Swimbots - Final Paper

Phillip’s Fan Club

Sarah Baalbaki

Dylan Estep

William Hsu

Tanxin Qiao

December 2022

Introduction

Evolution and the Genetic Algorithm

The phenomenon of evolution, in which the frequencies of heritable characteristics in populations change over generations, impacts all life and manifests at various scales, from the biomolecular to the ecological. Accordingly, in addition to fundamentally upending humankind's conception of ourselves and our relationship to all other living things, our understanding of evolution has profoundly informed all fields of the life sciences in the past few centuries. Its consequences – in medicine, that populations of pathogens can evolve resistance to treatments; in agriculture, that artificial selection of favorable traits among a domesticated crop can dramatically transform its lineage; among innumerable others – are pervasive and unignorable to the modern biologist.

In addition to its transformative impact on biology, our understanding of the basic mechanisms of evolution have also inspired computational methods to a large degree. Among the more famous of these methods is the genetic algorithm (GA) that rose to prominence in the 1970s. This algorithm was formulated as a broadly applicable computational strategy to solving well-defined optimization problems through successive improvement of candidate solutions over many generations in a manner analogous to the way a living population adapts to its environment. Descriptions of the GA tend to draw clear parallels between its steps and the modification, reassortment, and reproduction of genes – the building blocks of heredity – that occur in biology, particularly in the context of natural selection.

In his well-known text on the subject, the engineer David Goldberg summarized the GA as follows: "In every generation, a new set of artificial creatures (strings) is created using bits and pieces of the fittest of the old; an occasional new part is tried for good measure."[1] The "creatures" in Goldberg's description are potential solutions to the optimization problem encoded in strings, each of which has a score indicative of its quality obtained through a defined fitness function. Solutions are selected randomly, in proportion to their fitness score, to "reproduce" for the next generation, with pairs of solutions exchanging contiguous segments of their values up to specific points, in a manner mirroring the chromosomal crossover that occurs during the production of gametes in sexually reproducing organisms. Modifications to values within the "offspring" solutions can also be made randomly at low probability, similarly to the way mutation manifests in actual genes. For a successful GA, the fitness of the solution population increases as generations elapse, and the algorithm can be terminated when some condition is met, for example when no further improvement in fitness is observed or simply when the top-scoring solution has achieved a certain threshold fitness.

Swimbots and Other GA-Based Artificial Life Simulations

Given the strength of the parallels between the workings of the GA and the underlying mechanisms of biological evolution, it is hardly a surprise that computational simulations of living systems have themselves heavily utilized the GA. In the early 1990s, the digital artist

Jeffrey Ventrella, then a graduate student at the MIT Media Lab, created a series of GA-driven artificial life simulations with the goal of facilitating the emergence of life-like physical forms and behaviors, particularly conducive to efficient locomotion, in rudimentary artificial beings.[2] In the first of these, simple three-dimensional bipedal creatures that he called walkers could modulate the motion of their legs based on parameters encoded in genes; the walkers which could successfully traverse the farthest distance toward a predefined goal in a given period of time scored highest in fitness, and their genes would serve as the basis for the next generation.[3]

In subsequent simulations of this type, the dimension of the simulated creatures and their environment was reduced to two, with fewer constraints on the creatures' morphology. In these simulations, rather than directing its own motion toward arbitrary predefined goals, a creature could pursue another individual as its goal and directly "mate" with it sexually to produce offspring, thus departing with the conventional GA conception of generations advancing in a synchronous, lock-step manner, a feature inspired by a hope Richard Dawkins had expressed for artificial life simulations in one of his books.[4] The artificial creatures, which Ventrella eventually called *swimbots*, consisted of segments of varying dimensions linked to one another – in the simulation's first iteration, all through a central node, and in subsequent versions end-to-end with an underlying tree-like structure.[5, 6] Each swimbot had an internal energy value and could pursue a goal within its field of vision – either one of the food bits scattered through the environment, or another swimbot – based on its state. Swimbots with genetically-encoded morphologies and motion parameters favorable to efficient pursuit of their chosen goal were those whose genes would be used to produce offspring, obviating the need for an explicitly defined fitness function.

Ventrella presented his work on swimbot and related simulations at several artificial life conferences throughout the 1990s, and in 1996 one became the basis for an intended commercial product, called *Darwin Pond*, by his eventual employer, the computer game studio Rocket Science Games.[7] The studio's prior releases struggled to find a sizeable audience and the developer ultimately went out of business in 1997, before *Darwin Pond* could be released. Nevertheless, Ventrella continued work on swimbot simulations, continually updating them and making them accessible via internet browser onward through the late 1990s and 2000s, with the latest iteration, GenePool, still available today.[8]

The Computational Problem: Designing Our Own Swimbot Simulation

In the ensuing paper, we will describe the design of our own swimbot simulation. Simulating an evolving artificial life system requires three main components:

1. The **environment** in which the living things exist
2. A **representation** of the living things themselves, including underlying data fields and structures
3. A series of **rules** delineating the living things' interactions with their environment and one another, along with consequent changes in their internal states

As in Ventrella's simulations, swimbots in our own simulation could move through a two-dimensional environment to pursue their goals. We represented swimbots as objects with fields for motion, including their position in the environment, their internal energy, and the genes that wholly govern the structure and connectivity of their body segments as well as motion parameters. Genes were represented in their own structures, with float (and some integer) values encoding the swimbots' traits. The rules driving swimbot behavior are encoded in the functions called during the execution of each timestep. Further details of all these elements are given below.

For a successful swimbot simulation, we hoped to observe, in addition to emergent behavior among the swimbots in the visually rendered system, changes in the frequencies of swimbots' traits as the simulation progresses, with an enhancement in frequencies of traits advantageous to swimbot motion. Thus, we designed additional functions to track these frequencies at any given timestep, and to export the data for visualization.

Algorithms

Simulation Overview and Initialization

As with other simulations developed over the course of the semester, we simulate our system by creating an initial environment – in this case, a Pond object – containing the appropriate number of swimbots and food bit objects, each assigned a random position within a central square of the pond. Swimbots are created with random genomes, with their segments assembled based on the genes' values, and assigned a random initial direction as their velocity. Each successive "generation" (timestep) of the simulation is then obtained by copying the prior pond configuration and updating it: each swimbot queries its internal state to retain its prior goal – either a food bit or another swimbot – or select a new one, attempts to move toward the goal, then takes further action if it is within a critical threshold distance of its goal. Swimbots gain energy from eating food bits and lose it from moving or mating; if their energy is depleted or they reach a maximum allowable age, they die and are removed from the system. As the simulation advances, additional food bits are added to the environment with frequency defined in a global parameter.

When a swimbot is constructed for the first time, whether during the simulation's initialization or during the birth of offspring, its constituent segments must be assembled. The first segment encoded in its genes is designated the main segment, which derives its position and angle fields from the fields of the swimbot as a whole. Remaining segments, depending on the number of segments encoded in the swimbot's genome, are attached to the building swimbot one at a time, with the segment of attachment determined in a random manner. More specifically, when the segment is added, it can be added as a subsegment of any of the attached segments on the swimbot. Consequently, swimbots' segment connectivity is tree-like in nature. Because the Swimbot object encodes its main segment, and hence all its subsequent segments, through pointers, the structure of the segment tree must be copied in a recursive manner in later generations to maintain proper segment connectivity in the copied swimbot without modifying the configuration from prior timesteps. When we update the position of the swimbot, we have to update the position of the subsequent segments as well.

Swimbot Goal Determination

Several steps are required for a swimbot to decide whether to retain or update its goal in each timestep. The goal is ultimately determined by two factors: the swimbot's internal energy state and what is visible to it, with swimbots capable of seeing all objects within a defined radius of their own position. For example, if a swimbot's prior goal was another swimbot, but it is now hungry due to its internal energy being below the hunger threshold, it will need to determine a new goal: a food bit. When the swimbot's prior goal *does* match its current energy state, it nonetheless needs to check if the previous goal is still valid – that it has not been removed from the system due to having been eaten (for food bits), having died (for other swimbots), or simply no longer being within the swimbot's field of vision. If the goal remains valid, it is retained and the swimbot will ultimately orient its motion toward the same goal.

If the goal is invalid, then the swimbot determines a new goal according to its energy level. If the energy level is below the hunger threshold, it decides to pursue a food bit. It will choose the closest food bit to itself and set that as its goal. If the swimbot is not hungry, it will be characterized as pursuing mates and will thus choose a specific mate within its field of vision to pursue. We initially decided to implement choosing a mate randomly, but wanted to also study how our system would be affected if we have specific preferences or desired characteristics. So, we implemented multiple options to determine mate preference, equally applicable to all swimbots: preference for random swimbots, for swimbots with small or large numbers of body segments, for swimbots with faster translational motion parameters, for swimbots with a similar number of segments to itself, and for swimbots with a similar length of primary segment to itself. Mate choice is constrained in that swimbots will not choose one of its own children or parents as a goal. If no suitable goal is found within its field of vision, it will simply have no goal for that timestep, and its motion will be oriented in the same direction as it had been in the prior timestep.

As swimbots move towards their target, they lose energy. Swimbots lose energy based on their speed and their mass, where swimbots that are faster, and ones that are heavier, lose more energy than those slower or smaller than them.

Taking Action: Eating or Mating

If, following their motion during a timestep, swimbots find themselves within the critical distance of their goals, they will need to take action accordingly. We thus implemented a function called EatOrMate, a method of the Pond object, that performs these checks for all swimbots and executes the associated actions. In the case that a swimbot's goal is a food bit, the swimbot eats the food bit, gains its associated energy, and the food bit is removed from the pond. In the case that the goal is another swimbot, the swimbots mate, lose energy, and generate an offspring that is created in the pond nearby. Mating is constrained such that no swimbot can mate more than once per timestep.

When an offspring swimbot is created, we execute a genetic recombination operation to ensure the offspring receives a combination of genes from both parents. For the swimbot's "common genes" (the three defining traits applicable to the entire swimbot), the value of each is randomly selected from either parent. These encode the swimbot's number of total body segments, plus translational and rotational movement parameters that define and constrain their movement capabilities. The method then proceeds to the offspring's eight segment genes, corresponding to the maximum number of segments any swimbot can possess. (The number of segments "expressed" when constructing the swimbot is defined by the associated common gene, but gene values for the remaining "latent" segments are nonetheless retained.) For each segment, these genes specify the color, width, and length of the segment, as well as the angle from the segment to the one it's attached to. For each of the eight segments, a one-point genetic crossover is performed: gene values up to the randomly selected crossover point are derived from one randomly selected parent, with the remainder derived from the other. The offspring

swimbot is then built from the resultant genome and added to the pond. At this time, the “family” fields of the swimbots are modified to prevent parents or children from pursuing one another in subsequent timesteps.

Results

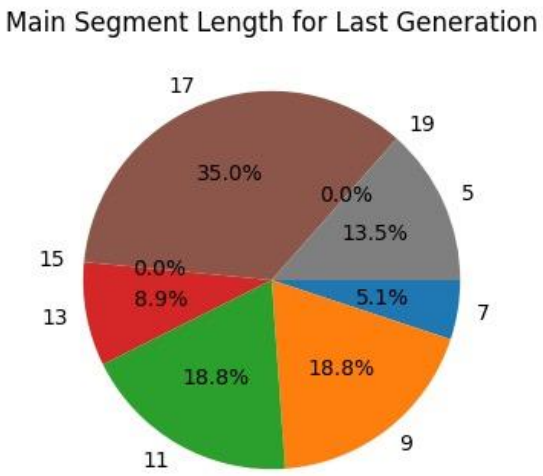
In order to analyze the ways that a simulated pond's evolution is affected by specific traits or external factors, and in order to identify the most significant features, we generated summary statistics and graphed their distributions to compare the initial and final populations of swimbots. We measured the following traits: the number of segments, main segment length (one among the several possible mate preference criteria), translational and rotational movement parameters, age, and energy.

We varied several simulation initialization parameters and observed the following:

Parameters Varied	Observations in Final Generation	Comments
Choosing a swimbot with more segments	Most frequent segment number in the last generation: 2 The most frequent segment number in the last generation: 8	We would expect that the swimbots at the end would have more segments, but it appears that despite having a preference for it, another trait might be more prominent.
Choose swimbots that move faster	Average translational velocity parameter… in last generation: 3.536965 in first generation: 4.53 Versus default… in last generation: 2.395939 in first generation: 4.53	In both simulations, translational velocity parameter distribution becomes less uniform; higher average translational velocity parameter observed in simulation with mate preference for faster swimbots
Choosing swimbots with similar length of mainSegment	Average main segment length… in last generation: 2.008357 in first generation: 5.255 Versus default… in last generation: 2.27665 in first generation: 5.255	Average length of the main segment in the last generation remained similar compared to default. And the most frequent range group of main segment length didn't show significant differences either. However, the distribution of the length groups changed. The proportion of the most frequent group increased compared to default.
Choosing swimbots with similar number of segments	Average segment number… in last generation: 2.008357 in first generation: 5.255	At the conclusion of both simulations, the most common segment number is

	Versus default… in last generation: 2.27665 in first generation: 5.255	2, but in the simulation in which swimbots favor similar numbers of segments, almost no swimbots have segment numbers other than 2.
Varying Food frequency Increasing: food placed at every timestep instead of every 5 timesteps. Decreasing: food placed at every 10 timesteps instead of every 5 timesteps.	Increasing: 1610 swimbots, 97.521874 max energy Decreasing: 270 swimbots, 90.059049 max energy Versus default case: 529 swimbots and average energy of 94.159245	More food helps in generating more swimbots since they acquire more energy more easily and shift their preferences to mating, while less food hinders the generation of more swimbots since less energy supply is available, and so the goal of getting food is more important than generating bots.
Initializing all the swimbots with the same translational and rotational energy	We start out with 200 swimbots, and end up with 232 bots. Versus default: 498 swimbots are generated. The average energy values, movement parameters, .. remain relatively similar.	We observe that when all the swimbots move at the same pace, the system maintains itself in a similar fashion, in a type of “equilibrium”, while if we vary the initial energies, we observe more variation and evolution of the system.

A sample of the graphs we generate is observed below:



Where the labels refer to the characteristic we are dealing with (main segment length), and the percentages in the center correspond to the percentage of swimbots in the last generation with these characteristics. In cases where our characteristics are continuous variables, we define bins of specific ranges and allocate the swimbots to them accordingly. So based on this figure, 35% of swimbots have their main segment length between 17 and 19.

Conclusion

Overall, by tweaking our parameters, we were able to observe evolutionary patterns in our artificial lifeforms and study how external factors (such as food), and organisms' preferential characteristics are related and how they are affected by each other.

Conditions facilitating more food intake, such as increased frequency of food bit generation or relaxing the proximity required to reach a goal, helped increase reproduction compared to default conditions; because energy gain was easier, swimbots could more readily pursue mates as goals and ultimately produce offsprings.

If all the swimbots have the same characteristics defining their motion, the system maintains a type of "equilibrium" where the number of swimbots present at every generation is relatively stable, with other correlated factors maintained similarly. However, varying the initial energies and ensuring differences in swimbots' abilities to move leads to a more complex system where traits advantageous to more efficient goal pursuit are more likely to propagate to later generations.

When it comes to what is "attractive vs. efficient", we do not have a conclusive answer because we do not necessarily know if the attractive traits contribute to efficiency or inefficiency under our current system of motion and energy depletion. With a more advanced and physically realistic system of motion, in which the morphology of the swimbots contributes more directly to their ability to move – as those in Ventrella's swimbots with "wiggling" segments do – we would be able to answer these questions more definitively. However, we can comment on both these aspects, and say that when certain "efficient" traits are favored such as choosing a swimbot that moves faster, organisms will be produced more easily, and the evolved organisms in later generations will carry part of these traits that help them evolve (as observed with a higher average translational velocity). When it comes to what is "attractive", preferred traits such as choosing a swimbot with more segments or a similar number of segments to itself are chosen. When choosing a more attractive trait, we expect the system to eventually evolve to a system where the favored gene is more prominent. However, this is not necessarily the case for all types of mate preference, as we did not observe that in all our simulations. We observed that in some cases, the distribution of the groups changed, where the proportion of the most frequent group increased compared to default, but the overall average of these characteristics appears quite similar.

References

[1] Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. (Reading, MA: Addison-Wesley), pp. 1-26.

[2] Ventrella, J. (1998). Designing Emergence in Animated Artificial Life Worlds. *Virtual Worlds: First International Conference Proceedings.* (Berlin: Springer), pp. 143-155.

[3] Ventrella, J. (1994). Explorations in the Emergence of Morphology and Locomotion Behavior in Animated Characters. *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.* (Cambridge, MA: MIT Press), pp. 436-441.

[4] Ventrella, J. (2005). GenePool: Exploring the Interaction Between Natural Selection and Sexual Selection. *Artificial Life Models in Software.* (London: Springer), pp. 81-96.

[5] Ventrella, J. (1996). Sexual Swimmers: Emergent Morphology and Locomotion Without a Fitness Function. *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior.* (Cambridge, MA: MIT Press), pp. 484-493.

[6] Ventrella, J. (1998). Attractiveness vs. Efficiency: How Mate Preference Affects Locomotion in the Evolution of Artificial Swimming Organisms. *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life.* (Cambridge, MA: MIT Press), pp. 178-186.

[7] Ventrella, J., Dodd, B. (2004). Darwin Pond. <https://www.ventrella.com/Darwin/darwin.html>. Accessed December 14, 2022.

[8] Ventrella, J. (2021). GenePool. <https://www.swimbots.com/genepool>. Accessed December 14, 2022.