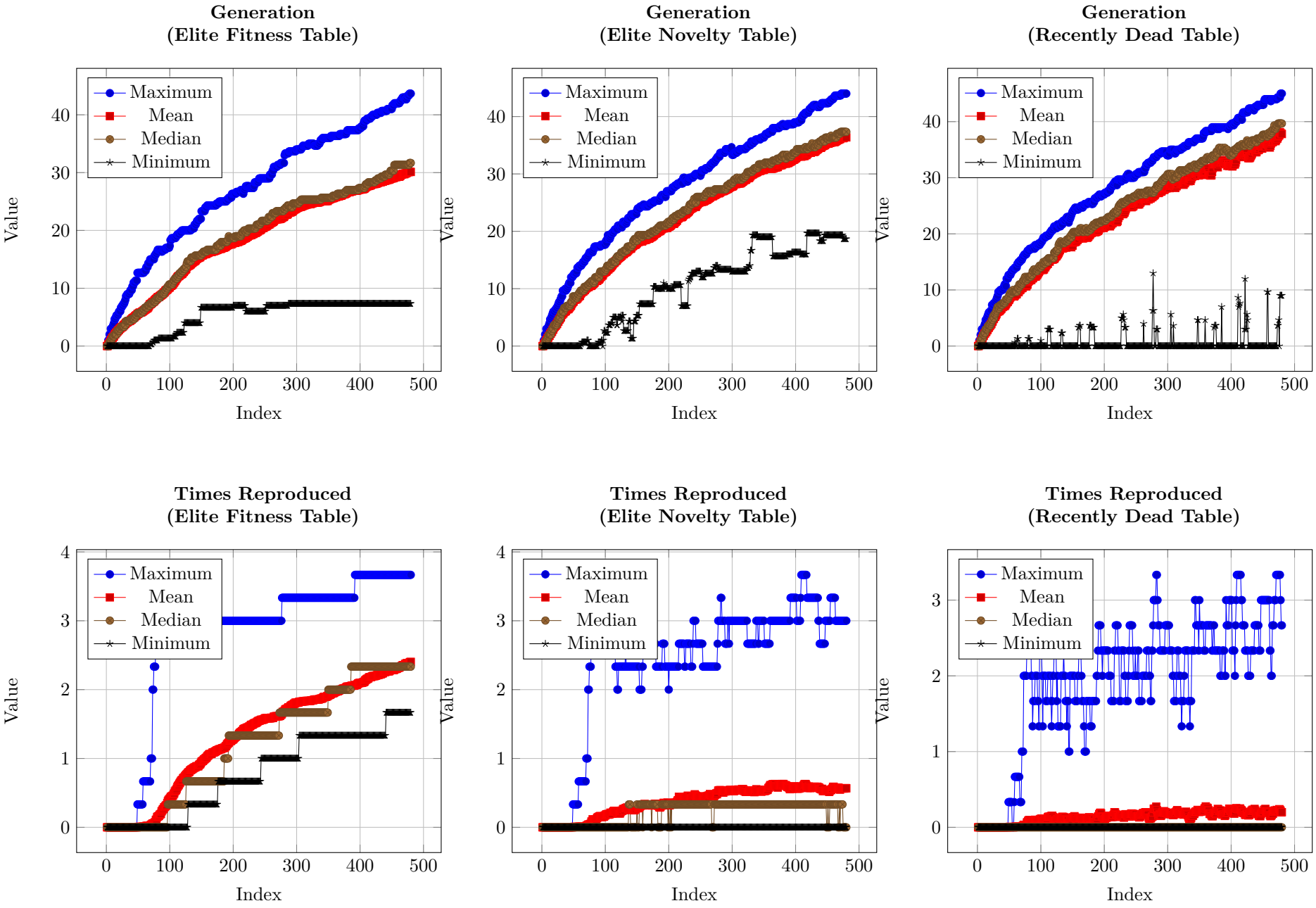


Food Eaten (Elite Novelty Table)

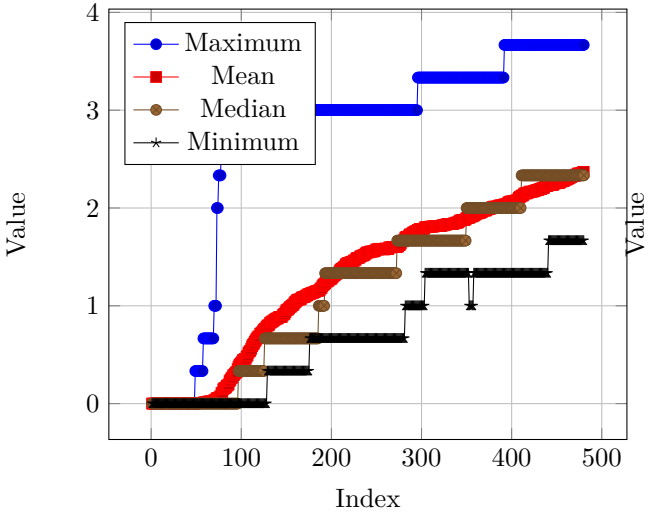


Food Eaten (Recently Dead Table)

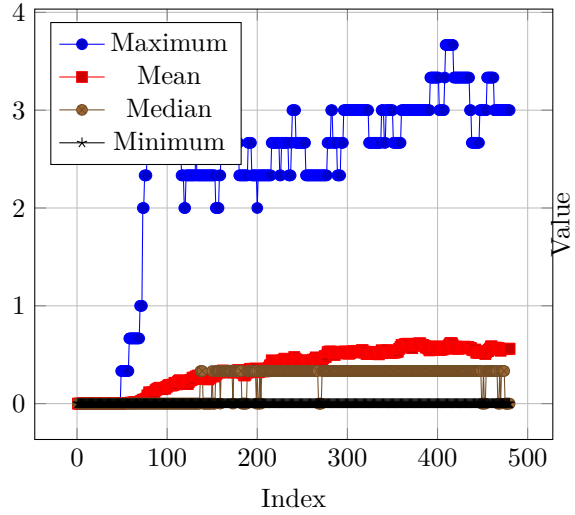



Generation
(Elite Fitness Table)
Generation
(Elite Novelty Table)
Generation
(Recently Dead Table)
Times Reproduced
(Elite Fitness Table)
Times Reproduced
(Elite Novelty Table)
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
10
20
30
40
1
2
3
4

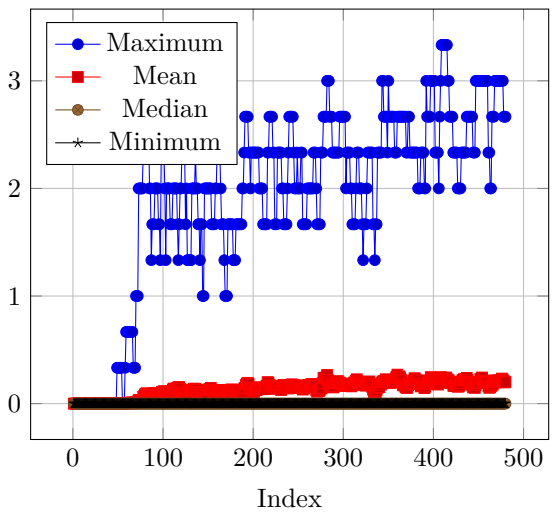
Times Reproduced Asexually (Elite Fitness Table)



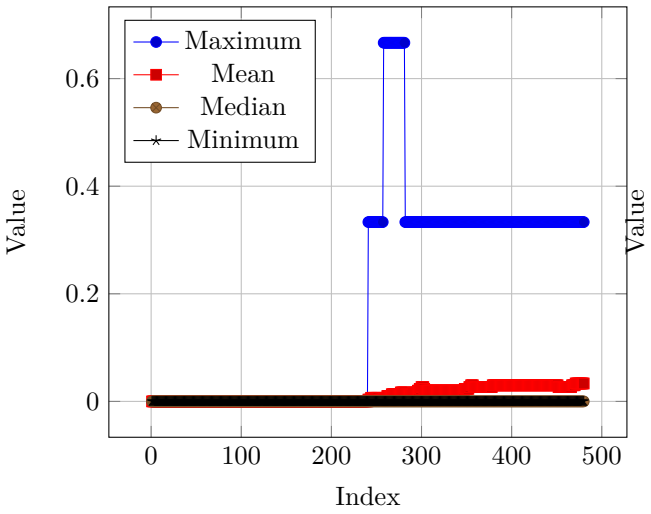
Times Reproduced Asexually (Elite Novelty Table)



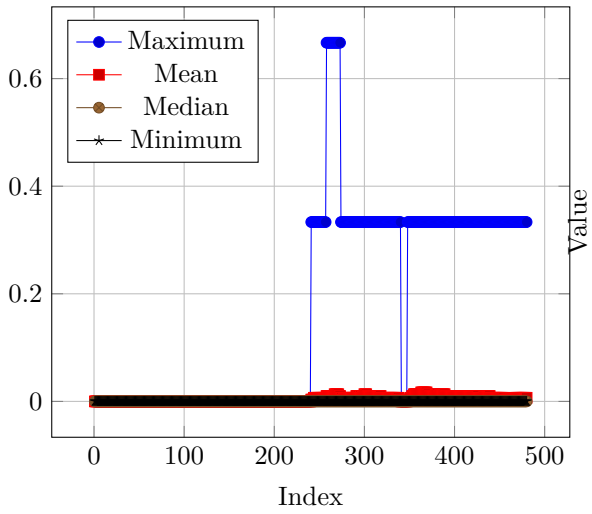
Times Reproduced Asexually (Recently Dead Table)



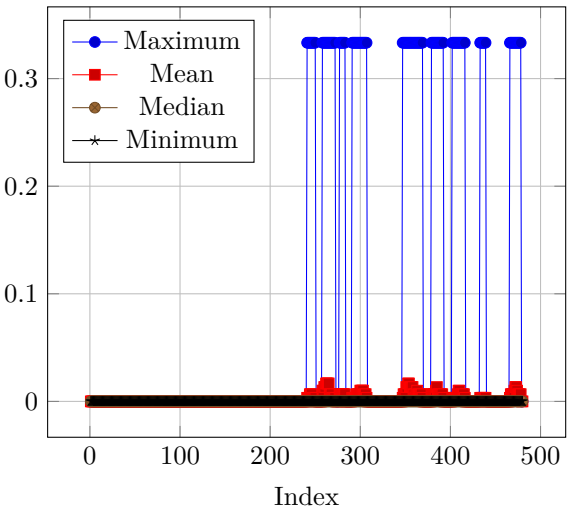
Times Reproduced Sexually (Elite Fitness Table)

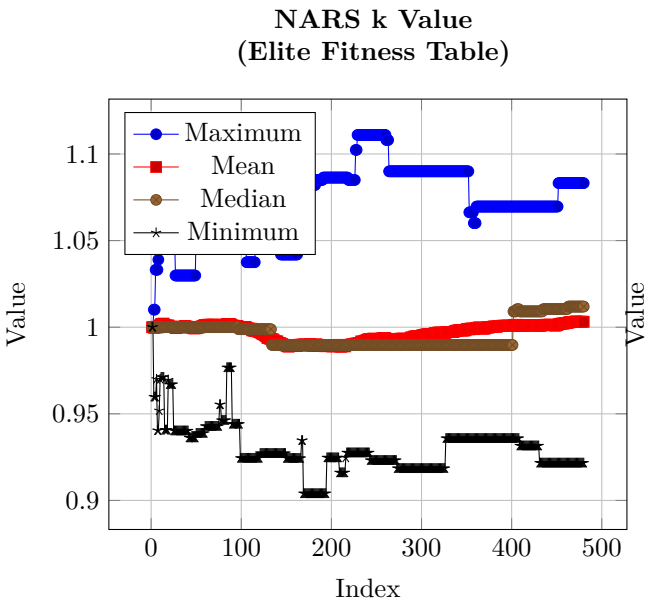


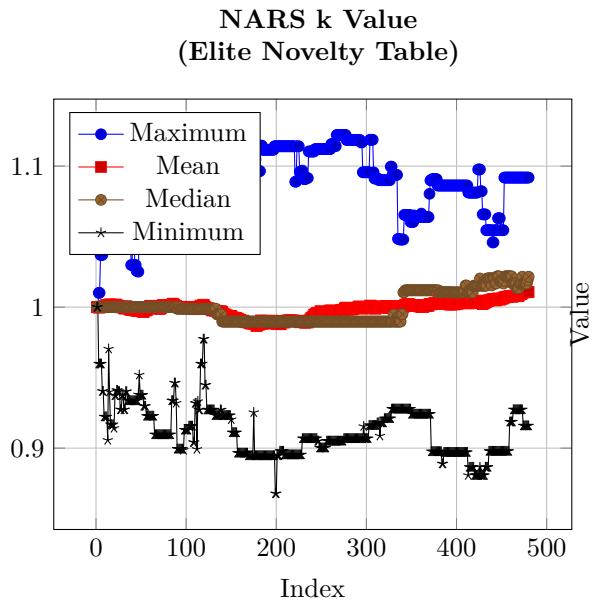
Times Reproduced Sexually (Elite Novelty Table)

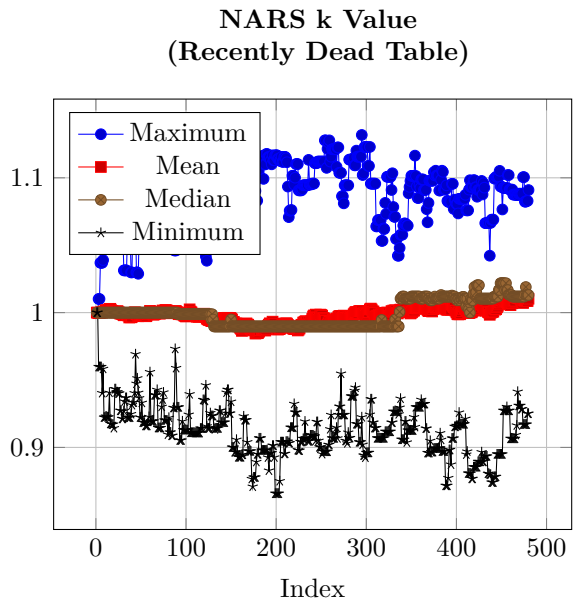


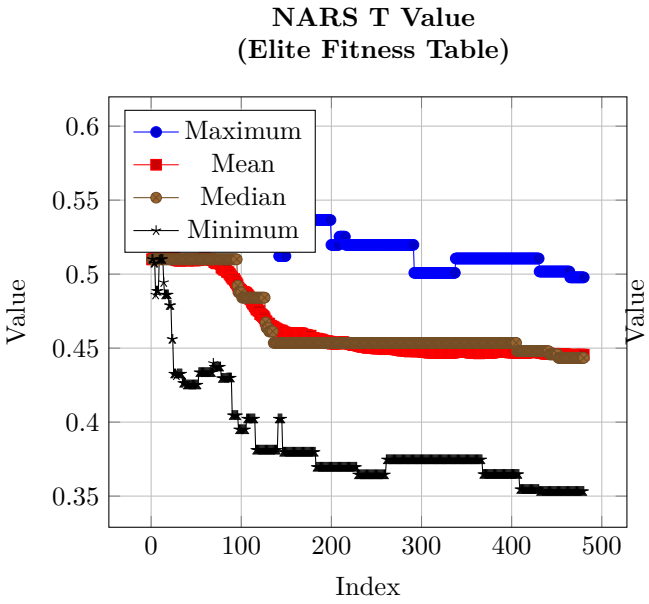
Times Reproduced Sexually (Recently Dead Table)

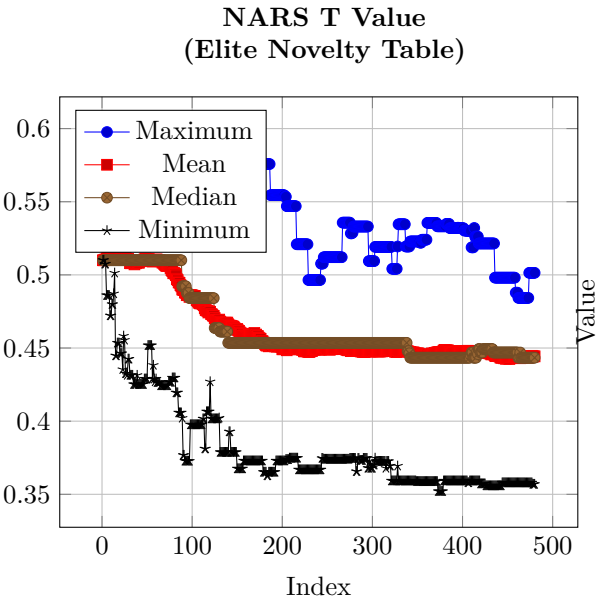


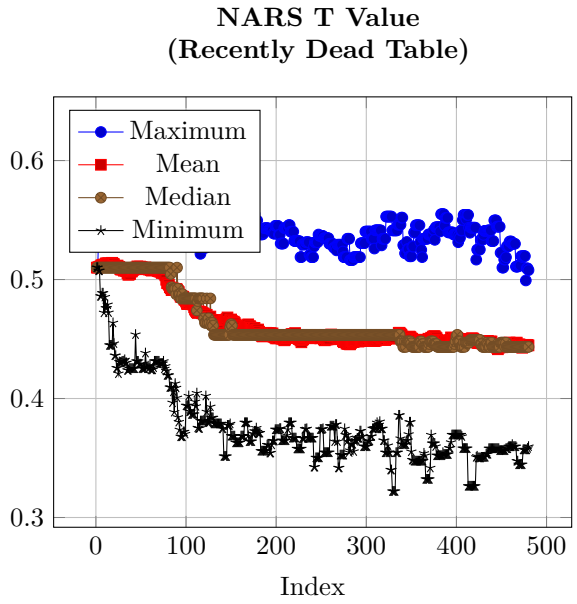

NARS k Value
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
1.1
1.05
1
0.95
0.9
0
100
200
300
400
500


NARS k Value
(Elite Novelty Table)
Maximum
Mean
Median
Minimum
Value
Index
1.1
1
0.9
0
100
200
300
400
500

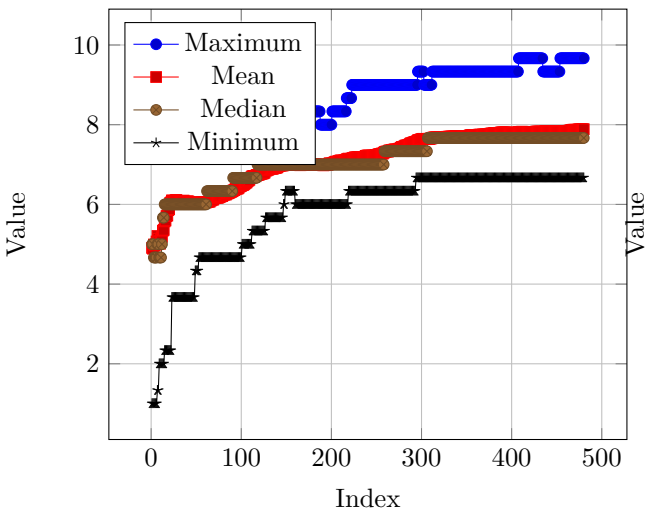

NARS k Value
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
1.1
1
0.9
0
100
200
300
400
500


NARS T Value
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
0.6
0.55
0.5
0.45
0.4
0.35
0
100
200
300
400
500

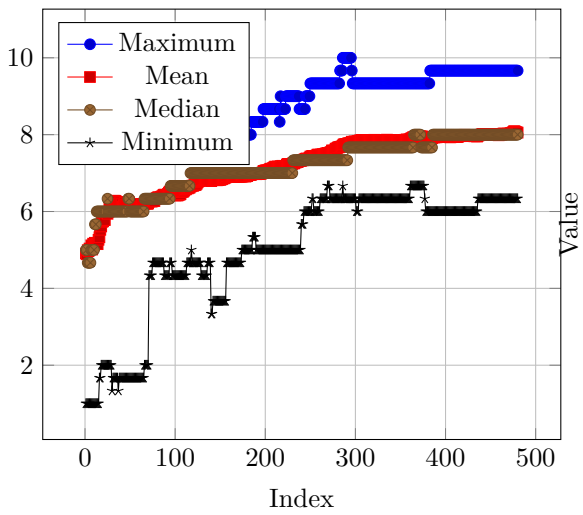

NARS T Value
(Elite Novelty Table)
Maximum
Mean
Median
Minimum
Value
Index
0.6
0.55
0.5
0.45
0.4
0.35
0
100
200
300
400
500


NARS T Value
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0.6
0.5
0.4
0.3
0
100
200
300
400
500

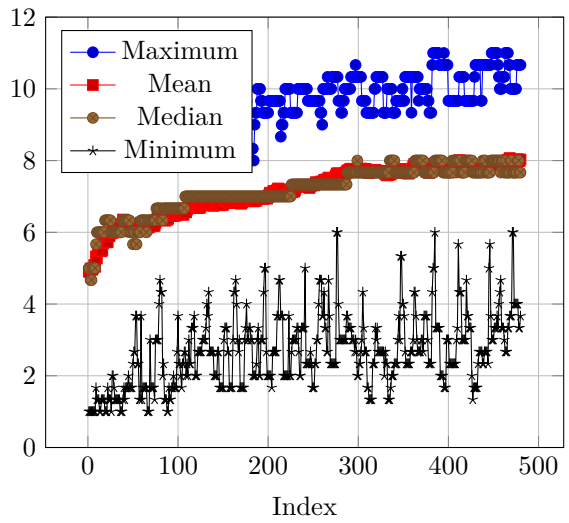
Number of Beliefs (Elite Fitness Table)



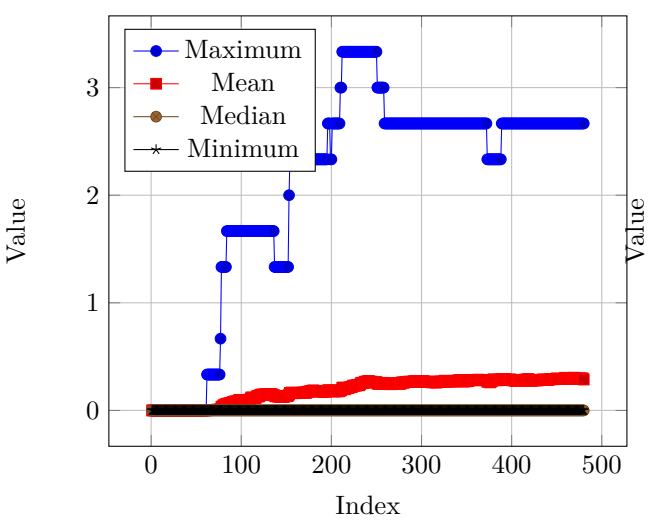
Number of Beliefs (Elite Novelty Table)



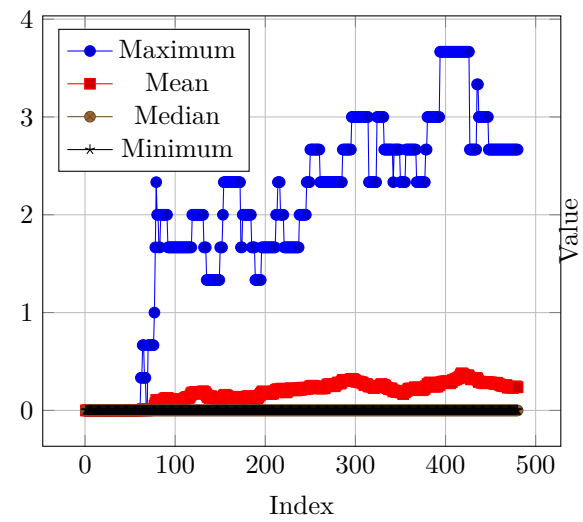
Number of Beliefs (Recently Dead Table)



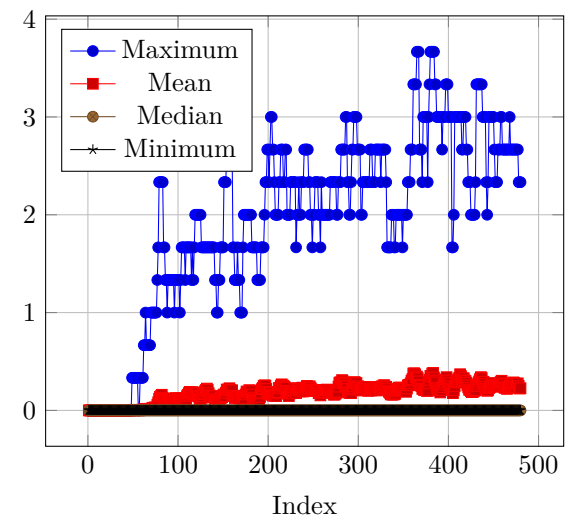
Reproduction Chain (Elite Fitness Table)



Reproduction Chain (Elite Novelty Table)



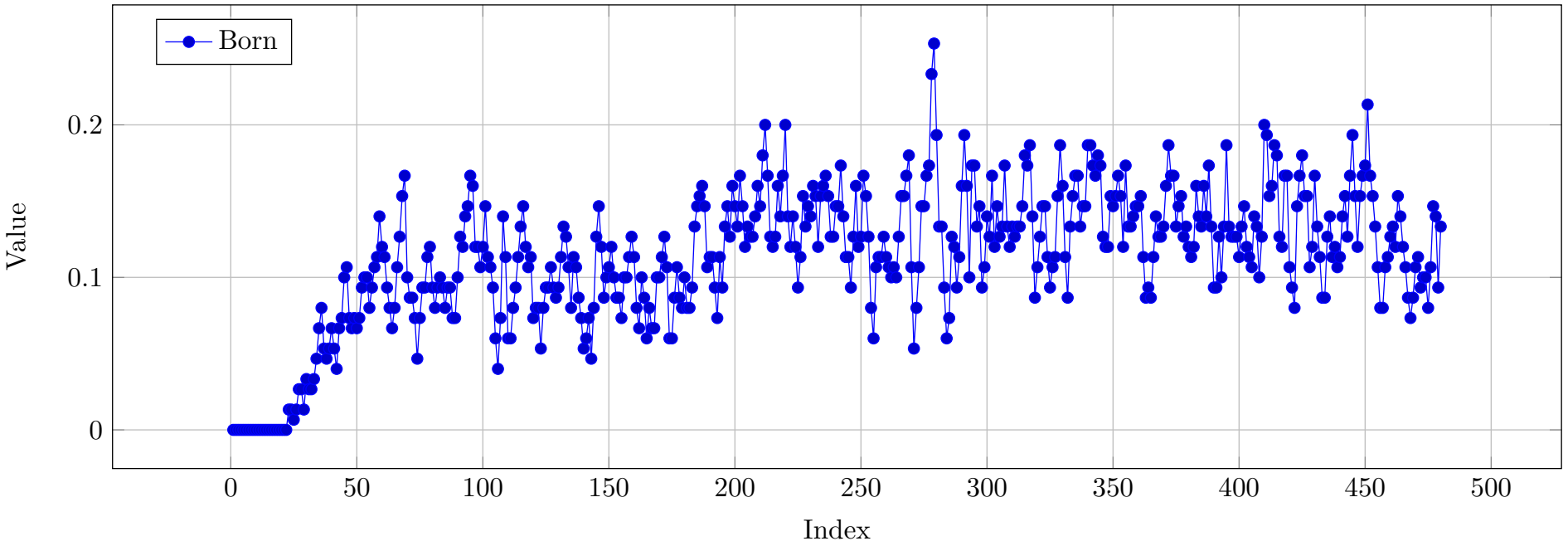
Reproduction Chain (Recently Dead Table)



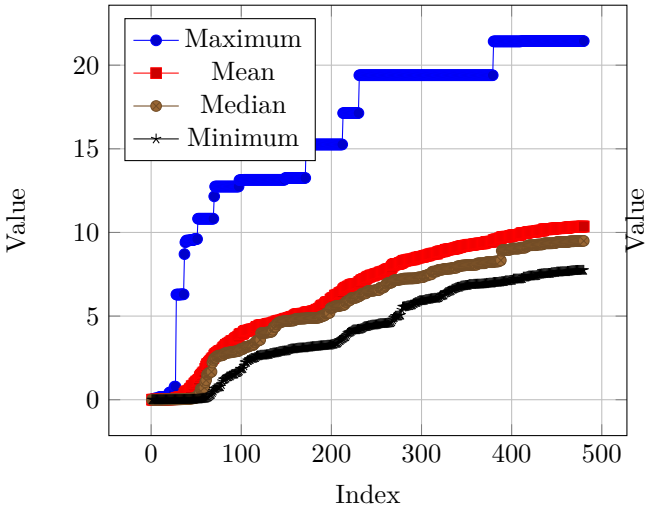
Soft Voxel Robot - Sum & Squash - No Learning - Flat World

RESULTS

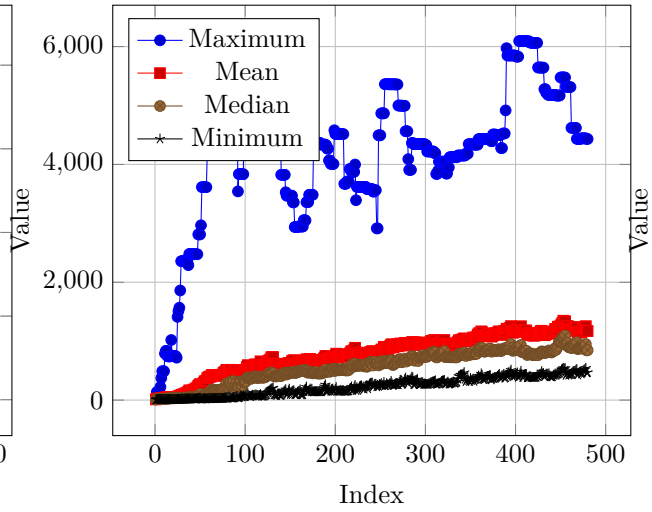
Percent of Minimum Population that was Born



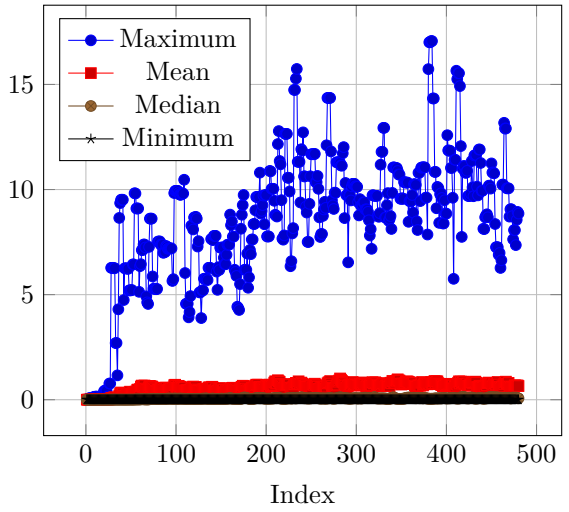
Fitness Score
(Elite Fitness Table)



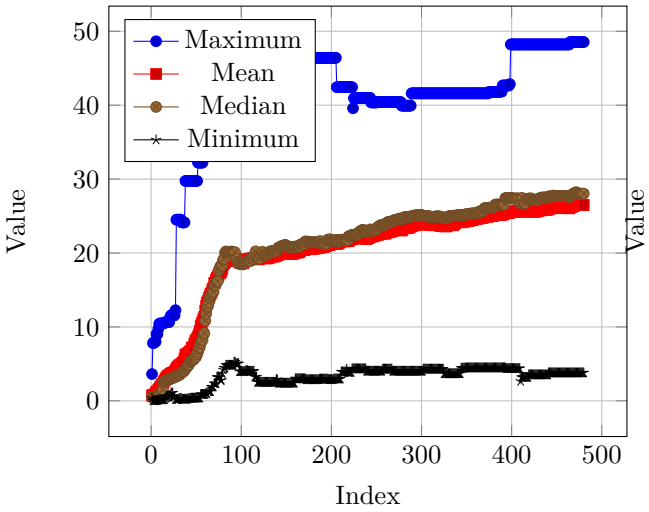
Novelty Score
(Elite Novelty Table)



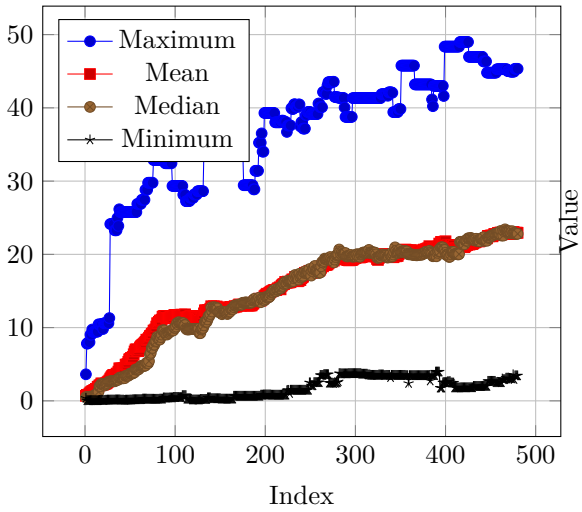
Fitness Score
(Recently Dead Table)



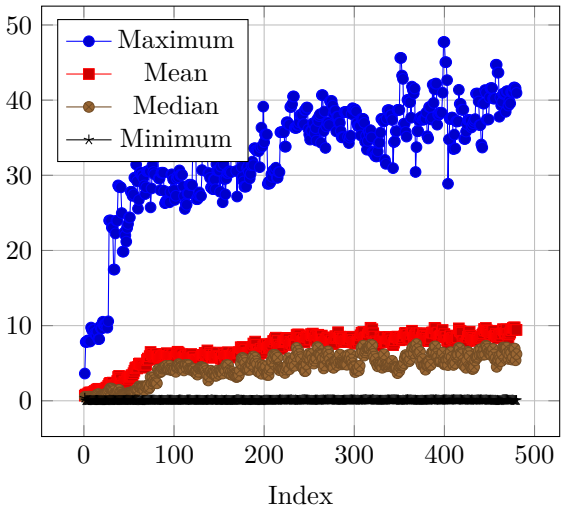
Displacement from Birthplace (Elite Fitness Table)



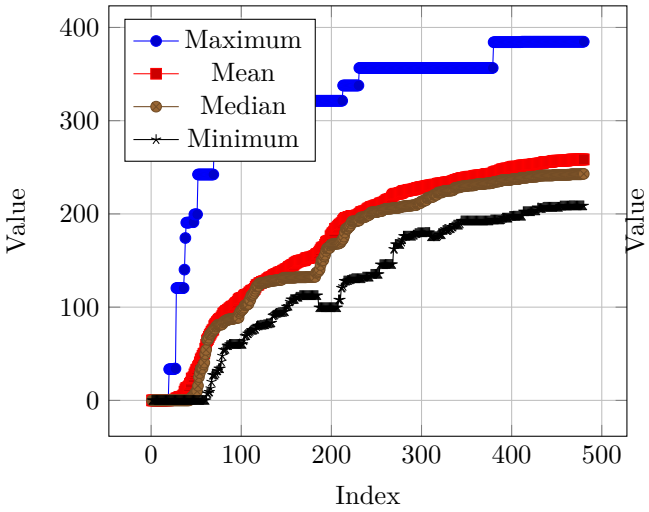
Displacement from Birthplace (Elite Novelty Table)



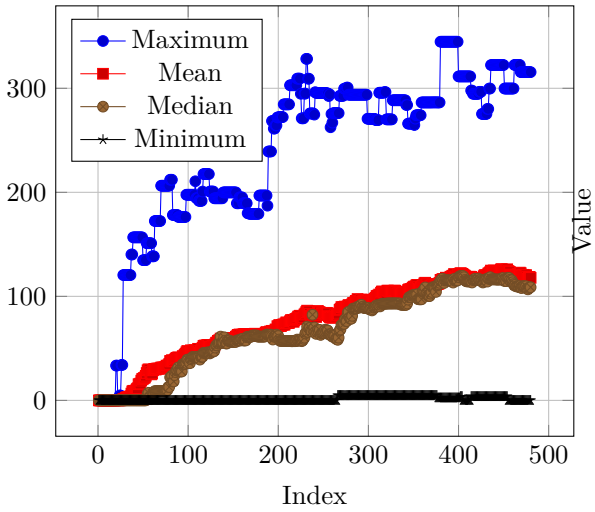
Displacement from Birthplace (Recently Dead Table)



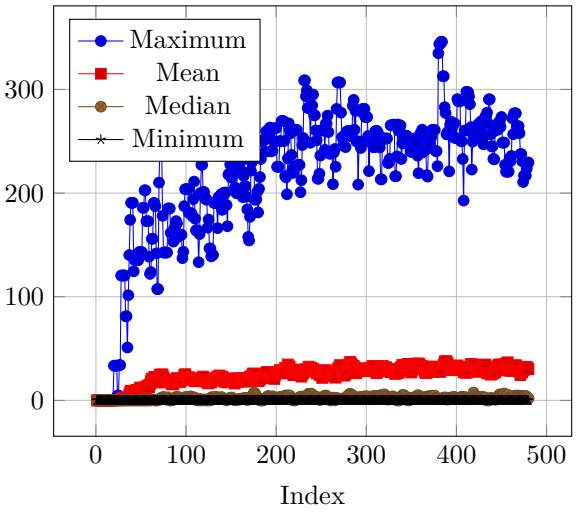
Food Eaten (Elite Fitness Table)

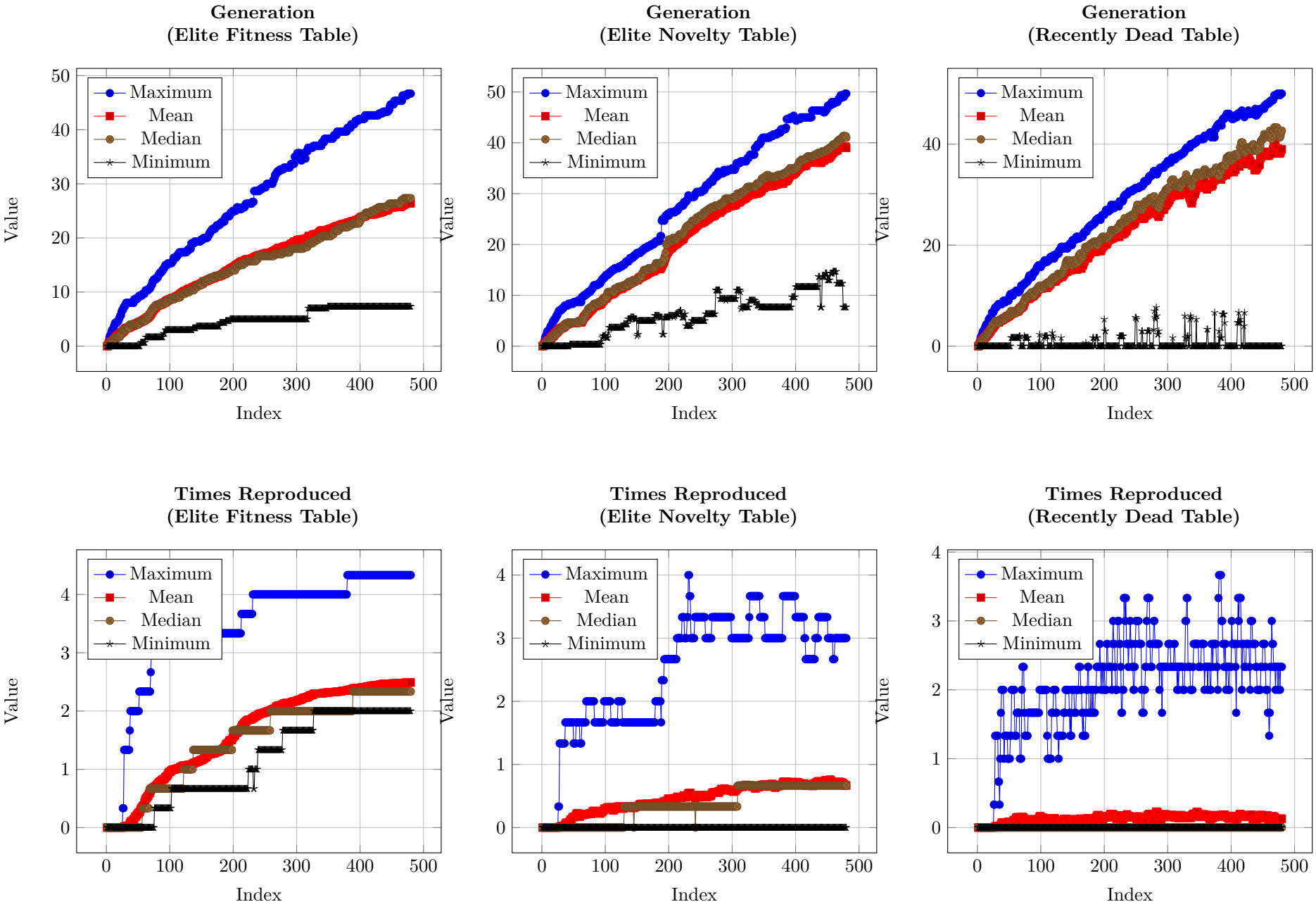


Food Eaten (Elite Novelty Table)

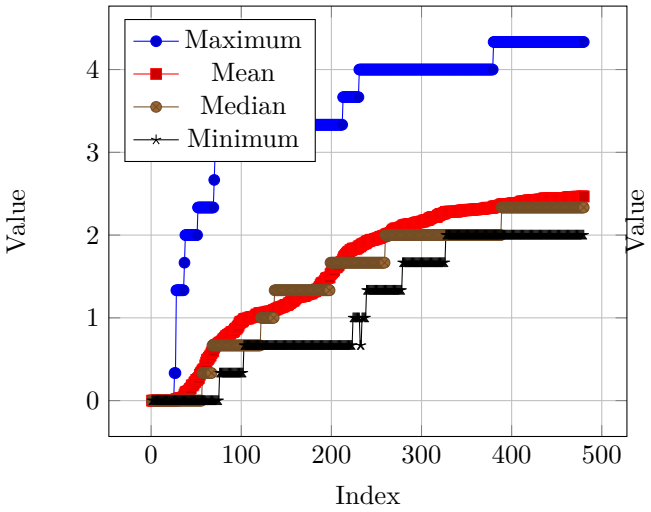


Food Eaten (Recently Dead Table)

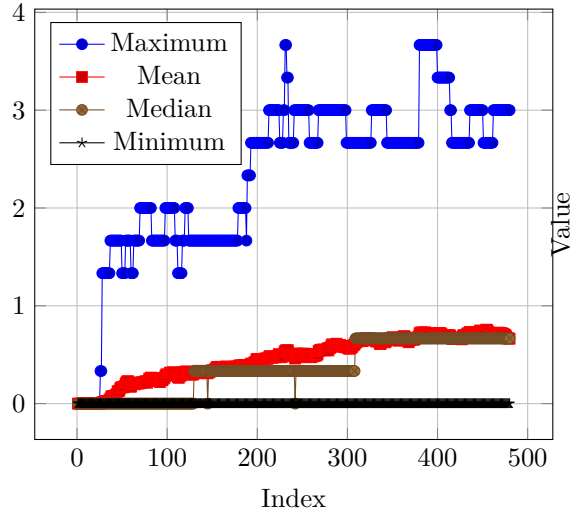



Generation
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
Generation
(Elite Novelty Table)
Maximum
Mean
Median
Minimum
Value
Index
Generation
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Index
Times Reproduced
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
Times Reproduced
(Elite Novelty Table)
Maximum
Mean
Median
Minimum
Value
Index
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Index

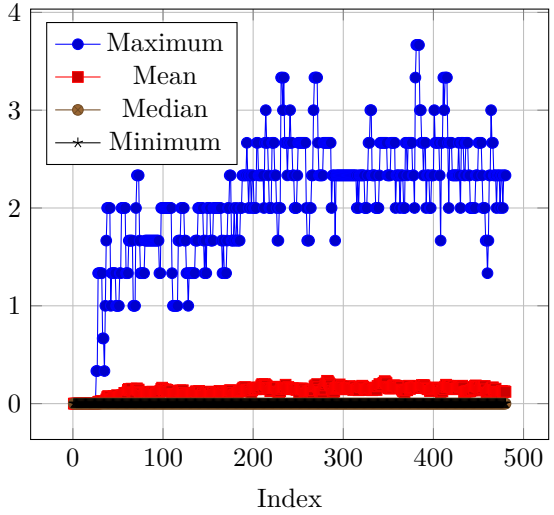
Times Reproduced Asexually (Elite Fitness Table)



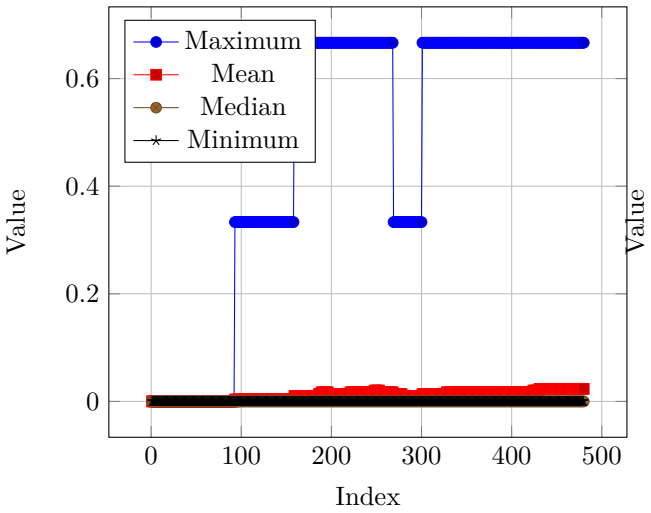
Times Reproduced Asexually (Elite Novelty Table)



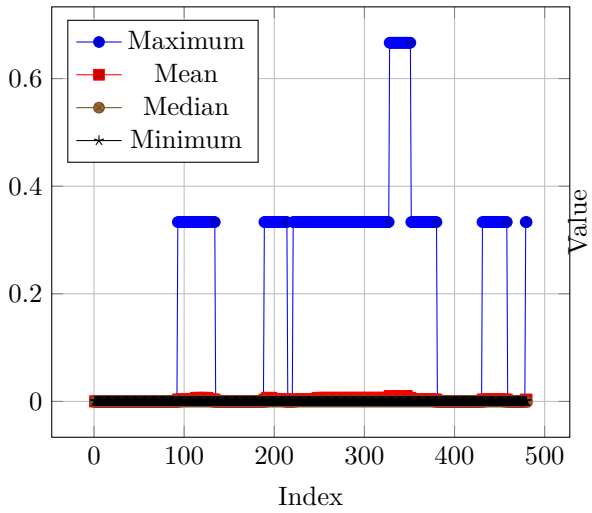
Times Reproduced Asexually (Recently Dead Table)



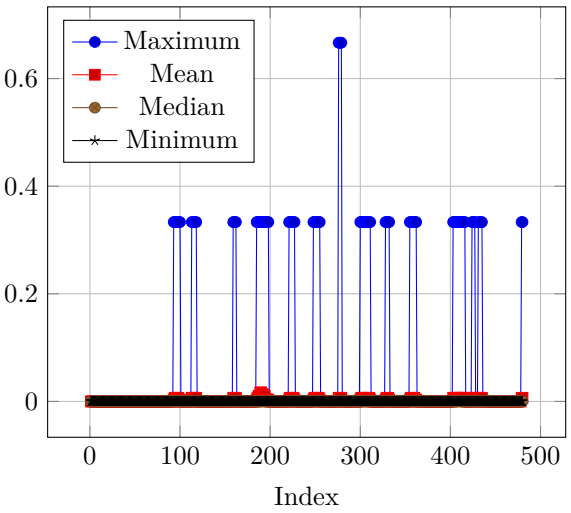
Times Reproduced Sexually (Elite Fitness Table)



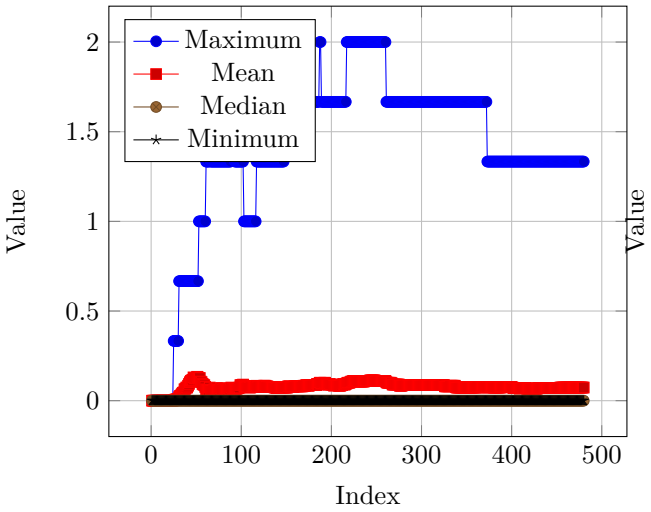
Times Reproduced Sexually (Elite Novelty Table)



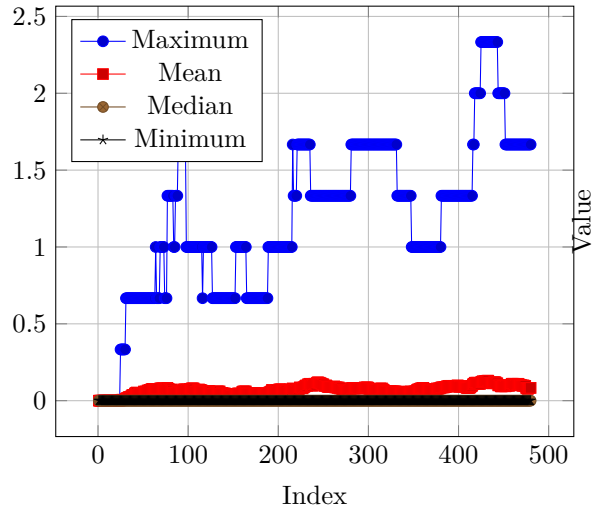
Times Reproduced Sexually (Recently Dead Table)



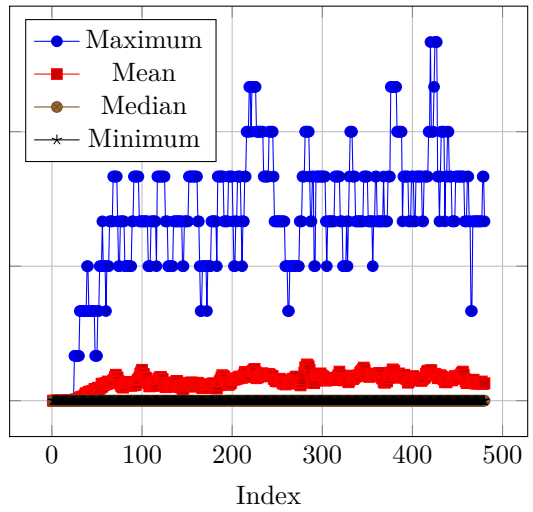
Reproduction Chain (Elite Fitness Table)



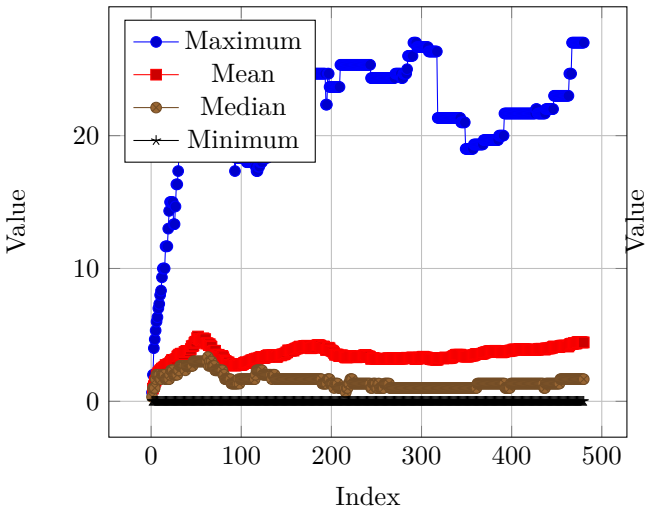
Reproduction Chain (Elite Novelty Table)



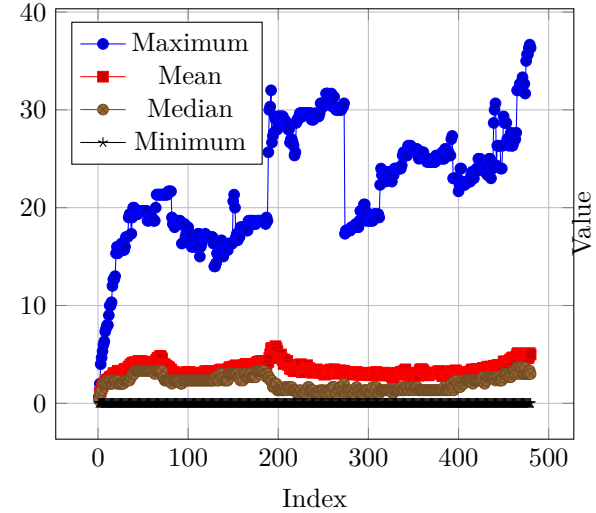
Reproduction Chain (Recently Dead Table)



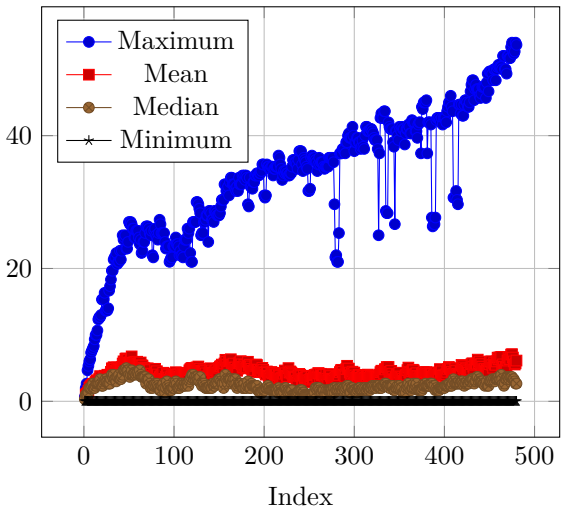
Hamming Distance (Elite Fitness Table)



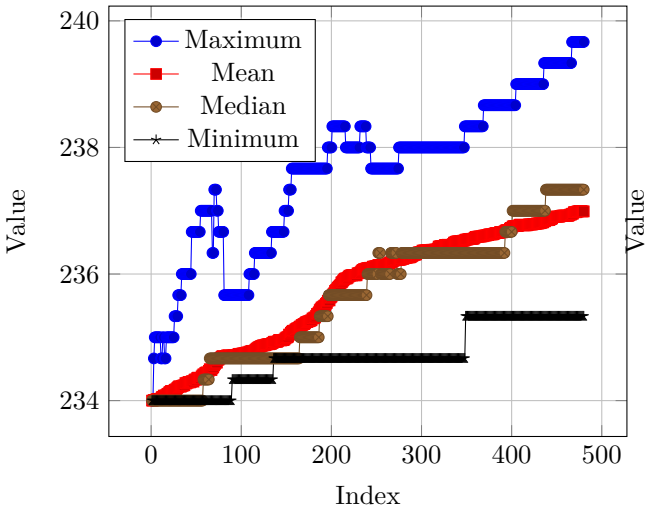
Hamming Distance (Elite Novelty Table)



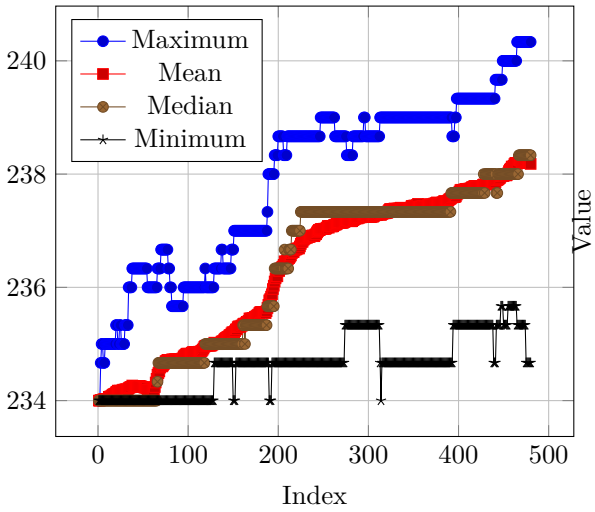
Hamming Distance (Recently Dead Table)



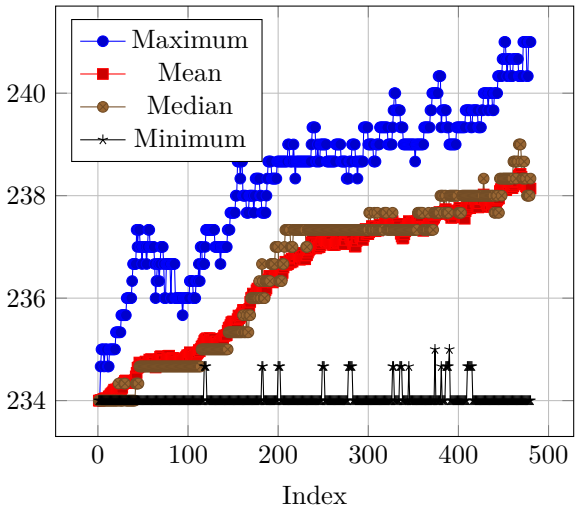
Number of Neurons (Elite Fitness Table)



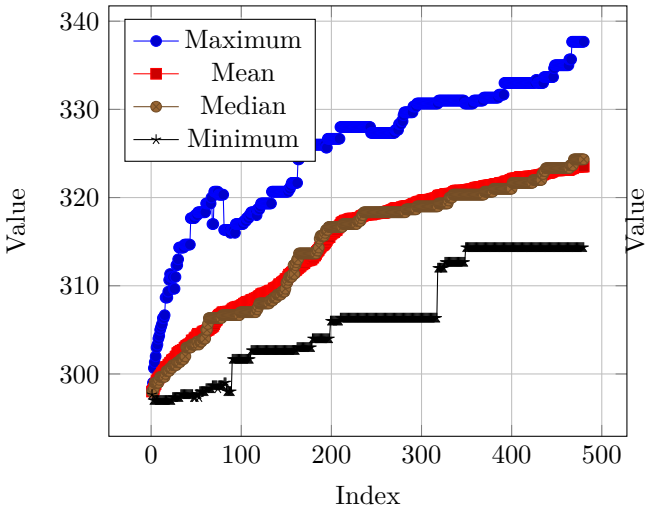
Number of Neurons (Elite Novelty Table)



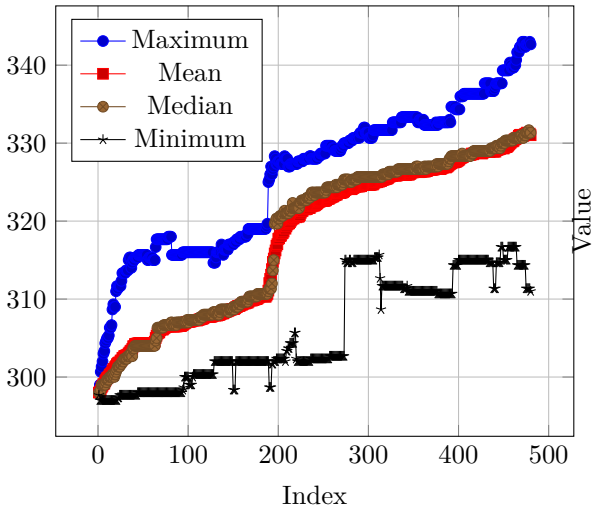
Number of Neurons (Recently Dead Table)



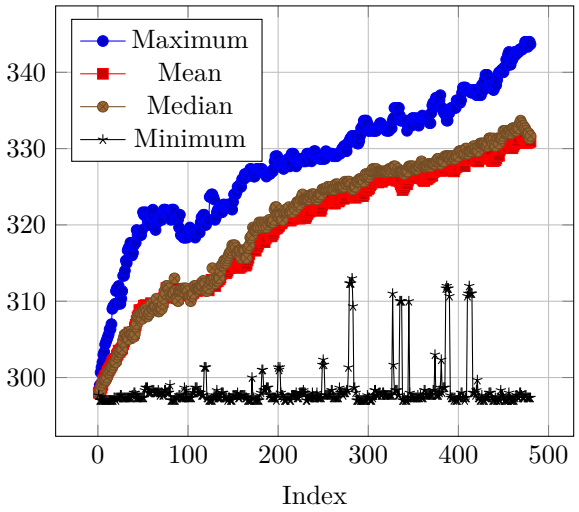
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)



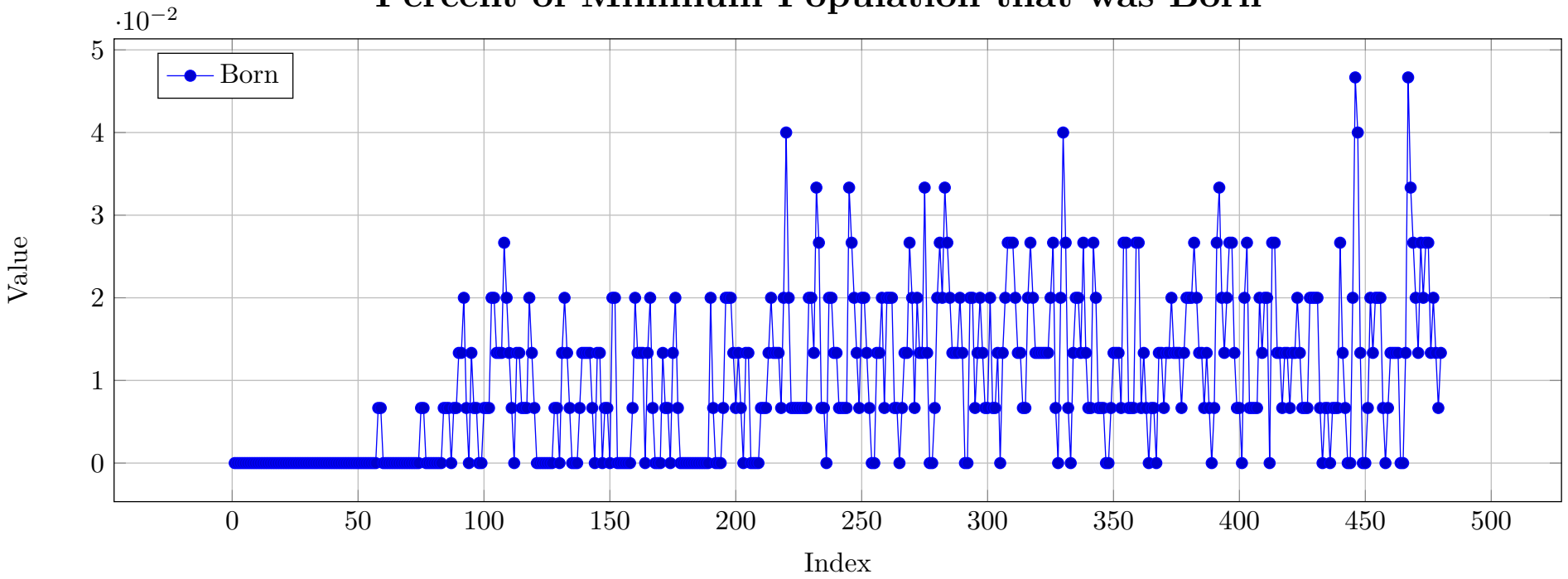
Number of Connections (Recently Dead Table)



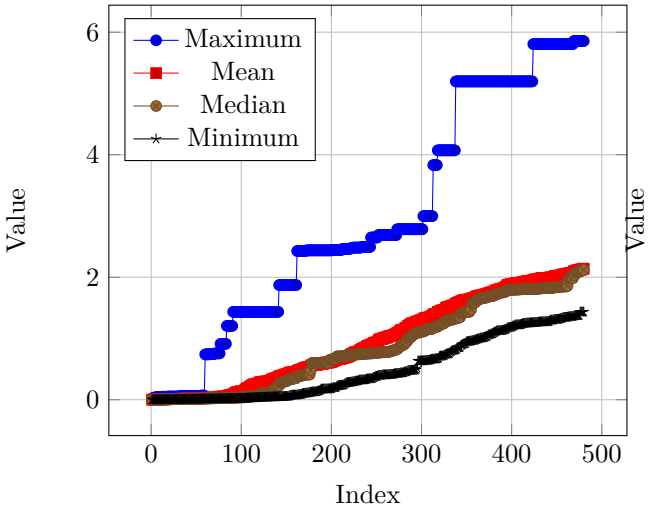
Articulated Robot - Sum & Squash - Hebb ABCD - Flat World

RESULTS

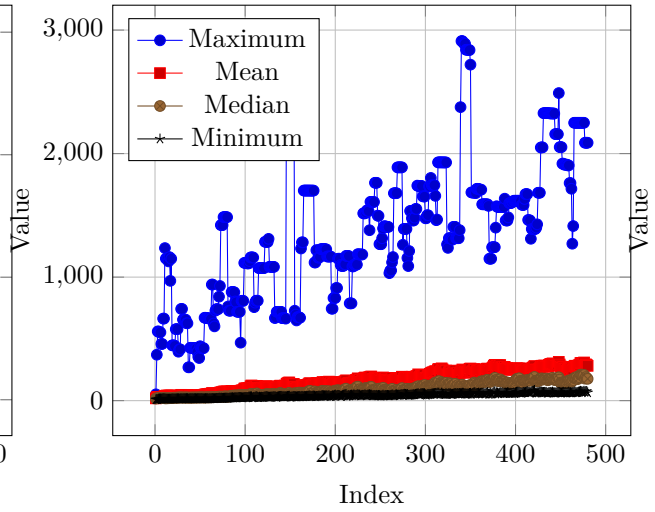
Percent of Minimum Population that was Born



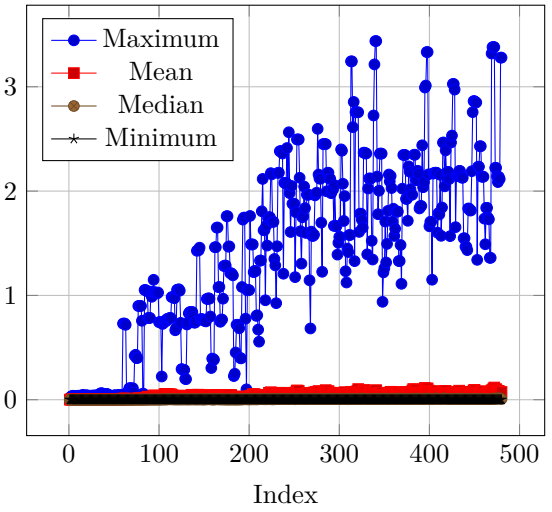
Fitness Score
(Elite Fitness Table)



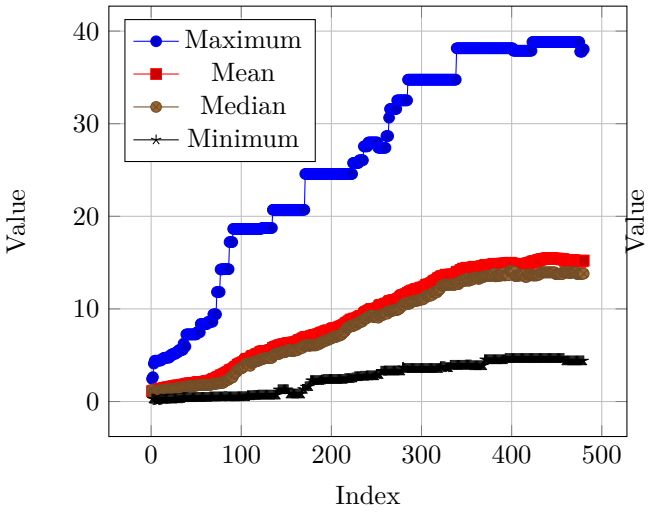
Novelty Score
(Elite Novelty Table)



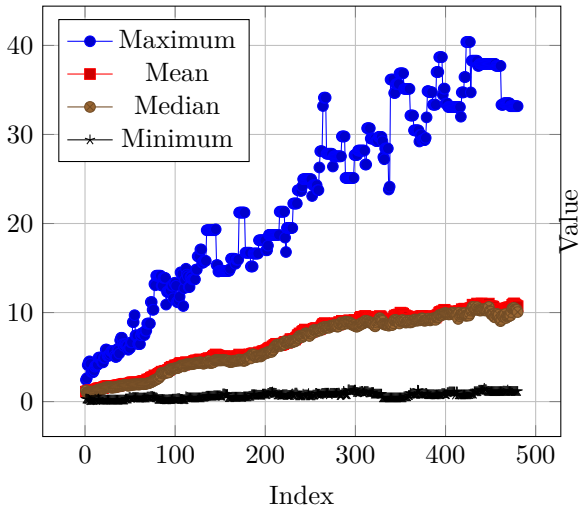
Fitness Score
(Recently Dead Table)



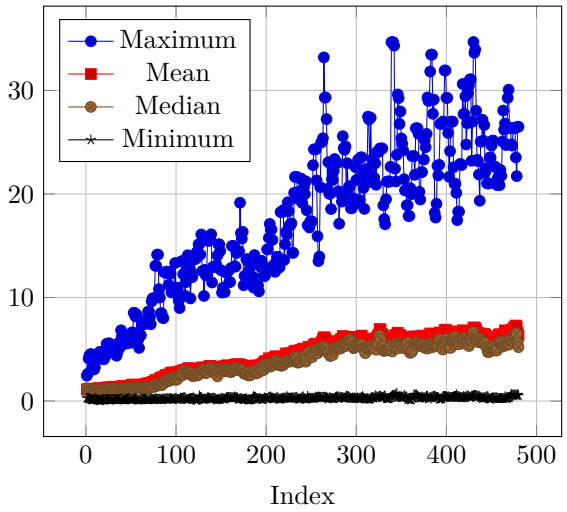
Displacement from Birthplace (Elite Fitness Table)



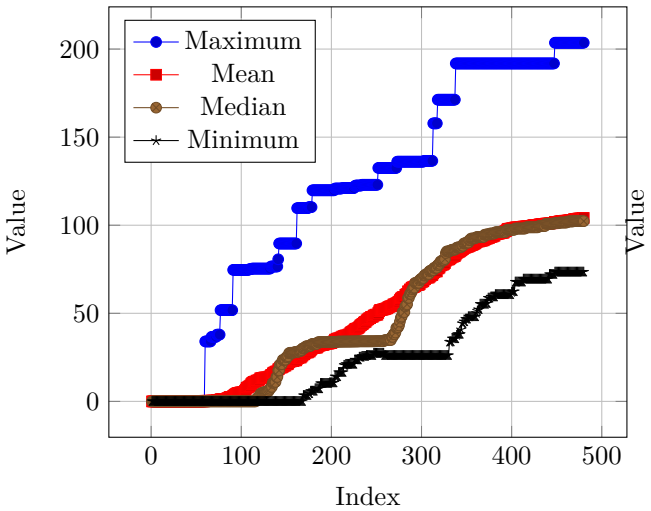
Displacement from Birthplace (Elite Novelty Table)



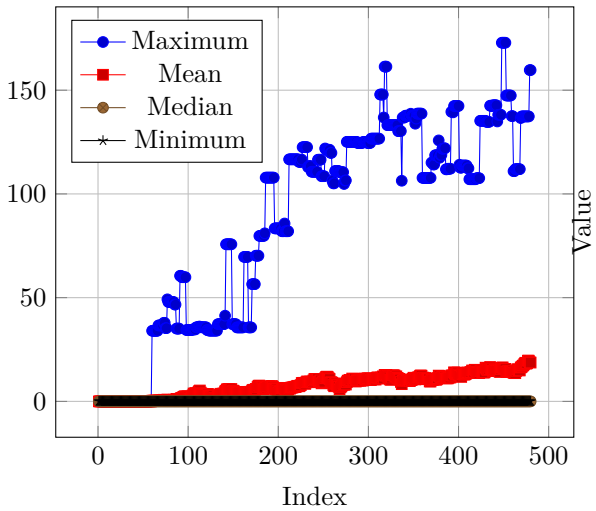
Displacement from Birthplace (Recently Dead Table)



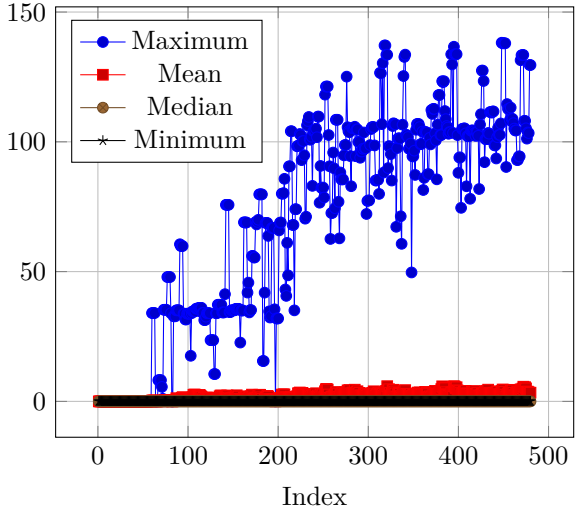
Food Eaten (Elite Fitness Table)



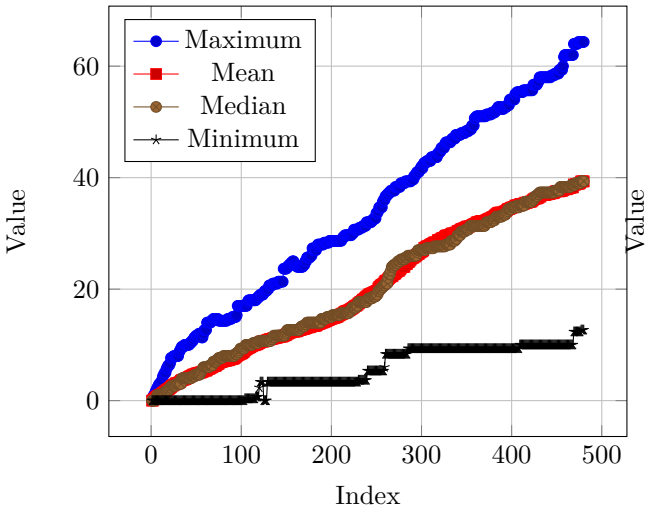
Food Eaten (Elite Novelty Table)



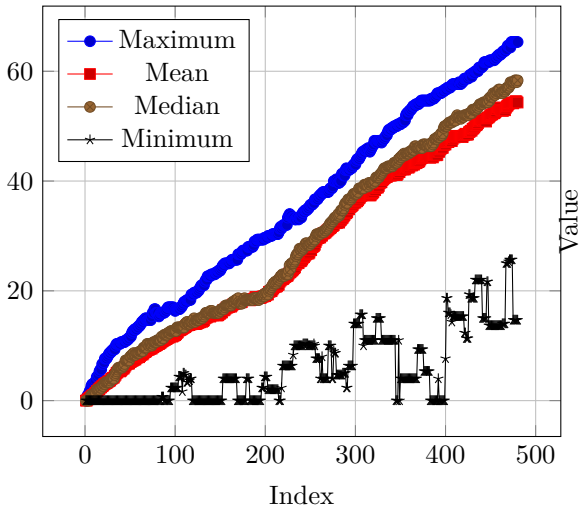
Food Eaten (Recently Dead Table)



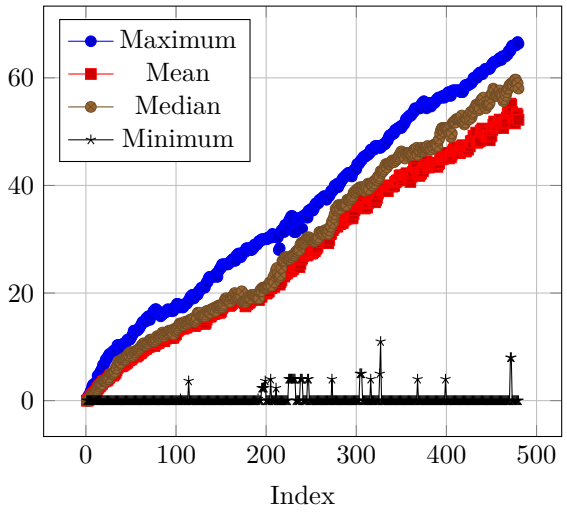
Generation (Elite Fitness Table)



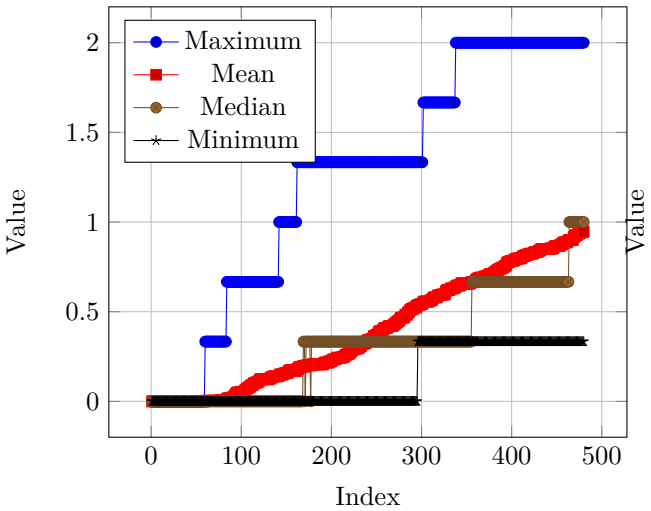
Generation (Elite Novelty Table)



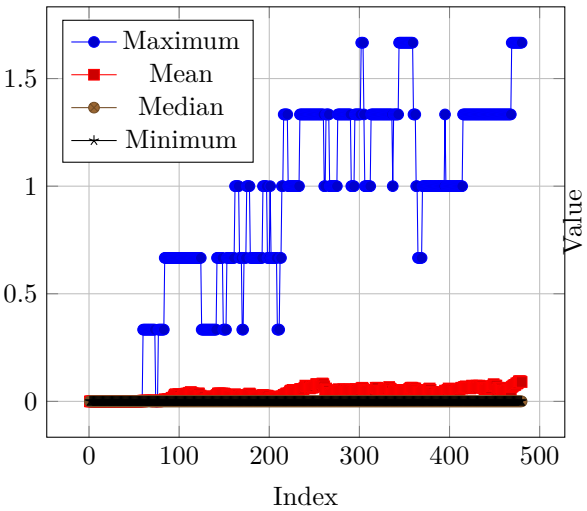
Generation (Recently Dead Table)



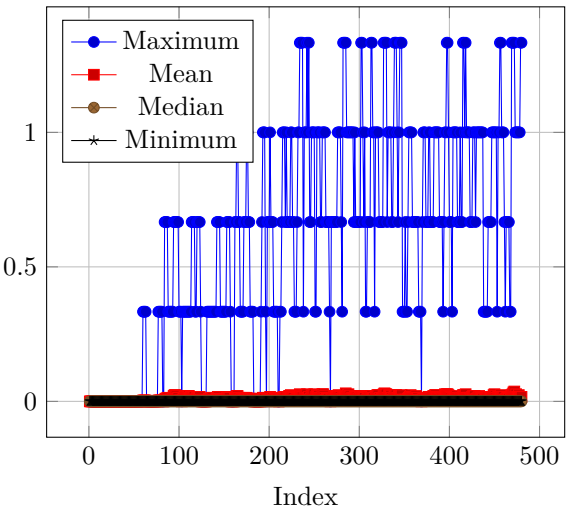
Times Reproduced (Elite Fitness Table)



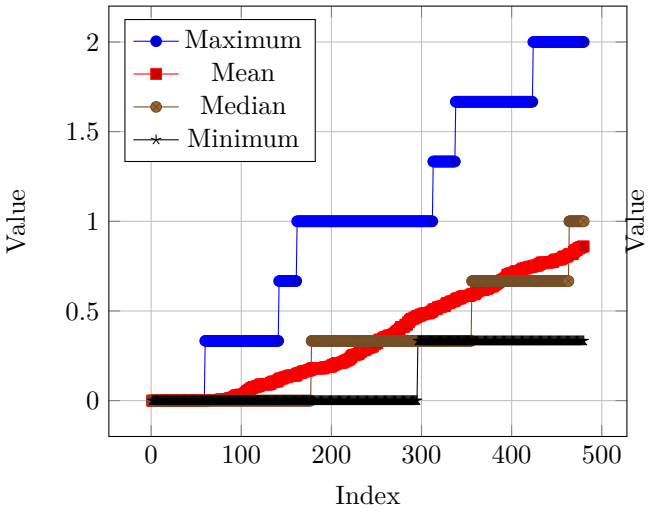
Times Reproduced (Elite Novelty Table)



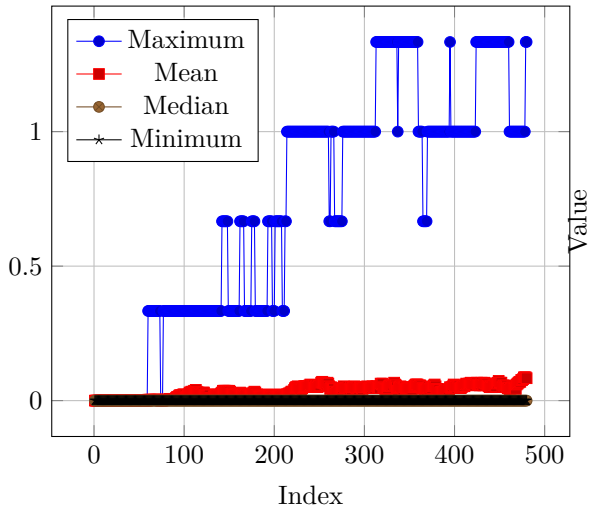
Times Reproduced (Recently Dead Table)



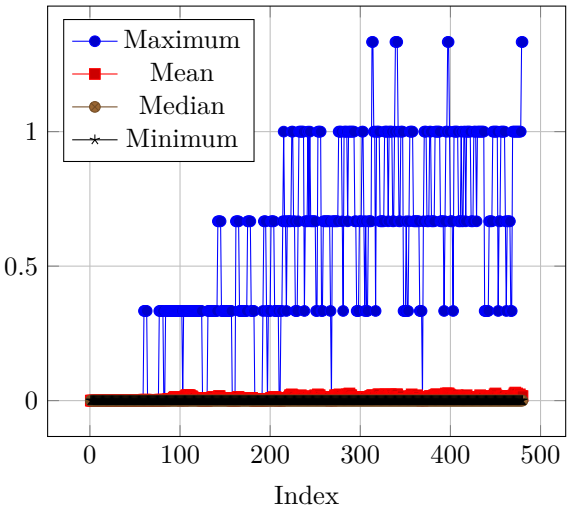
Times Reproduced Asexually (Elite Fitness Table)



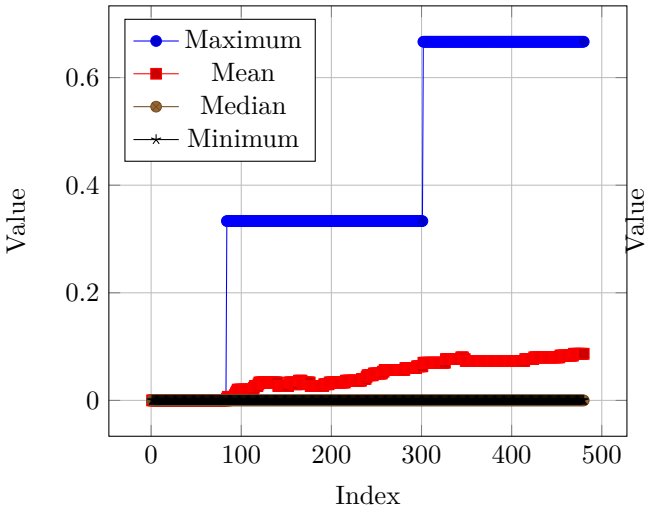
Times Reproduced Asexually (Elite Novelty Table)



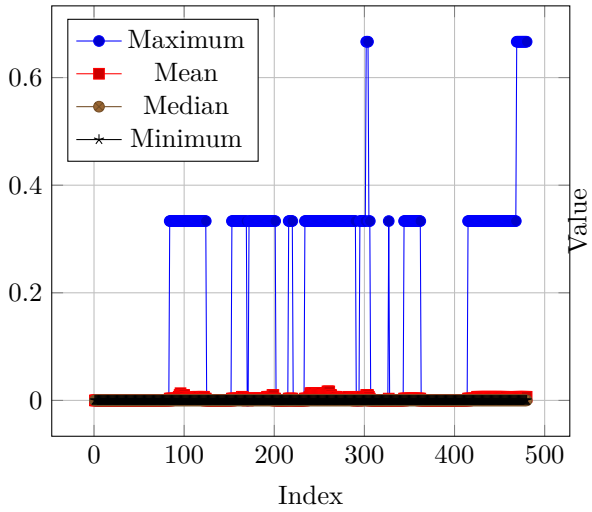
Times Reproduced Asexually (Recently Dead Table)



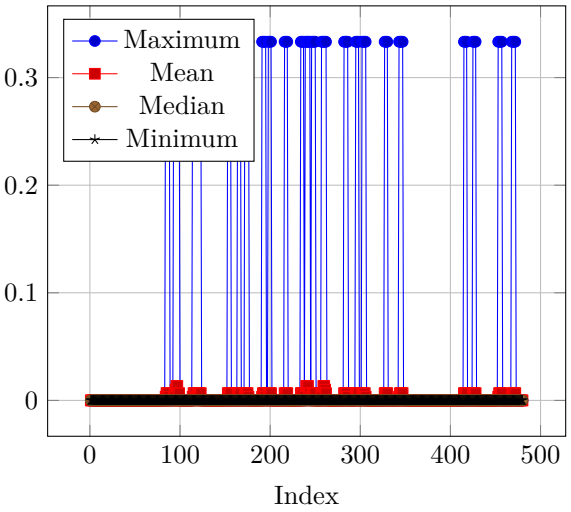
Times Reproduced Sexually (Elite Fitness Table)



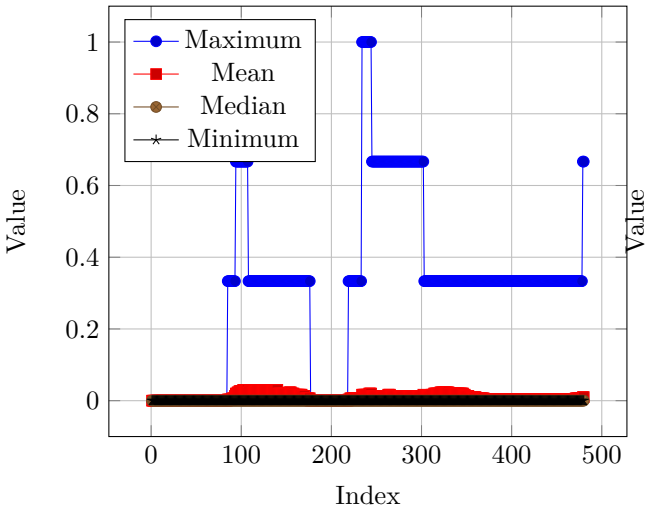
Times Reproduced Sexually (Elite Novelty Table)



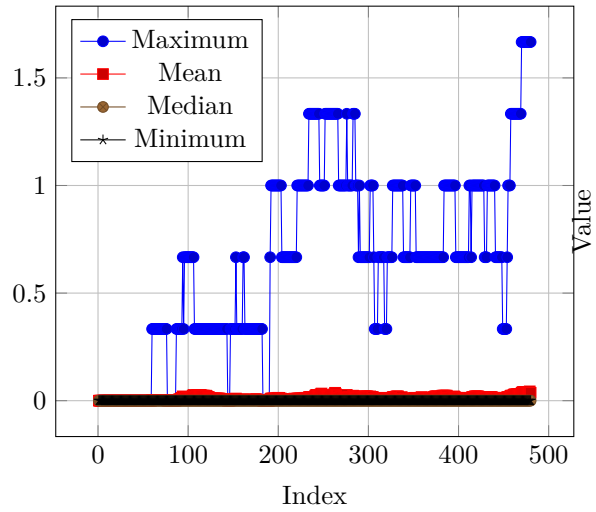
Times Reproduced Sexually (Recently Dead Table)



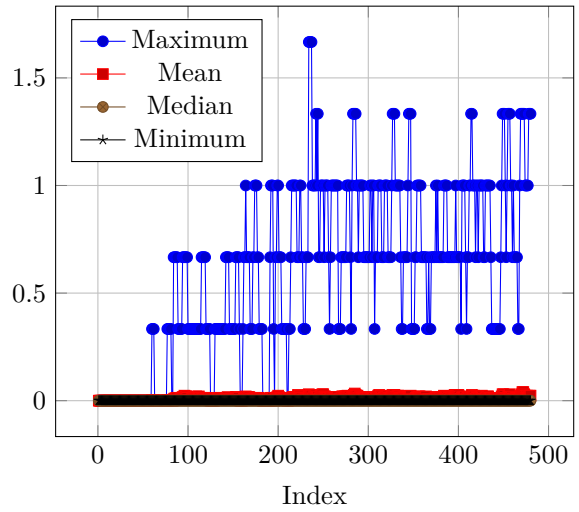
Reproduction Chain (Elite Fitness Table)



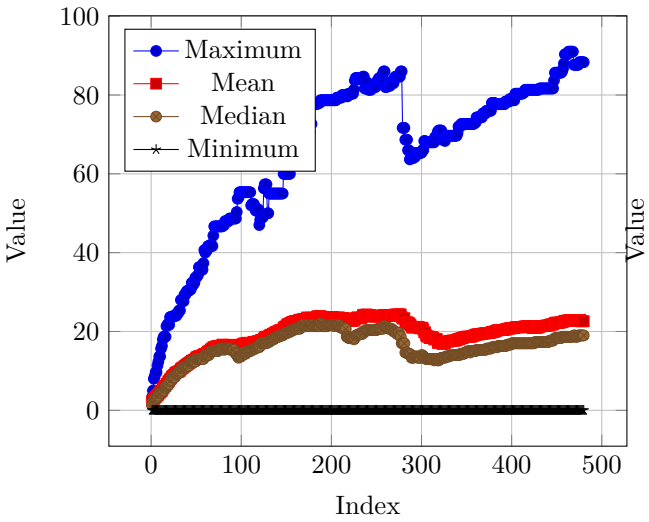
Reproduction Chain (Elite Novelty Table)



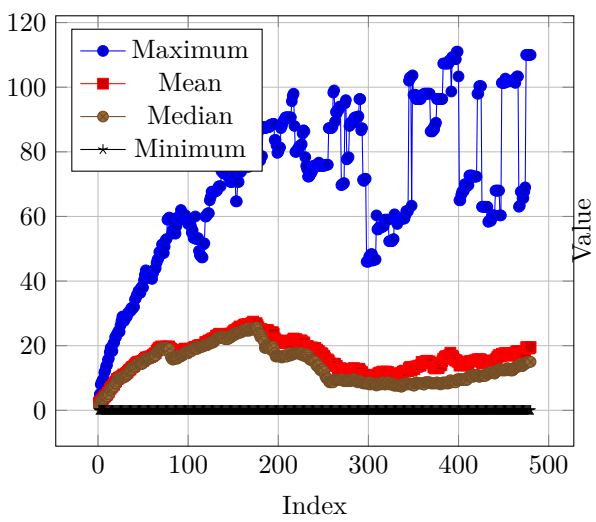
Reproduction Chain (Recently Dead Table)



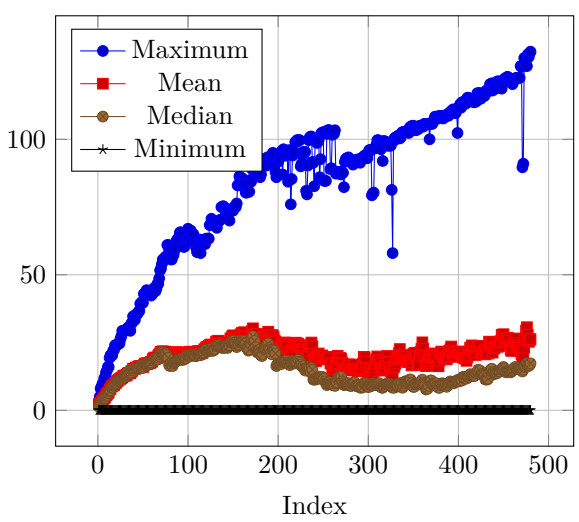
Hamming Distance (Elite Fitness Table)



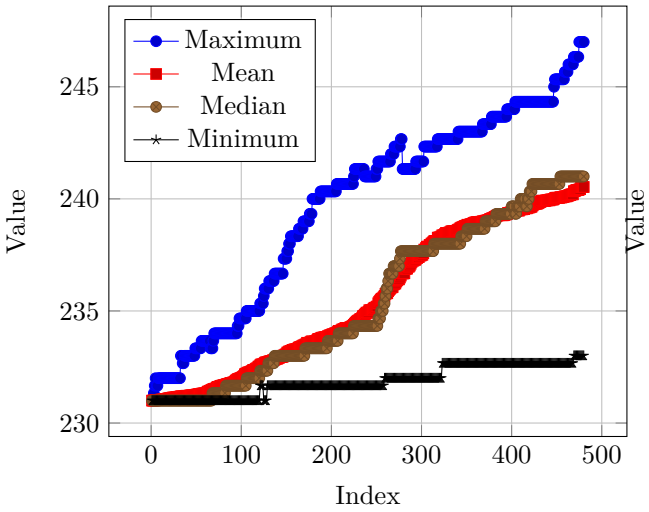
Hamming Distance (Elite Novelty Table)



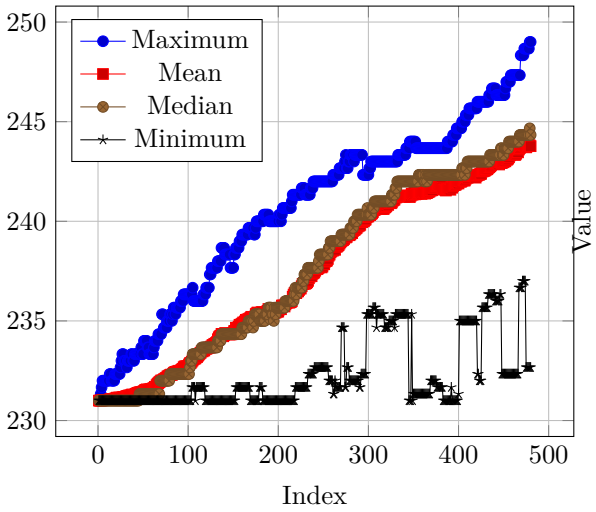
Hamming Distance (Recently Dead Table)



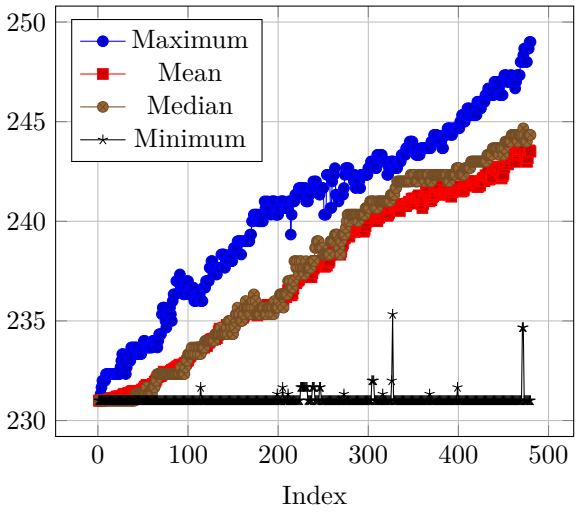
Number of Neurons (Elite Fitness Table)



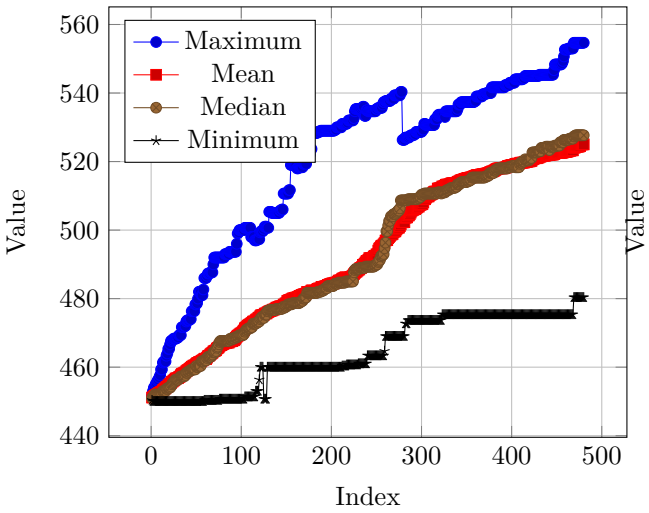
Number of Neurons (Elite Novelty Table)



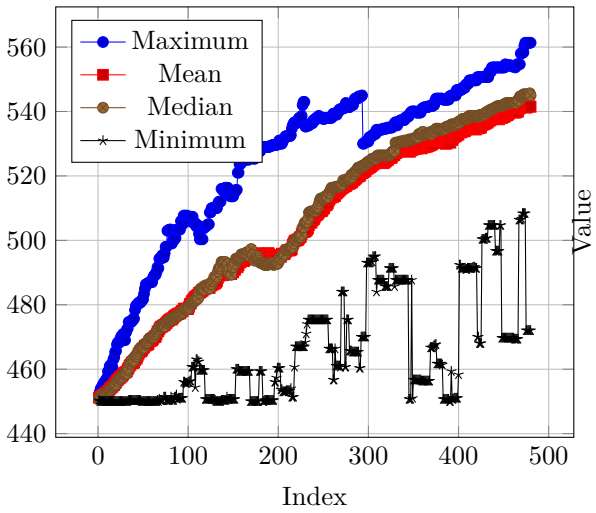
Number of Neurons (Recently Dead Table)



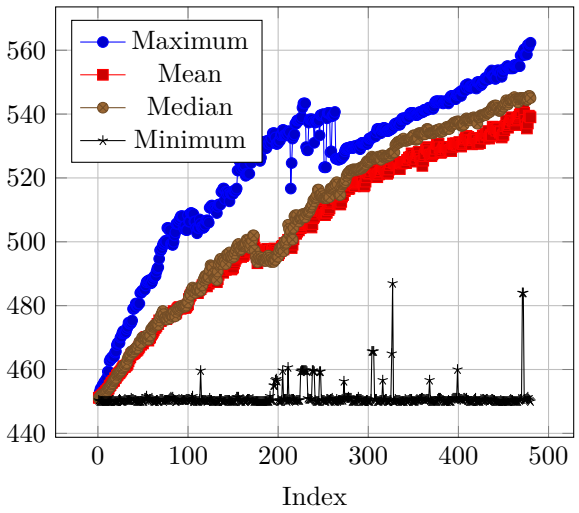
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)



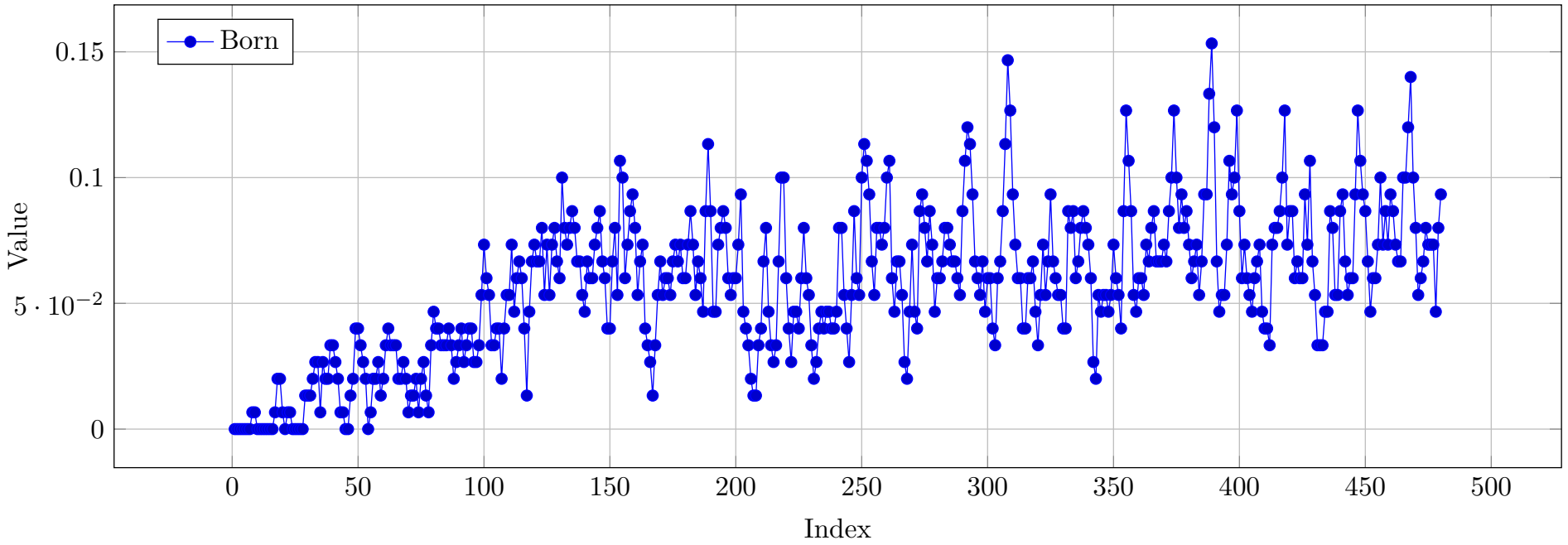
Number of Connections (Recently Dead Table)



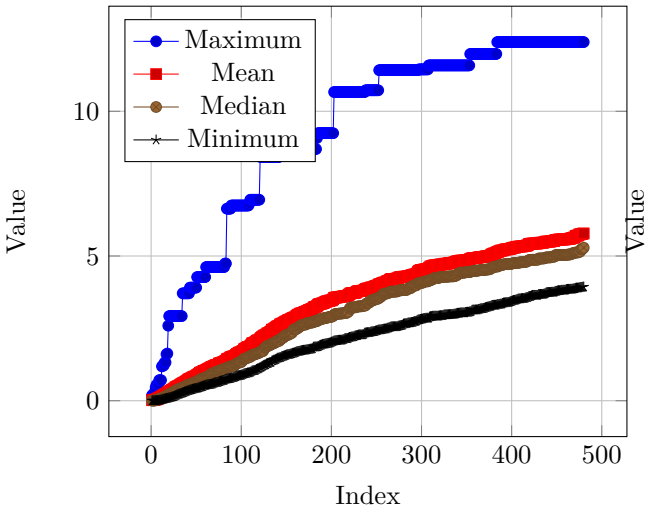
Wheeled Robot - Sum & Squash - No Learning - Interactive Voxel World

RESULTS

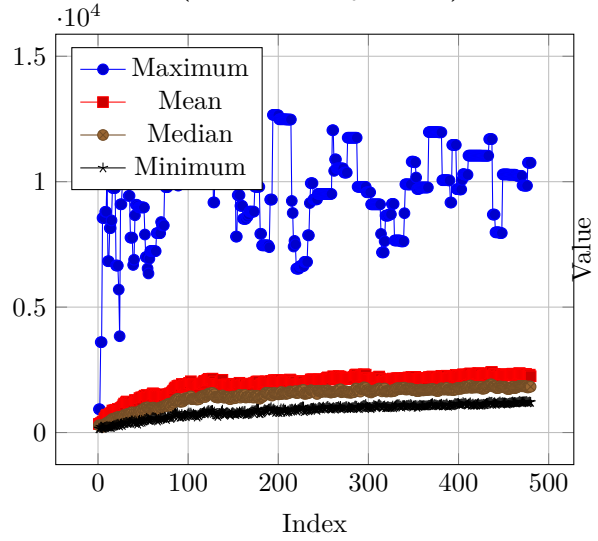
Percent of Minimum Population that was Born



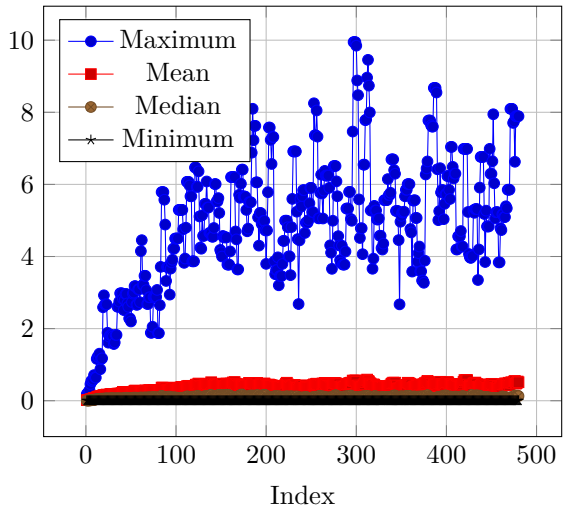
Fitness Score
(Elite Fitness Table)



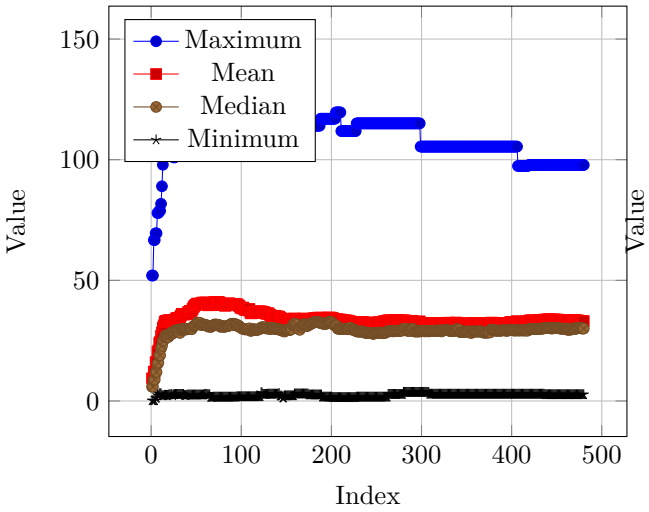
Novelty Score
(Elite Novelty Table)



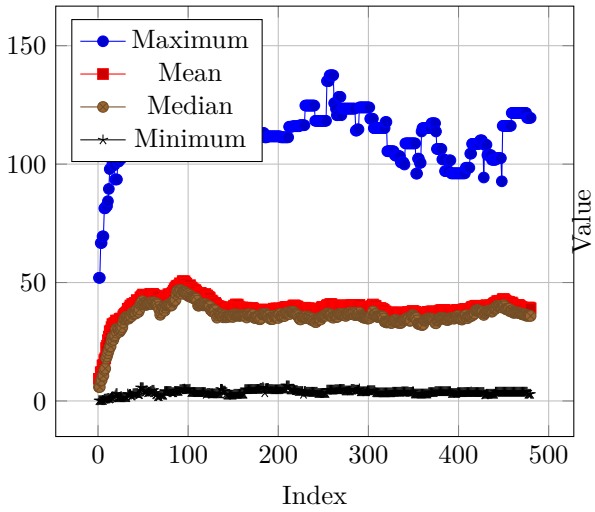
Fitness Score
(Recently Dead Table)



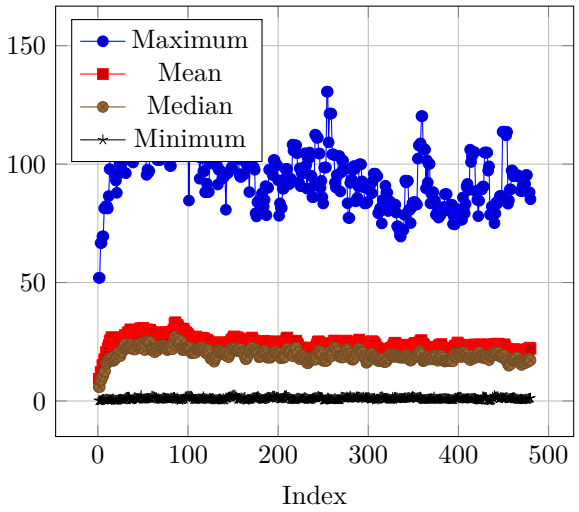
Displacement from Birthplace (Elite Fitness Table)



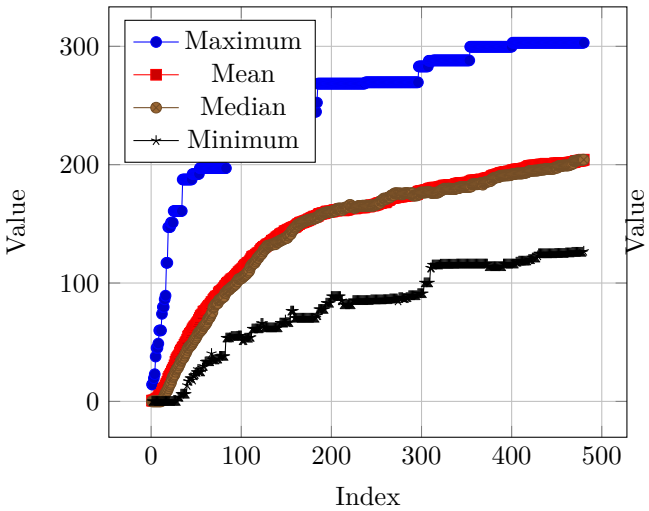
Displacement from Birthplace (Elite Novelty Table)



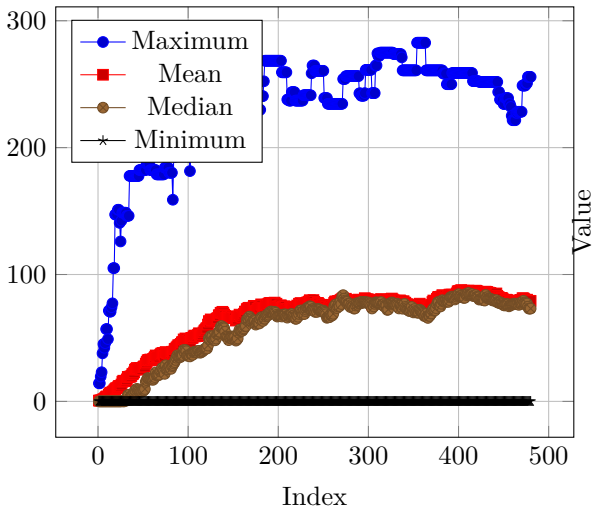
Displacement from Birthplace (Recently Dead Table)



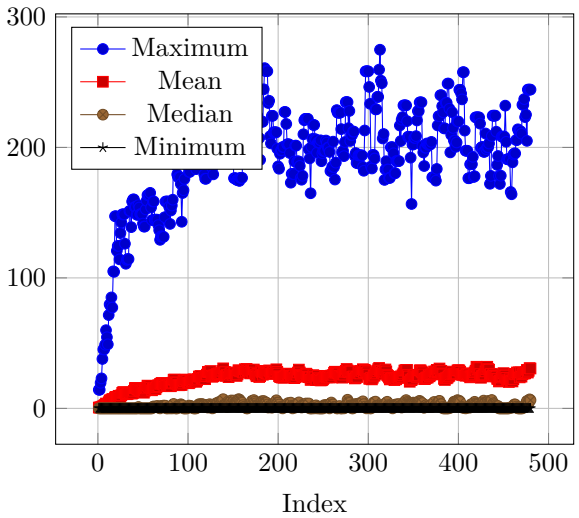
Food Eaten (Elite Fitness Table)



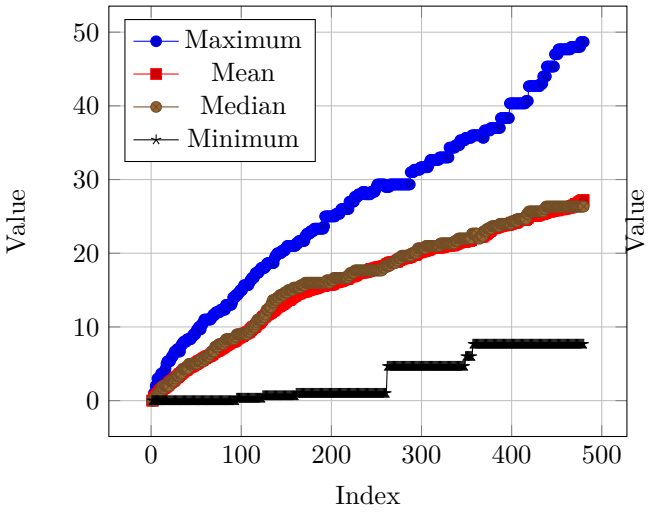
Food Eaten (Elite Novelty Table)



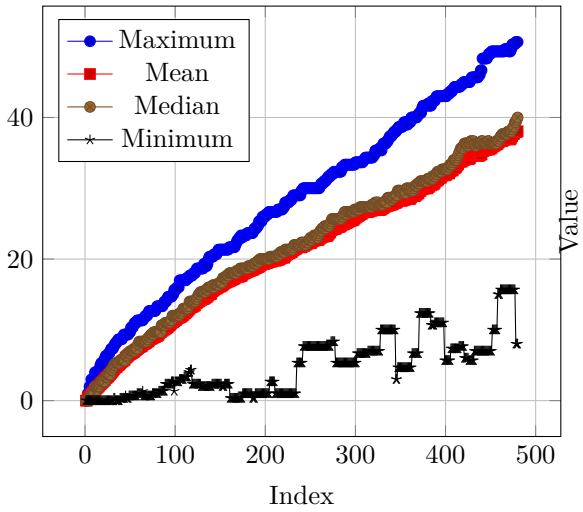
Food Eaten (Recently Dead Table)



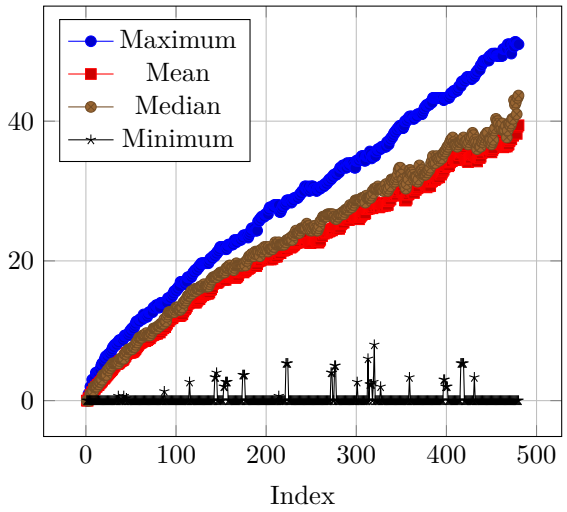
Generation
(Elite Fitness Table)



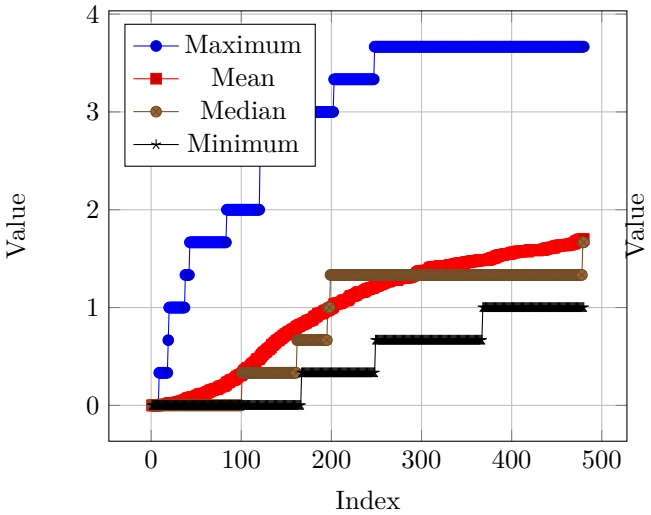
Generation
(Elite Novelty Table)



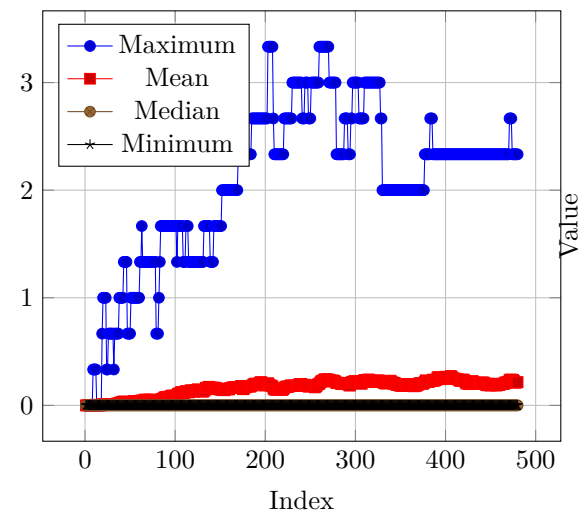
Generation
(Recently Dead Table)



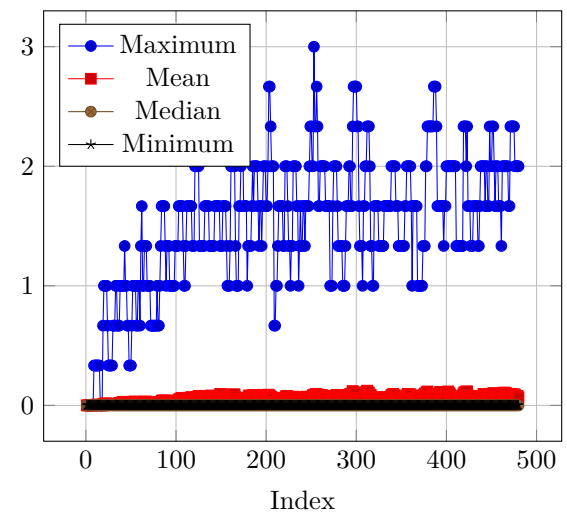
Times Reproduced
(Elite Fitness Table)

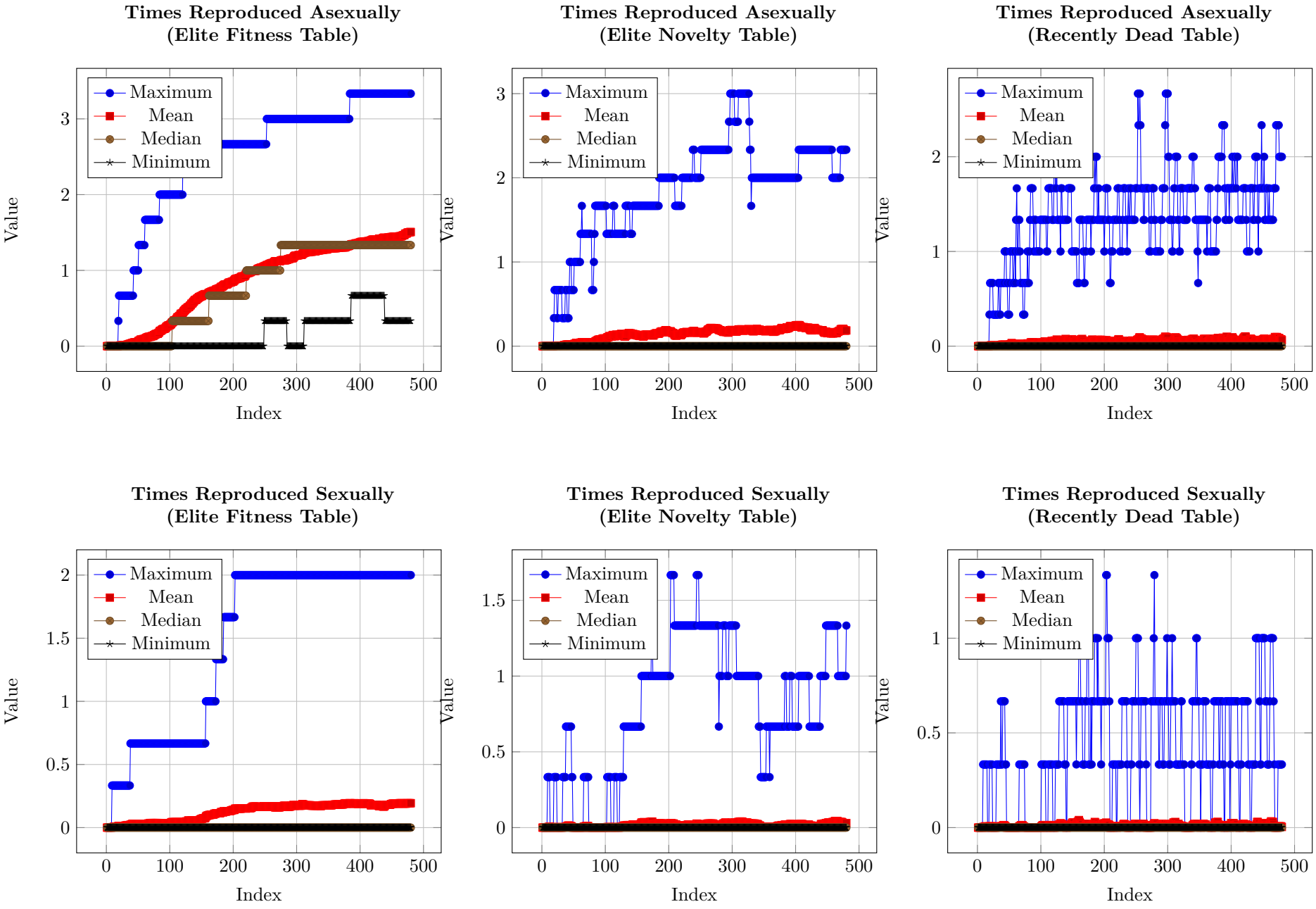


Times Reproduced
(Elite Novelty Table)

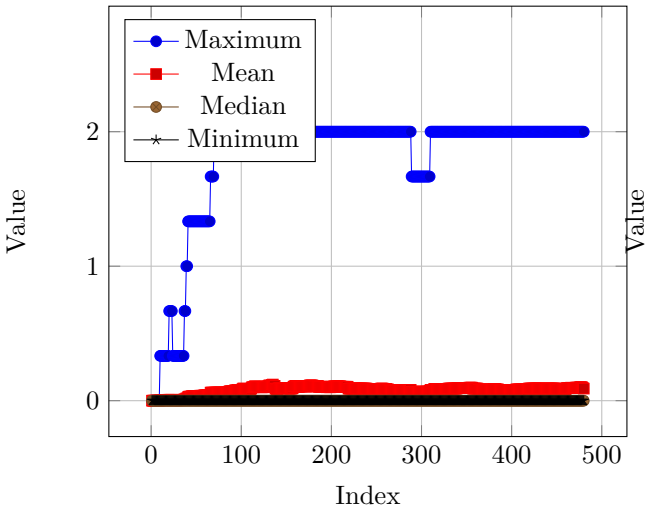


Times Reproduced
(Recently Dead Table)

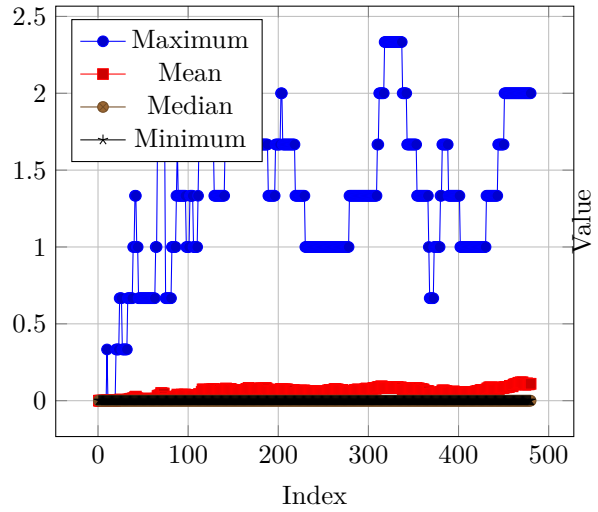



Times Reproduced Asexually
(Elite Fitness Table)
Times Reproduced Asexually
(Elite Novelty Table)
Times Reproduced Asexually
(Recently Dead Table)
Times Reproduced Sexually
(Elite Fitness Table)
Times Reproduced Sexually
(Elite Novelty Table)
Times Reproduced Sexually
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500

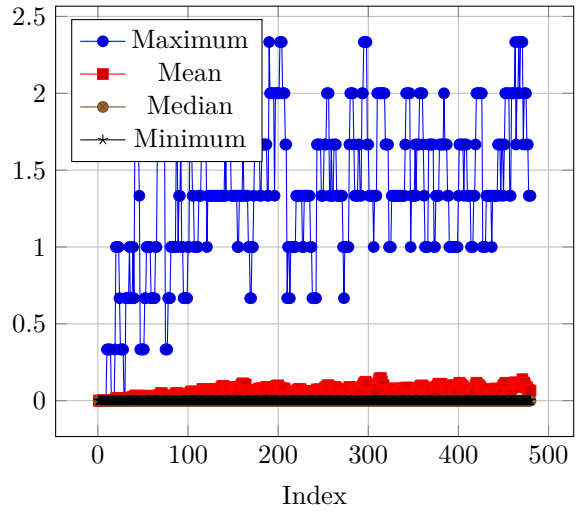
Reproduction Chain (Elite Fitness Table)



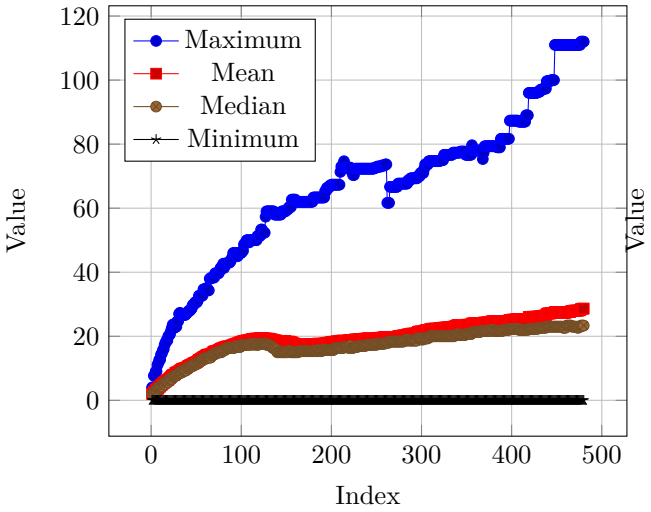
Reproduction Chain (Elite Novelty Table)



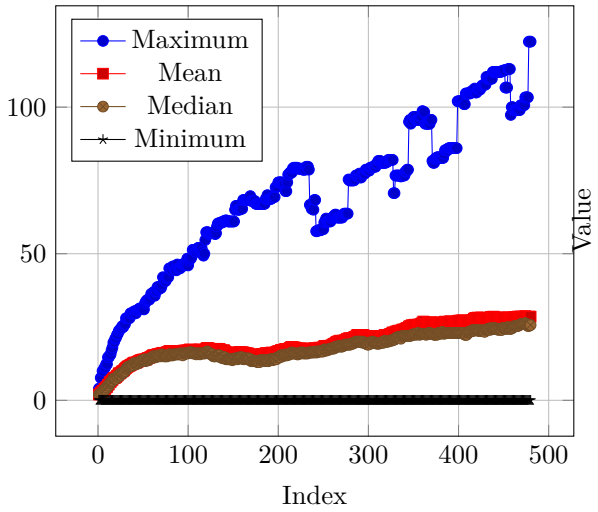
Reproduction Chain (Recently Dead Table)



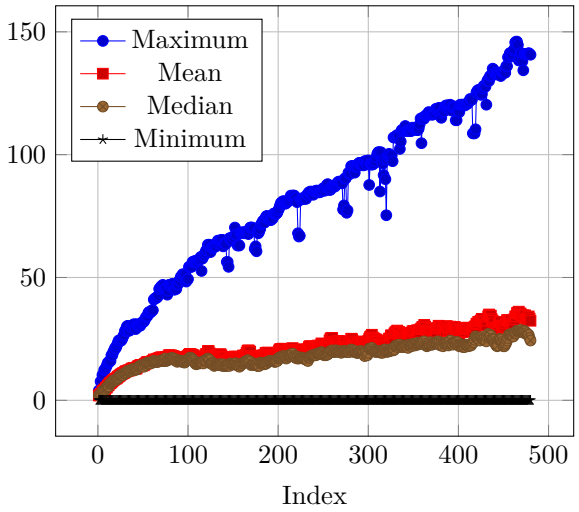
Hamming Distance (Elite Fitness Table)



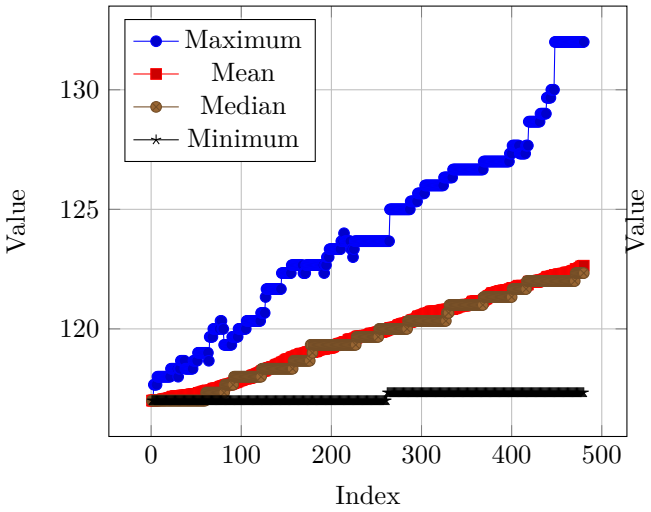
Hamming Distance (Elite Novelty Table)



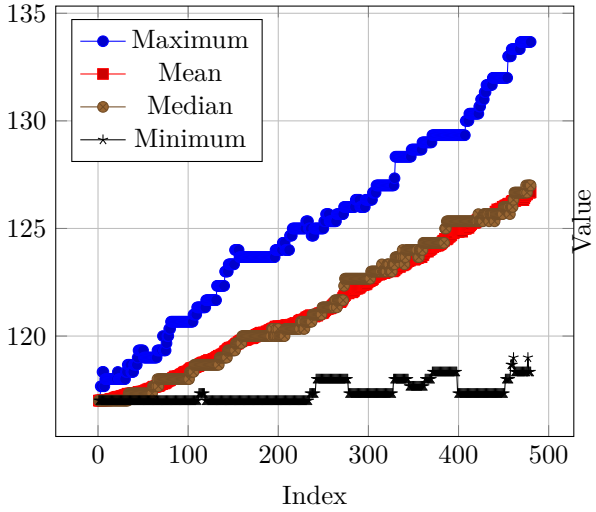
Hamming Distance (Recently Dead Table)



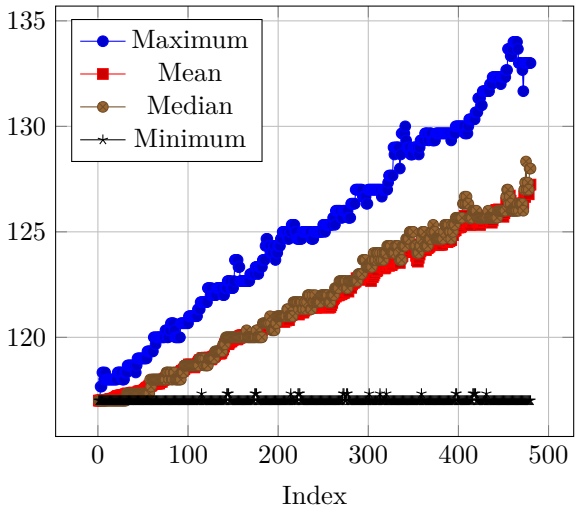
Number of Neurons (Elite Fitness Table)



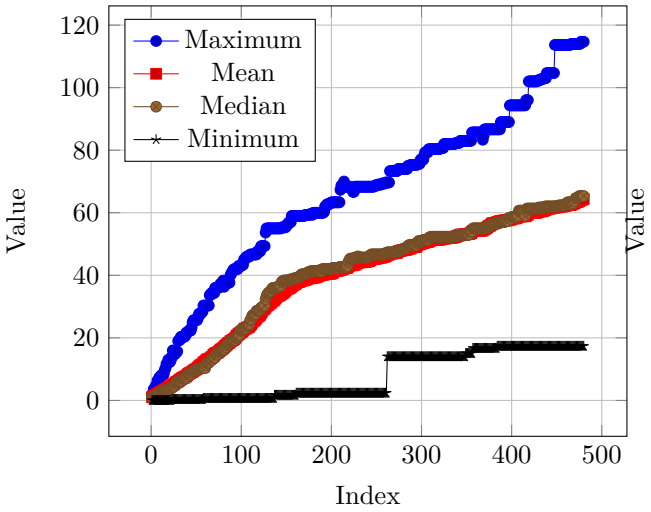
Number of Neurons (Elite Novelty Table)



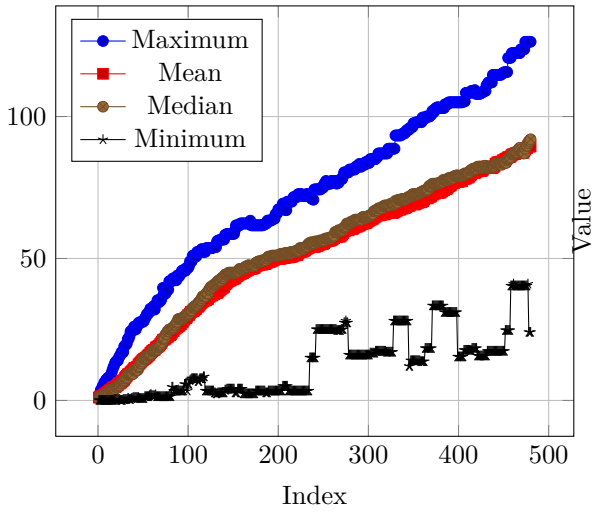
Number of Neurons (Recently Dead Table)



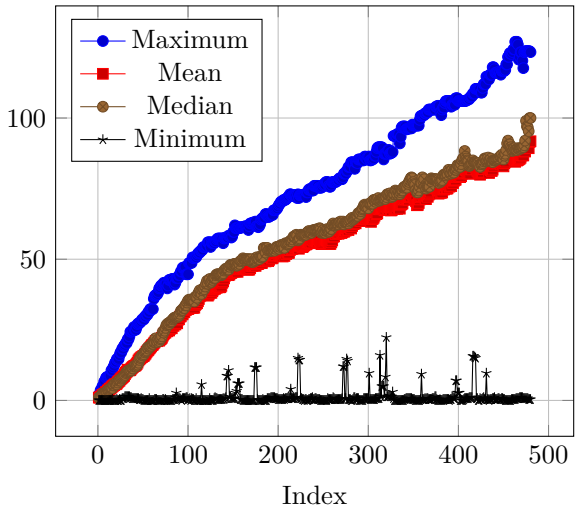
Number of Connections (Elite Fitness Table)



Number of Connections (Elite Novelty Table)

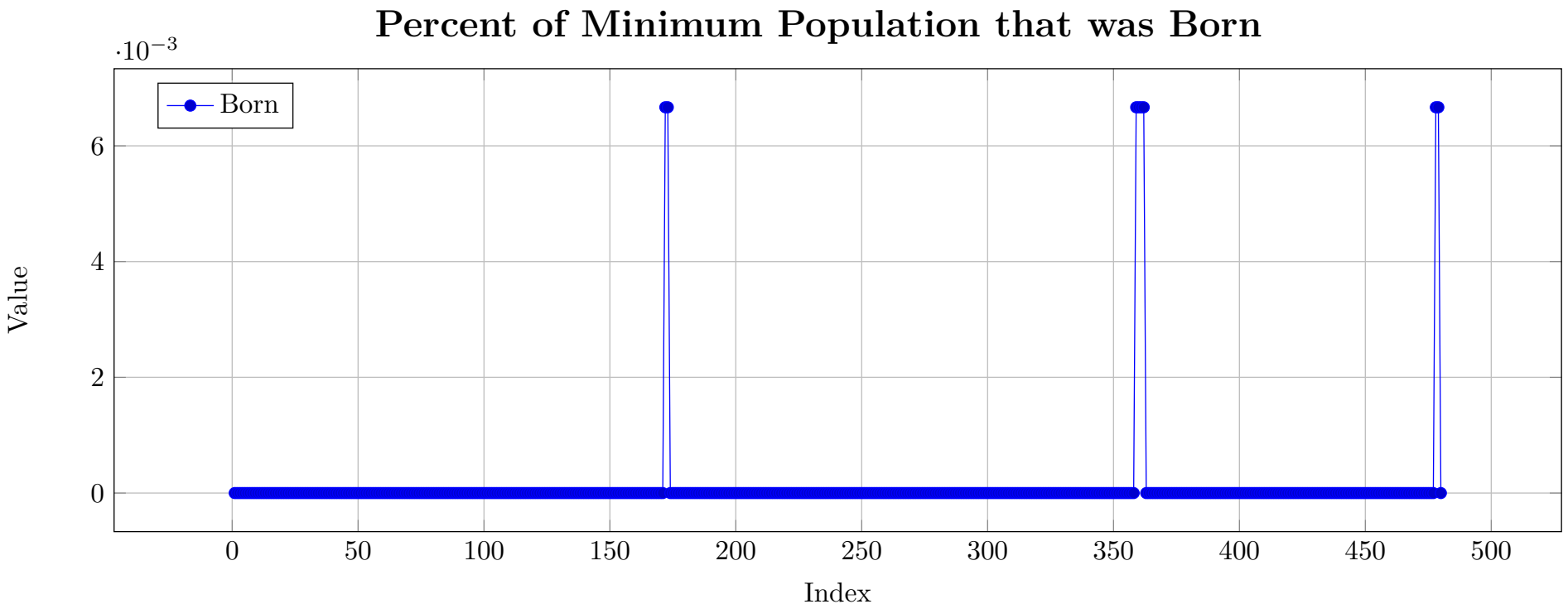


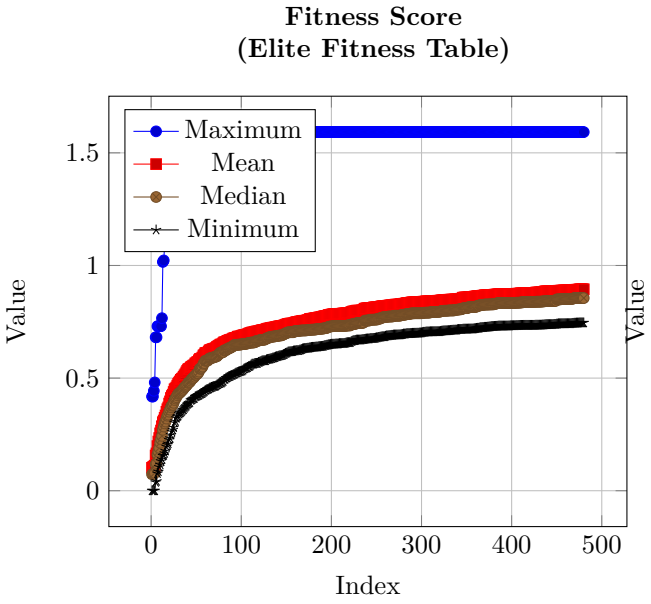
Number of Connections (Recently Dead Table)

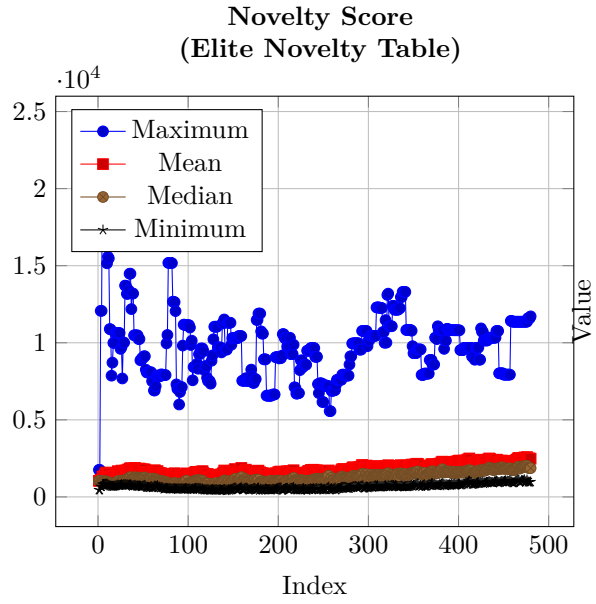


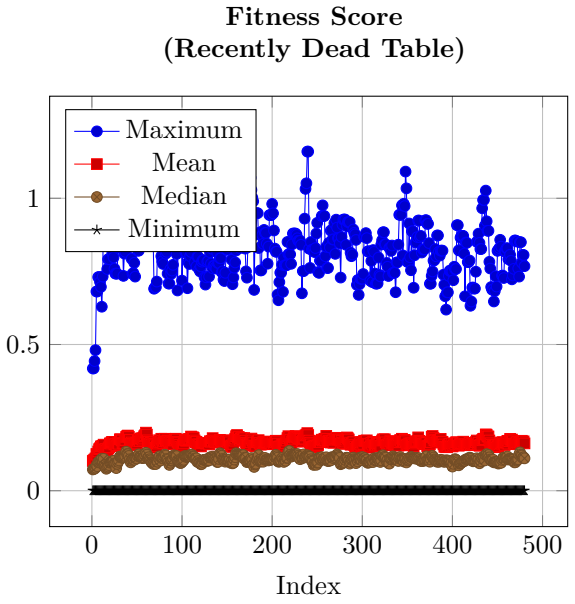
Wheeled Robot - Random - Flat World

RESULTS

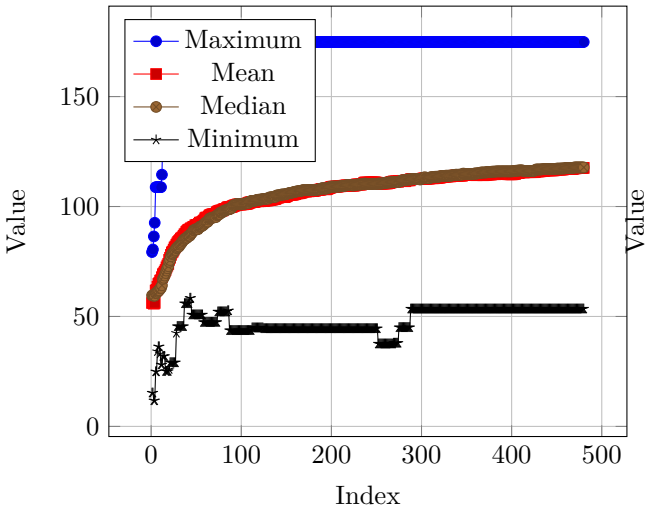

Percent of Minimum Population that was Born
$\cdot 10^{-3}$
Born
Value
Index
0
50
100
150
200
250
300
350
400
450
500
0
2
4
6


Fitness Score
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
0
0.5
1
1.5

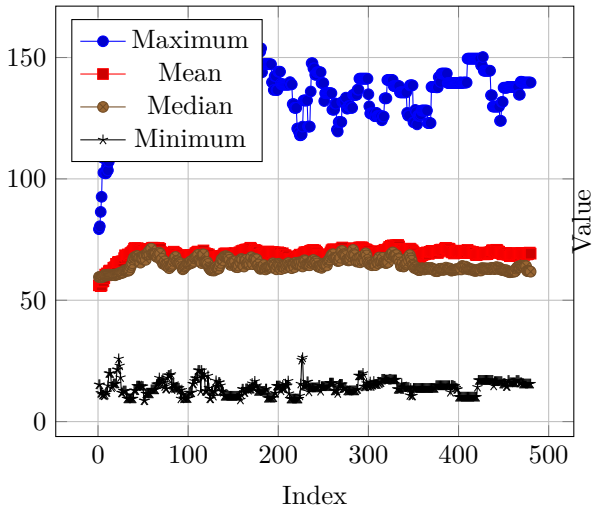

Novelty Score
(Elite Novelty Table)
$\cdot 10^{4}$
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
0
0.5
1
1.5
2
2.5


Fitness Score
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Index
0
100
200
300
400
500
0
0.5
1

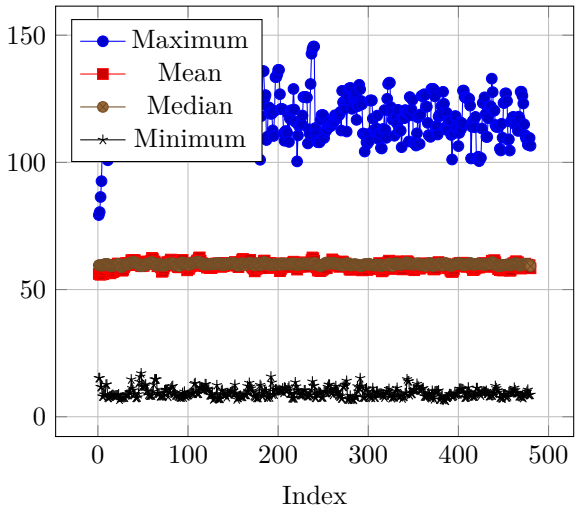
Displacement from Birthplace (Elite Fitness Table)



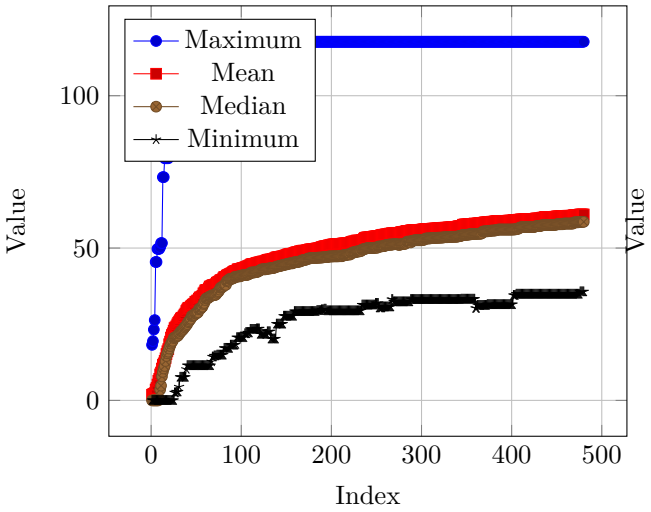
Displacement from Birthplace (Elite Novelty Table)



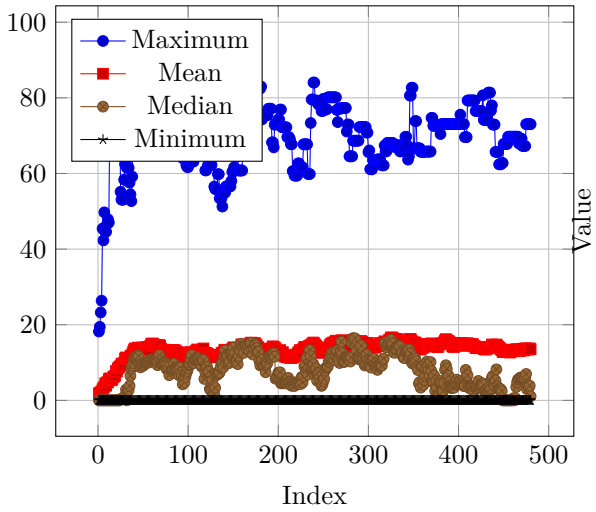
Displacement from Birthplace (Recently Dead Table)



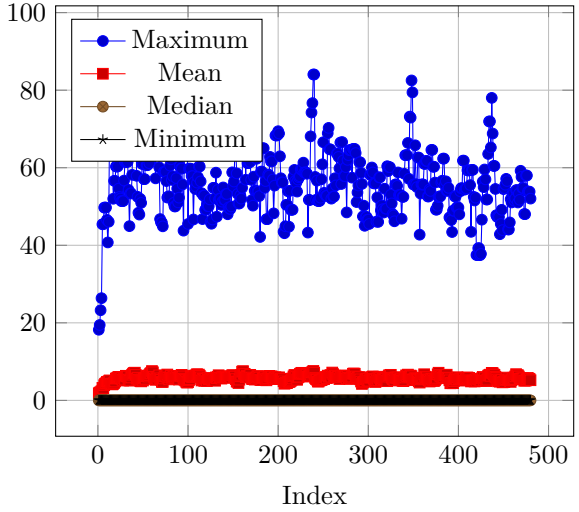
Food Eaten (Elite Fitness Table)

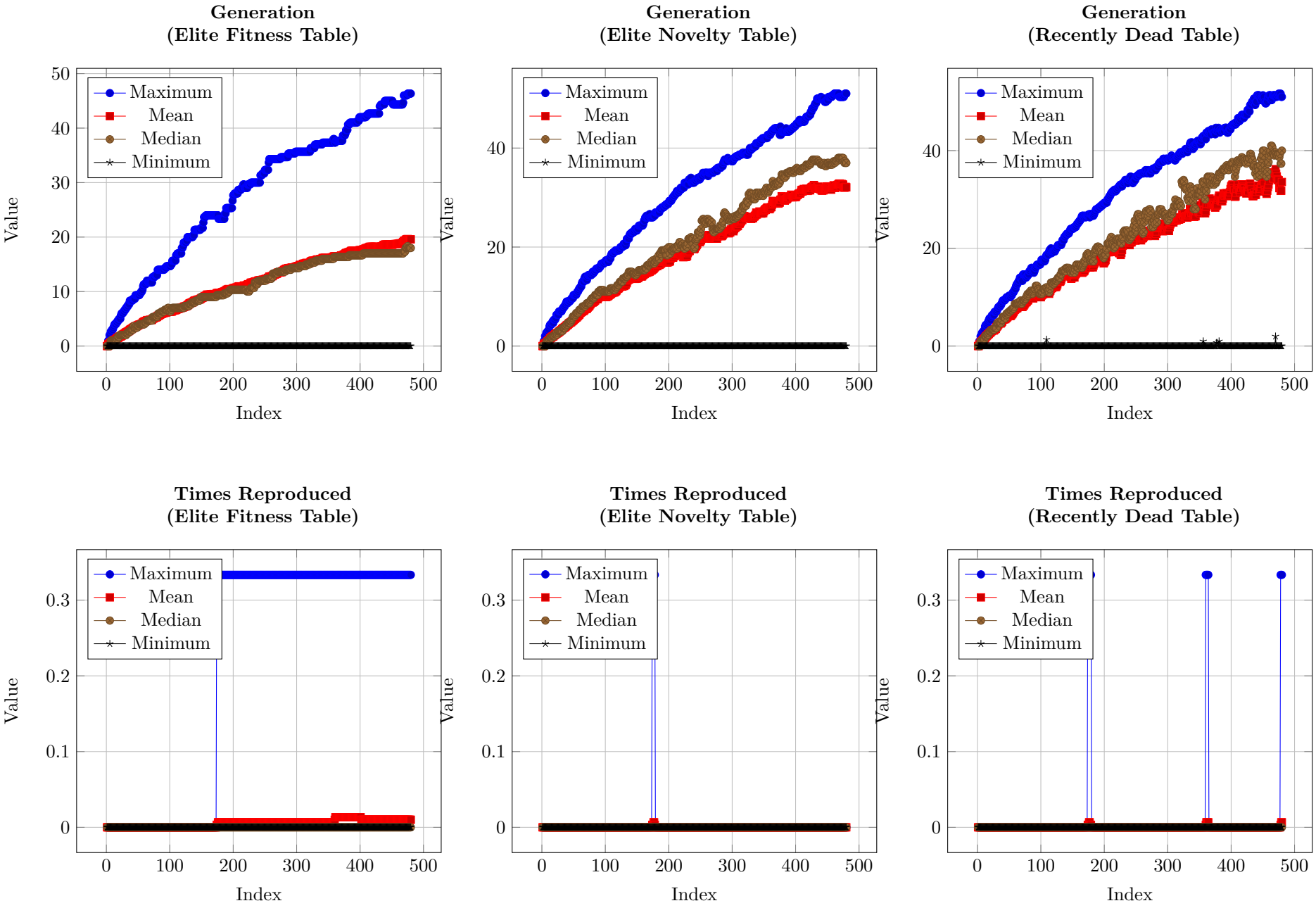


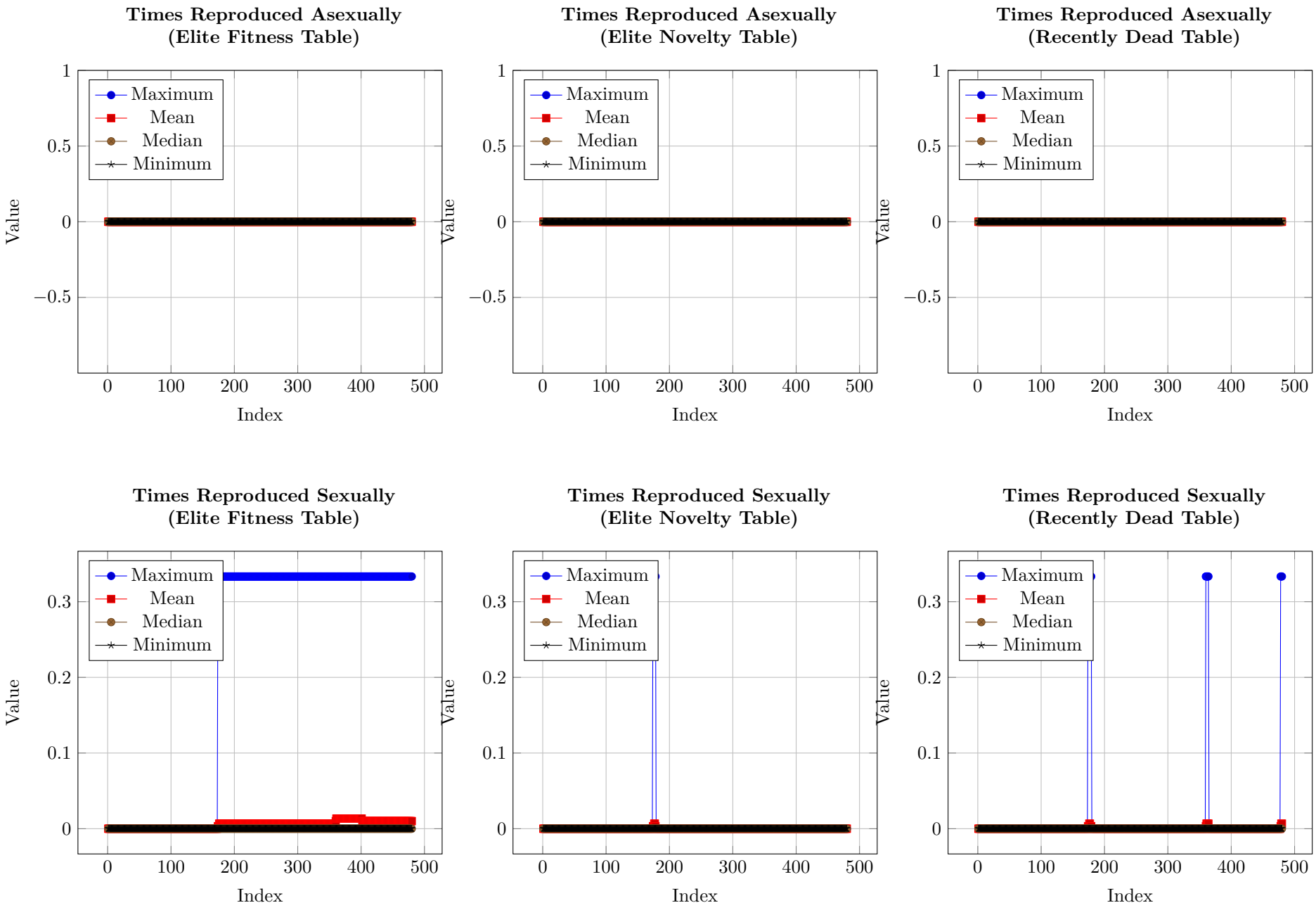
Food Eaten (Elite Novelty Table)



Food Eaten (Recently Dead Table)




Generation
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
Generation
(Elite Novelty Table)
Maximum
Mean
Median
Minimum
Value
Index
Generation
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
Times Reproduced
(Elite Fitness Table)
Maximum
Mean
Median
Minimum
Value
Index
Times Reproduced
(Elite Novelty Table)
Maximum
Mean
Median
Minimum
Value
Index
Times Reproduced
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index


Times Reproduced Asexually
(Elite Fitness Table)
Times Reproduced Asexually
(Elite Novelty Table)
Times Reproduced Asexually
(Recently Dead Table)
Times Reproduced Sexually
(Elite Fitness Table)
Times Reproduced Sexually
(Elite Novelty Table)
Times Reproduced Sexually
(Recently Dead Table)
Maximum
Mean
Median
Minimum
Value
Index
0
100
200
300
400
500
1
0.5
0
−0.5
0.3
0.2
0.1