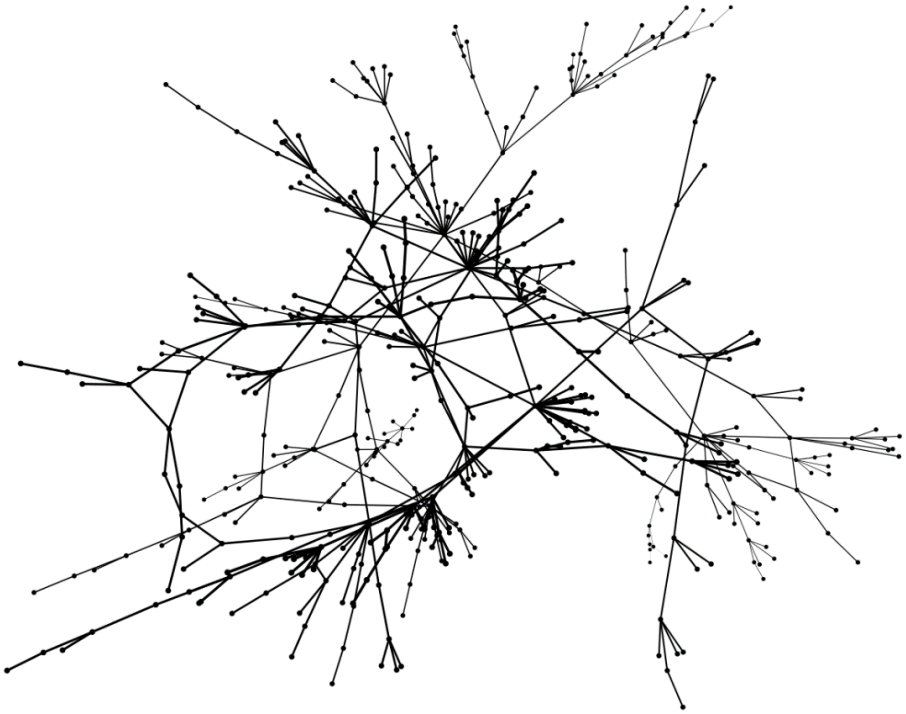
Game of Evolution

Emergence of 3D Networks through Autonomous Particles

Stefan Beeler
stefanbeeler@hotmail.com
github.com/GameOfEvolution
April 2023 / 785202
Transaction: aa284d63c2c065962246ba7bbeffaf32c5f895e06a99c3827c2ba7d91fd14fac



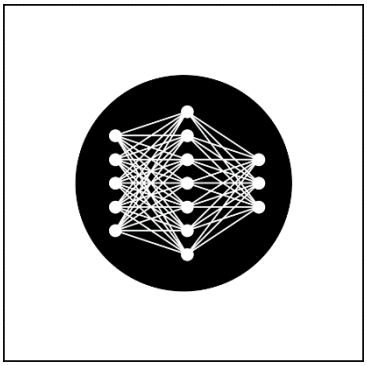
Abstract: In this paper, we present a new algorithm that explores the evolution of complex and organic networks through the behavior of autonomous particles, which have properties of living creatures. The algorithm utilizes different feedforward neural networks to govern the behavior of individual particles, which are linked together to form a graph. These particles interact with their neighbors and compete for scarce fungible tokens to survive and reproduce. Over time, natural selection sorts out fragile behaviors while promoting the growth of antifragile ones. The algorithm allows for a wide range of settings, leading to diverse incentive structures and macroscopic structures. Through testing various combinations of settings, we were able to observe the emergence of autonomous, decentralized, and three-dimensional networks which are always evolving.

1. Introduction

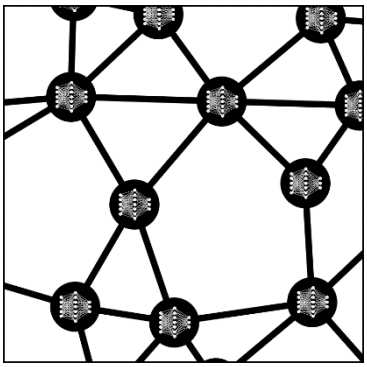
Humans have used mathematical equations to describe the nature of reality for a long time. Currently our best physical description of the universe is a combination of quantum mechanics and general theory of relativity. Both theories describe the mechanisms of the universe with an astonishing accuracy. Despite of their success, they couldn't be unified to a fundamental theory of everything to this day, although there are many candidates of course. Either one of the candidates turn out to be correct as research progresses or mathematical equations are not the right tool to describe the fundamental mechanism of the universe. For example, even though Conway's game of life is an algorithm, there are no mathematical equations that can predict the evolution of structures inside of the algorithm in a macroscopic way, instead the algorithm must be executed iteration after iteration to find out what is going to happen. Additionally, there is no way to predict if the algorithm ever stops at a stationary state, which is called the halting problem. In this paper it is not about describing reality with equations, instead it's about finding an algorithmic framework that could potentially model reality. The original idea was to program an autonomous simulation with evolutionary rules. For the past two years we have worked iteratively on this algorithm trying to make it work more general. The goal was to not implement behavior in a hard coded way, instead the behavior should emerge by itself using the mechanism which led to biological life on earth: natural selection. We were curios how far we could push this idea of an evolutionary universe. We didn't want to assume the dimensionality of this algorithm, instead the goal was the emergence of a 3D network through autonomous particles, since the universe also has three spatial dimensions. We tried to make the algorithm as fundamental as possible, so that there are as few ambiguities as possible. Every time ambiguities have arisen; different options have been implemented that can be turned on or off. Nevertheless, many assumptions have had to be made. After 50+ different versions we finally came up with a framework that we were satisfied with. Many thousands of random option combinations have been tried out. The combinations that led to the emergence of 3D networks with satisfying properties have been selected by hand. Most probably there are many more features that have not been thought of, that could be experimented with.

2. Assumptions

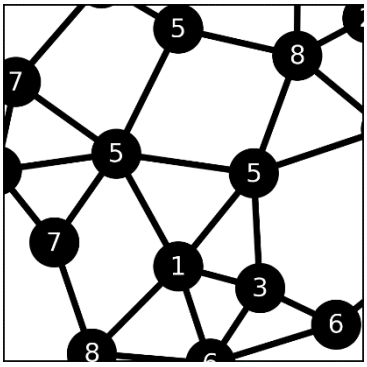
The fundamental assumption is that the most basic part of the algorithm is a living creature in the form of an autonomous particle. It interacts with other particles with the use of its own behavior, and it can spawn new mutated replicates of itself. The particle needs a mathematical function to process information from its environment to produce decisions based on the output of this function. This function needs to be able to change through mutations and should be able to mimic all kinds of different behaviors. For this reason, feedforward neural networks have been chosen as a mathematical function since they satisfy the conditions well. The idea is to delegate as many decisions as possible to this particle.



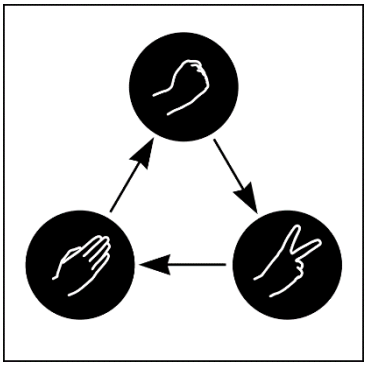
The next assumption is the principle of local action. Particles must somehow interact with other particles. But particles shouldn't interact with every other particle. Instead, they should only interact with some other particles. For that reason, particles are linked together in a network/graph, and they only interact with particles that they are directly connected with. That way the particles are the nodes of the graph. Furthermore, no coordinates are assigned to the particles so that no dimensionality of a coordinate system must be assumed. Note that in this way the particles don't exist in a higher-level environment, instead they form together their own environment.



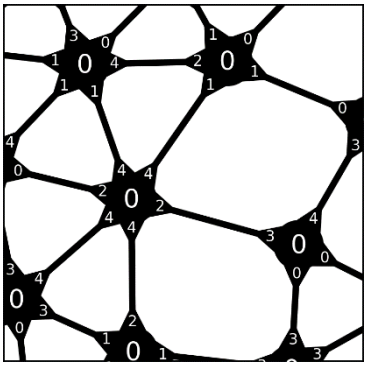
For evolution to happen there needs to be natural selection. Therefore, a competition must be implemented so that a selection process of the fittest particles can happen. For that reason, the assumption was made that a scarce resource in the form of fungible tokens exists for which the particles compete. The particles need a minimum number of tokens to survive. If this number drops to zero, the particle vanishes. Particles shouldn't be able to indefinitely reproduce by repeatedly halving the number of tokens they have. Otherwise, they always have a small percentage of a token left and don't ever vanish. For that reason, particles can only have an integer number of tokens. The higher the number of total tokens in the simulation, the larger the network will become. The limitation of tokens leads to a zero-sum game.



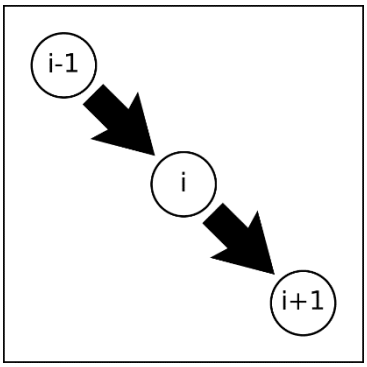
Particles must have a way to involuntarily lose tokens. For that reason, a game with winners and losers must be implemented. An additional assumption is that the game must be playable by a single observation of the particles. Rock, paper, scissors is a good candidate for such a game which is why it was chosen for this algorithm. Other games might have worked as well. There is also an option that deactivates the game and then the particles just decide, which neighboring particle should get their allocated token.



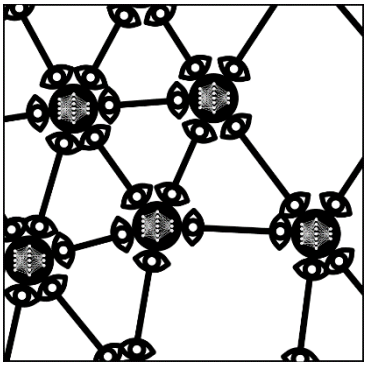
A particle that decides to do nothing with its tokens would survive indefinitely. Furthermore, a particle that could decide to use 100% of their tokens for reproduction wouldn't ever have to adapt to anything because their tokens would never leave their birth line. To avoid these static scenarios, the assumption was made that particles must allocate all their tokens to the game with their neighboring particle in each iteration.



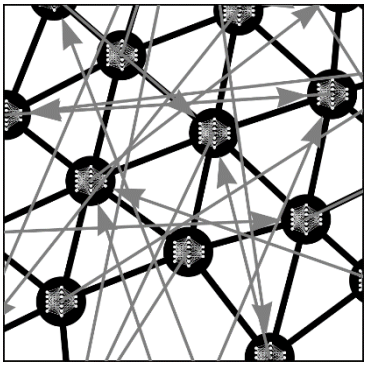
Because particles have an integer number of tokens, this number can also just change by an integer amount. This means that no continuous change of real numbers is possible and necessary. Furthermore, the geometry of the network can also just change discretely. A new link or particle can only appear instantly and not in a continuous way. That is why no infinitesimal time increment must be defined. This leads to the assumption that the network and values change discretely in iterations that are executed after each other. These iterations resemble the advancement of time.



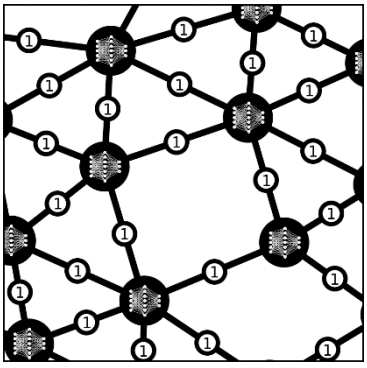
In each iteration all observations and actions are made simultaneously at the same network state. Along every link two observations are made in each direction by the respective particle. That way, only actions are possible that theoretically could be executed by all particles in the same iteration. This makes the action space of the particles smaller than it would be, if one particle at a time takes actions. But in doing so the order of the observations do not matter which was assumed to be more important.



Since a particle can only observe another particle along a link, it cannot directly reference any third particle that is not already connected with itself. But to connect with an unconnected new particle there needs to be some reference. That's why the additional assumption was made, that particles have a walker that can move around in the network to reference other particles. A particle can create new links with the referenced particle. Optionally this referenced particle gets also used for certain decision making.



Furthermore, the assumption was made that particles shouldn't be able to create an unlimited number of links. For that reason, a cost of one token was implemented for each link. This token is given back to a particle when the link vanishes. New links get created either by reproduction of a particle or if a particle decides to make a new link. In both cases particles must pay an additional token for the link.



3. Algorithm Design

3.1 Initialization

The simulation consists of particles and links, as well as nodes that are intermediaries between particles and links. At the beginning, several particles are spawned with randomly initialized neural networks and are connected with each other in a loop. This ensures that there is a variation of behaviors from the start, even though the simulation could start with only one particle as well. The total number of tokens must be defined, which are evenly distributed across all initial particles, as well as one token for each initial link.

3.2 Particle Decisions

In each iteration the particles observe their neighboring particles, which means that an input vector containing information of the observation is fed through the neural networks of the particles. The output determines the actions that the particles make. There are twelve different decisions each particle does each iteration, although some of them can be deactivated. In the following table the possible decisions are listed and explained as well as the amount of output numbers that are needed to make the decision.

Decision	Explanation	Output numbers
1. Go Home	Decides whether the walker should go back home to its particle, creating a new link with the referenced particle in the process.	2
2. Accept Move	Decides whether any particle that references the particle itself is allowed to go home and create a new link.	2
3. Move Particles	Decides in which direction the references should be moved.	1
4. Dying Direction	Decides which neighbor should inherit its own links in case the particle itself dies during this iteration.	1
5. Swapping	Chooses a neighbor with which it wants to swap places. This only happens if there is mutual consent.	1
6. Shifting Decision	Decides which links should be reconnected to the new particle in the case of reproduction.	2
7. Rock, Paper, Scissors	Decides whether to play rock, paper, or scissors with the observed neighbor.	3
8. Token Allocation	Decides how its own tokens are distributed and allocated to the rock, paper, scissors game with its neighbors.	1
9. Keep Tokens	Decides what percentage of reproduction tokens should not get used for reproduction.	2
10. Plant or Not	Decides whether a reproduced particle should either get planted or reproduced with an umbilical cord to its original particle.	2
11. Reproduce Which	If the option is activated, that a neighbor should get reproduced, this decides which neighbor to reproduce.	1
12. Plant on Which	If the option is activated, that a particle should be planted on a neighbor, this decides on which neighbor to plant the new particle.	1

There are two options for each of the decisions. One of the options determines by which particle the decision should be made. This means that each decision gets either made by the particle itself, its neighboring particle, or the referenced particle. For decisions 2 and 3 there is also the option that the decision gets made by the particles that reference the observing particle. That way the particles can make decisions for their reference. The second option determines how the outputted numbers lead to a decision. Either the maximum of the numbers is decisive, the numbers get treated as probabilities for the decision making or the numbers don't matter, and randomness decides instead.

3.3 Main Loop

After initialization the main loop starts which executes iteration after iteration. One iteration consists of different steps that are done in a certain order.

3.3.1 Iteration Preparation

The first step is to prepare information for the iteration. This includes resetting certain variables from the previous iteration for every particle, link, and node, initializing arrays for data analysis and the assembly of input values for the next step.

3.3.2 Decision Making

Now, input vectors are assembled for each particle in each possible direction. This input vector maps the current state of the network of the perspective of one particle looking at another particle. Information like token amount, link amount, number of other particles that are currently referencing this particle, as well as the choice of the last rock, paper scissors game from the last iteration both for the observing particle as well as the observed particle are put into this vector. Additionally, information about all neighbors and information about the referenced particle is included in this vector. Then the vector is fed through the feedforward neural network of the particle. Based on the generated output, decisions are made for each particle. In the algorithm the vector is not only fed through the neural network of the observing particle, instead it is fed through the neural networks of the observed particle, the referenced particle as well as all particles that reference the observing particle. For each possible decision there is an option, which particle is responsible for a decision. If for example the option for playing the rock-paper-scissors game is changed so that the neighboring particle should decide what to do, the particles start to do things not for themselves, but for their neighbors, which leads to all kinds of different incentive structures.

3.3.3 Game Evaluation

Along each link a game of rock-paper-scissors is played with tokens that are bet from both particles. There are different ways of distributing tokens based on the outcome of the game. It does matter if the tokens are redeemed as reproduction tokens or as regular tokens. Only reproduction tokens can later be used for reproduction by the particles. Many different variants have been implemented that can be chosen from by the settings of the simulation. If for example there is a draw, the tokens can either go back to the original particle or they can be forwarded to the other particle, since both methods are symmetric. Another example is that if one particle wins, it can either get all the tokens of the other particle or only the amount it has bet itself. There are more options that are best understood by inspecting the code.

3.3.4 Swap

Particles can choose another particle that they want to change position with. If there is mutual consent, they do swap positions. Otherwise, nothing happens.

3.3.5 Move

The references of the particles are moved into the chosen direction. This decision can not only be made locally, but also by the particle itself. Particles could have also chosen to go home, meaning that the pointer moves back to the particle itself, creating a new link between the referenced particle and itself.

3.3.6 Reproduction

If a particle got reproduction tokens from the game evaluation, it does reproduce. This spawns a new particle with a mutated neural network. Again, there are different simulation settings that do different things. For example, the particle doesn't necessarily have to reproduce itself, instead there is also an option that particles reproduce their neighbors or their referenced particle. Furthermore, there are multiple ways how the new particle is embedded in the network. Either it is planted on a neighbor or the referenced particle, or an umbilical cord is created with the original particle. Existing nodes can also be shifted to the new particle if an umbilical cord was made, which was chosen by the particle as well. There is also an option that, lets the particles keep their reproduction tokens as regular tokens.

3.3.7 Check Death

If a particle has no more tokens left, it vanishes. But the links of this particle cannot just vanish as well since the network would be at risk of splitting into multiple networks. Those networks would never have a way to reconnect again, which would lead to a fragmentation that gets worse as time advances, which is why all the links of the vanishing particle must be shifted to a neighboring particle, which was chosen by the vanishing particle. After all links have been shifted to this chosen particle, the particle as well as the one link that was connected to the chosen particle vanish. This is the only ways a link can vanish.

3.4 Remarks

All actions and interactions of the particles were designed in a way, so that they are not dependent on the order of execution. This way no particle has a systematic advantage against others. This also leads to a symmetric interaction between all particles.

4. Method

The algorithm was equipped with as much functionality as possible. Then, optional restrictions for most functionality was implemented. Most of the options just change a small detail which is best understood by inspecting the code itself. Different combinations of these options led to the emergence of different kind of networks. Many thousands of different random combinations were tested. We handpicked the simulation settings which led to the emergence of networks with an approximate dimensionality of 3 since that's the dimensionality of the universe.

The dimensionality of any network/graph can be calculated in the following way. A random particle of the network is chosen and colored red. Then the first step is to color all neighbors of this particle also red. The second step is to color all neighbors of the currently colored particles red. This is done until all particles are colored red. Now the number of colored particles $V(r)$ for each step r is known. Compare this to the formula for the area of a circle or the volume of a sphere where the radius represents the number of steps.

$$Circle\ Area: A(r) = \pi \cdot r^2, \qquad Sphere\ Volume: V(r) = \frac{4}{3} \cdot \pi \cdot r^3$$

The area/volume reached with the radius r scales with r^d. The general Form is:

$$V(r) = V_r = C \cdot r^d$$

Where C is a constant and d is the dimensionality. It is known that the constant C doesn't change for any r. That's why we solve for C and get a formula that must hold for all r. Replace r with $r + i$ and we get:

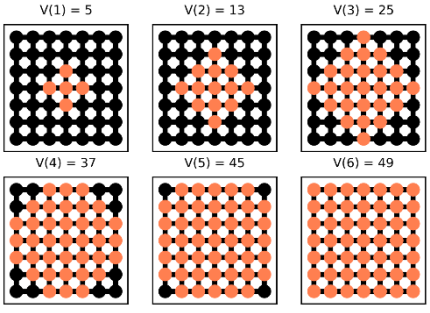
$$C = \frac{V_r}{r^d} = \frac{V_{r+i}}{(r+i)^d}$$

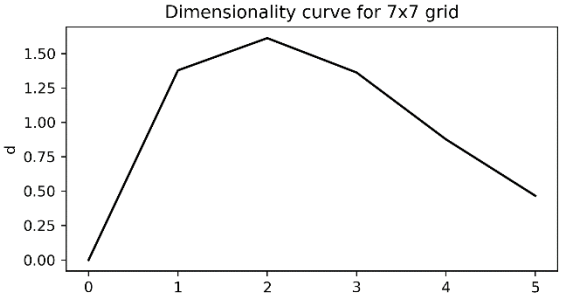
We solve for d:

$$\frac{V_{r+i}}{V_r} = \frac{(r+i)^d}{r^d} = \left(\frac{r+i}{r}\right)^d$$

$$d(r,i) = \log_{\frac{r+i}{r}}\left(\frac{V_{r+i}}{V_r}\right) = \frac{\log(V_{r+i}) - \log(V_r)}{\log(r+i) - \log(r)}$$

The dimensionality d can be calculated between steps r and $r + i$. We choose $i = 1$ and calculate d for adjacent steps. This gives another dimensionality number for each step. In the following example we analyze the dimensionality for a simple 2D grid network with the size 7x7. Step by step the whole network is colored red. The dimensionality d is calculated for each adjacent step. The dimensionality goes higher until the border of the grid is reached. The maximal value, which is 1.613, is the closest value to 2, and is therefore the best guess for the dimensionality of this grid. The larger the 2D grid is, the more this maximal value converges to the correct dimensionality of 2. The dimensionality curves are dependent on the initially chosen node. In an arbitrary network multiple random initial nodes are chosen from which we measure the dimensionality. The average of the maximal values of each curve is taken for an estimate of the dimensionality. The larger the network, the more precise the estimate will be.



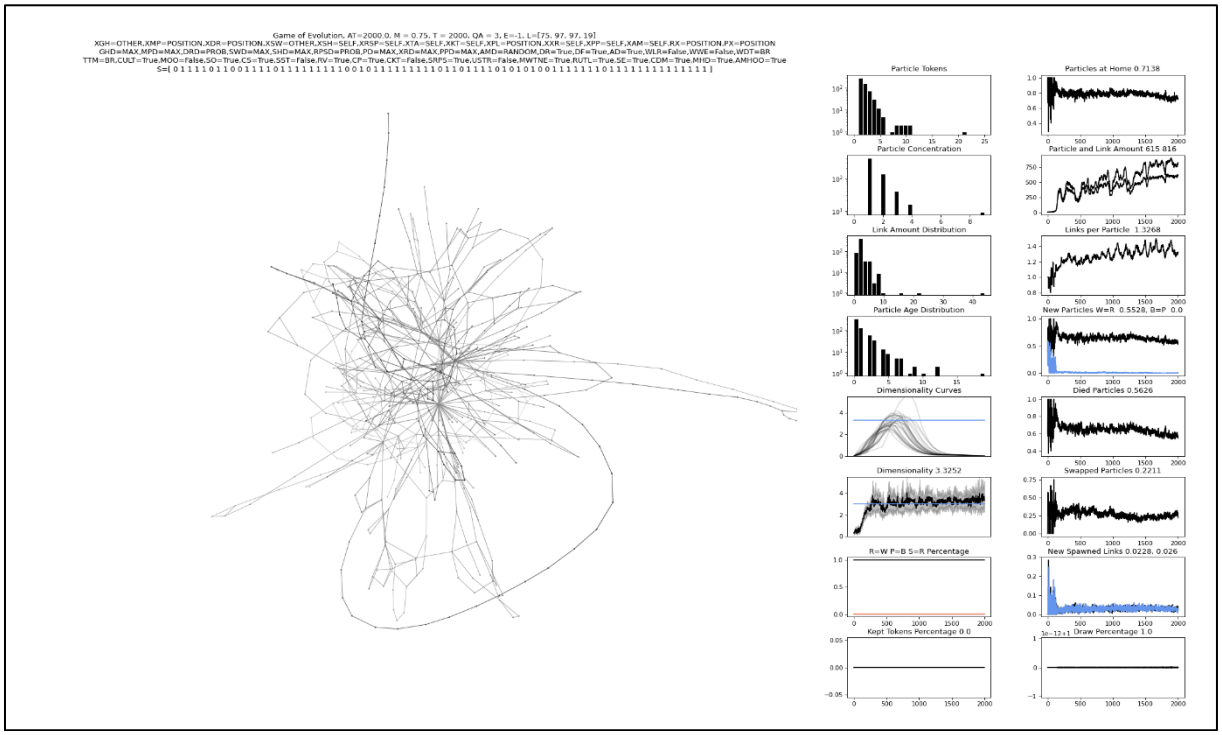


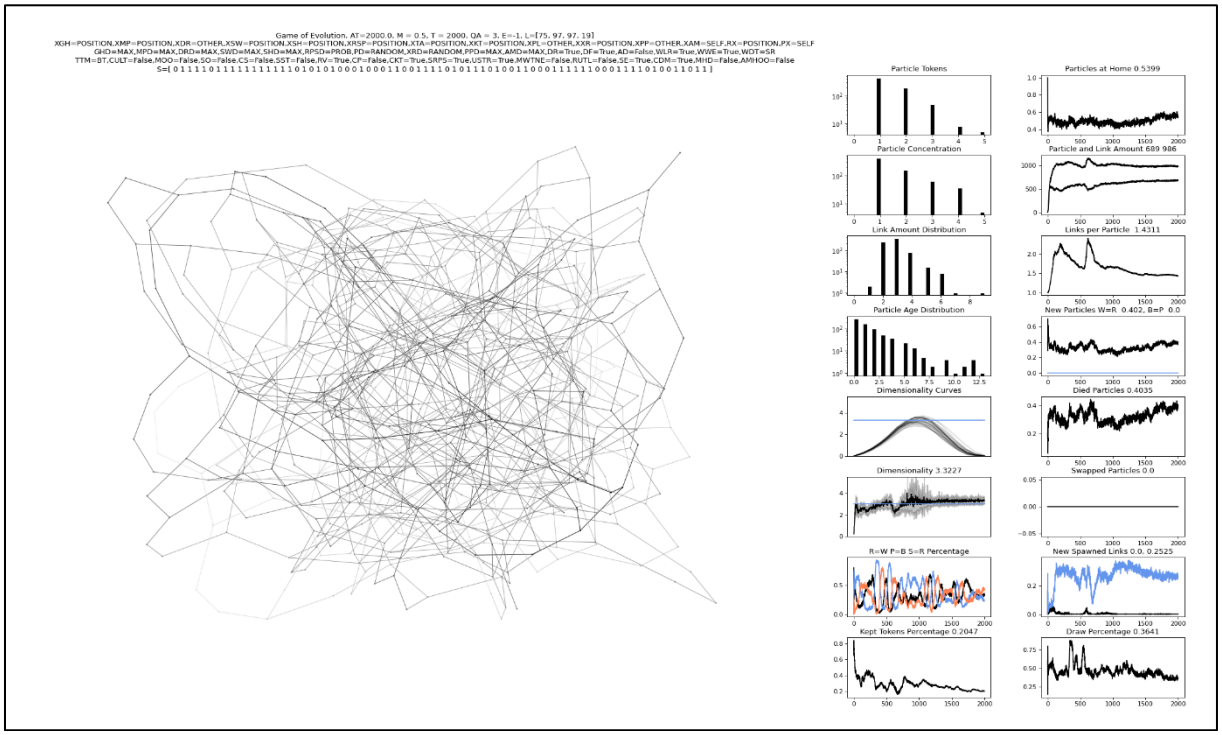
5. Results

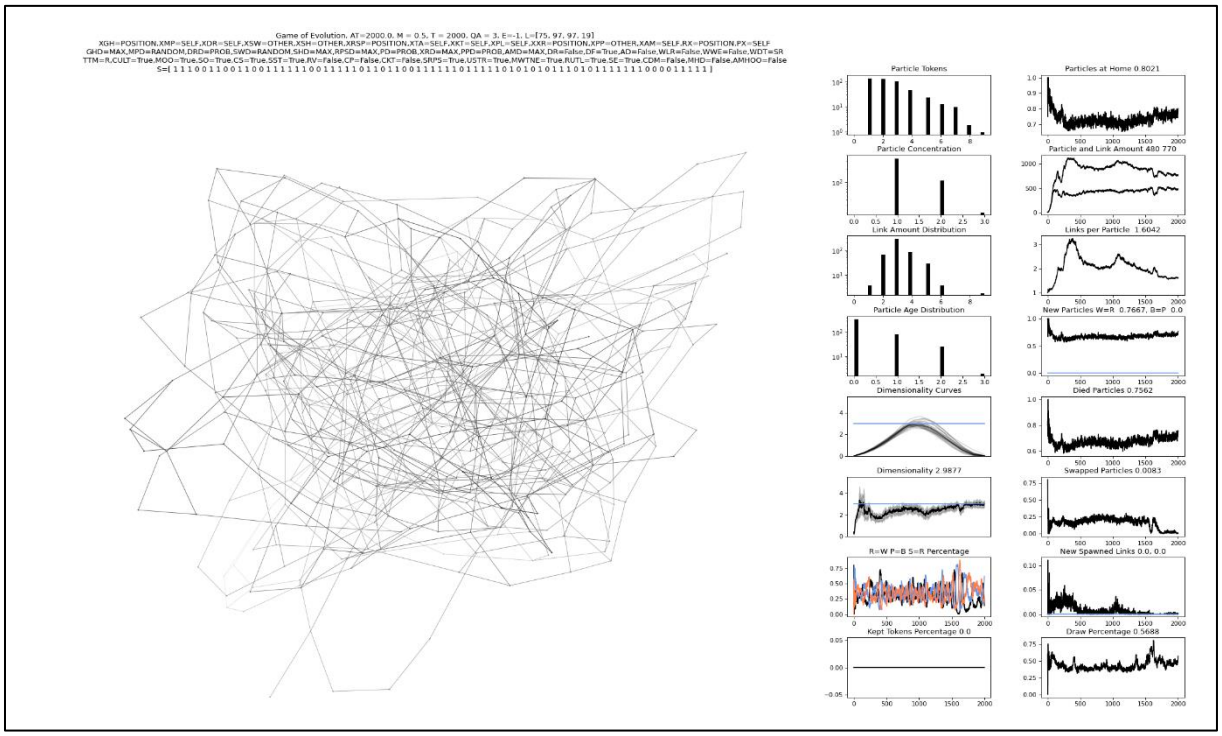
The network is visualized every few iterations using a spring model implemented in the python library networkx. At the top the settings of the simulation are pictured. Additionally, some plots about the state of the simulation are portrayed next to the graph. The pictures are put together in a video in which the dynamic evolution of the network can clearly be seen. The best versions were put together in a long video of three hours length. The link to this video can be found on github.com/GameOfEvolution.

All kinds of different network behaviors have been observed. Some settings lead to the emergence of static networks, where nothing changes anymore after a few iterations. Other settings lead to networks where particles survive indefinitely with one token because they can't afford to pay the additional fee of one token for a new link. Sometimes every particle is connected with every other particle, meaning that information reaches the whole network within one iteration. Sometimes the network is not able to grow or the network collapses to a single particle. But sometimes the settings lead to the emergence of approximately three-dimensional networks with a good distribution of tokens that look organic and dynamic.

In this paper three networks with different simulation settings are portrayed, each after 2000 iterations. The dimensionality plot shows in all three cases that the dimension is around three. Although this number may not be trusted because the networks are quite small. Only 2000 tokens were implemented in these simulations.




Game of Evolution. AT=2000.0, M = 0.5, T = 2000, QA = 3, E=-1, L=[75, 97, 97, 19]
XGH=POSITION,XMP=POSITION,XDR=OTHER,XSW=POSITION,XSH=POSITION,XRSP=POSITION,XTA=POSITION,XKT=POSITION,XPL=OTHER,XXR=POSITION,XPP=OTHER,XAM=SELF,RX=POSITION,PX=SELF
GHD=MAX,MPD=MAX,DRD=MAX,SWD=MAX,SHD=MAX,RPSD=PROB,PD=RANDOM,XRD=RANDOM,PPD=MAX,AMD=MAX,DR=True,DF=True,AD=False,WLR=True,WWE=True,WDT=SR
TTM=BT,CULT=False,MOO=False,SO=False,CS=False,SST=False,RV=True,CP=False,CKT=True,SRPS=True,USTR=True,MWTNE=False,RUTL=False,SE=True,CDM=True,MHD=False,AMHOO=False
S=[0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 1 1 1 0 1 0 1 1 1 0 1 1 0 1 0 0 1 1 0 0 0 1 1 1 1 1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 1]
Particle Tokens
Particles at Home 0.5399
Particle Concentration
Particle and Link Amount 689 986
Link Amount Distribution
Links per Particle 1.4311
Particle Age Distribution
New Particles W=R 0.402, B=P 0.0
Dimensionality Curves
Died Particles 0.4035
Dimensionality 3.3227
Swapped Particles 0.0
R=W P=B S=R Percentage
New Spawned Links 0.0, 0.2525
Kept Tokens Percentage 0.2647
Draw Percentage 0.3641


Game of Evolution. AT=2000.0, M = 0.5, T = 2000, QA = 3, E=-1, L=[75, 97, 97, 19]
XGH=POSITION,XMP=SELF,XDR=SELF,XSW=OTHER,XSH=OTHER,XRSP=POSITION,XTA=SELF,XKT=SELF,XPL=SELF,XXR=POSITION,XPP=OTHER,XAM=SELF,RX=POSITION,PX=SELF
GHD=MAX,MPD=RANDOM,DRD=PROB,SWD=RANDOM,SHD=MAX,RPSD=MAX,PD=PROB,XRD=MAX,PPD=PROB,AMD=MAX,DR=False,DF=True,AD=False,WLR=False,WWE=False,WDT=SR
TTM=R,CULT=True,MOO=True,SO=True,CS=True,SST=True,RV=False,CP=False,CKT=False,SRPS=True,USTR=True,MWTNE=True,RUTL=True,SE=True,CDM=False,MHD=False,AMHOO=False
S=[1 1 1 0 0 1 1 0 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 1 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1]
Particle Tokens
Particles at Home 0.8021
Particle Concentration
Particle and Link Amount 480 770
Link Amount Distribution
Links per Particle 1.6042
Particle Age Distribution
New Particles W=R 0.7687, B=P 0.0
Dimensionality Curves
Died Particles 0.7562
Dimensionality 2.9877
Swapped Particles 0.0083
R=W P=B S=R Percentage
New Spawned Links 0.0, 0.0
Kept Tokens Percentage 0.0
Draw Percentage 0.5688

6. Discussion

This algorithm is merely a prototype. It is unclear if real physical processes can emerge in the algorithm. However, the emergence of approximately three-dimensional networks through the behavior of autonomous particles seems very promising. Especially because the behavior of the particles is everchanging. There are probably more options that could be experimented with that weren't explored in this prototype. For example what happens if the links don't cost one token?

There are a lot of open questions regarding the algorithm. For example, can the particles evolve to work together? Can they form groups and help each other out of bad situations, increasing the probability of survival and reproduction? Would these groups act like elementary particles? If yes, could these groups form groups of groups that may be called atoms? Even if that's true, it's hard to proof, because in the real universe it took 380'000 years for the first atom to form. With the assumption that one iteration of the algorithm has the duration of one Planck time $t_P = 5.391 \cdot 10^{-44}$(which might be wrong), it would take $2.223 \cdot 10^{56}$ iterations to get to this point. Also, the amount of token units (energy units) that must be implemented is gigantic. It is not practically feasible to simulate this. Furthermore, it's unclear how an atom or an elementary particle would look and behave inside of the algorithm. One possible hypothesis that arises from this algorithm might be, that chemical reactions are just a barter system of groups of particles.

The simulation options and rules make a huge difference regarding the type of network and behavior that emerges. The incentive structure also changes with these settings. Sometimes the particles reproduce themselves and are incentivized to get tokens themselves and reproduce. Sometimes they reproduce their neighbor, meaning that they are incentivized to get chosen by their neighbor, which may lead to symbioses between the particles. There are a lot of different incentives that can emerge, which are hard to describe and predict. It's hard to say which settings can lead to real physical processes if that's possible at all. Also note, that there are most probably a lot more rules that could be experimented with that weren't tested out in this prototype. The assumptions should be questioned as well.

One thing that has not been tested, is the idea that instead of the particles having a behavior, maybe the links could have a behavior which might lead to interesting networks as well. Maybe there is a way to make both the links and particles have behaviors which interact with each other.

Another thing that might be interesting is to try to delegate the decision of who gets to decide an action to the particles themselves, which would remove a lot of simulation options. Then the question arises: which particle gets to decide which particle can decide? Maybe a kind of flip flop mechanism could work, where a particle chooses for itself until it decides that another particle can make the decisions. And this particle can make the decisions until it decides again that another particle can make the decision.

In the algorithm the assumption was made that the number of tokens is constant. But maybe it would be interesting if the particles had a way to create new tokens. Maybe this would lead to an expansion of the network like the expansion of the real universe that we can observe. What speaks against this is that it would ruin the mechanism of natural

selection because particles that don't adapt don't die because they still can reproduce with the newly generated tokens. Basically no particle would have to die anymore.

Despite the large number of open questions, there are some similarities to the universe. The three dimensionality that emerges in some cases is one of the similarities. Another is that there is a maximal speed of information flow inside of the network, because particles can only ever interact with their direct neighbors. The reference can also just move along one link per iteration. That way information can generally only flow one link per iteration. This is similar to the speed of light in the universe. Furthermore, the algorithm starts off very organized and small and gets larger and more disorganized very quickly, especially at the beginning. This is like the low entropy state at the big bang and the fast expansion of space at the beginning. The network gets larger until it stabilizes when the tokens are distributed in a kind of equilibrium. Additionally, the constant number of tokens does remind of conservation of energy, although it is unclear whether the tokens can be compared to real energy.

There are different theories that hypothesize what will happen at the end of the universe. In the algorithm it's unclear what will happen at the end. We could observe that sometimes the network comes to a stationary state because it has reached some kind of equilibrium. Its unclear if this is similar to the entropy death scenario of the universe. With other simulation settings no equilibrium could be observed, which may mean that the algorithm never reaches a stationary state. It may also mean that more iterations are necessary for an equilibrium to be reached.

The Idea for this algorithm emerged while dealing with the concept of bitcoin, where the behavior of each participant makes the whole ecosystem more antifragile. This is because the incentive structure of bitcoin was designed that way. We were wondering whether a "geometric" incentive structure with autonomous participants/particles can lead to an "energy" economy that becomes antifragile with the subjective behaviors. Also, the limited supply of bitcoin does remind of energy conservation and the conservation of tokens inside of this algorithm.

Maybe the most important question is whether this algorithm is compatible with quantum mechanics and relativity. It is unclear, how this can be checked. In both theories space is defined to be three dimensional from the beginning. The nodes of the network in this algorithm don't have coordinates though, they themselves form the space. Finding out if the behavior of the particles can be described by the Schrödinger equation or the Einstein equation is challenging. Although according to the wolfram physics project, a graph is potentially compatible with these equations.

Another question that is interesting: is it possible to find unitless physical constants inside of the algorithm? The origin of the unitless fine structure constant is currently unknown. We don't know an algorithm that spits out this number. Maybe this number emerges from an evolutionary process like this algorithm.

7. Conclusion

We have showed that this prototype of an algorithm can produce all kinds of different networks that are made of autonomous particles. Some of which are even approximately three-dimensional. The behaviors of the particles evolve according to natural selection. The system that governs this selection is set by different options and rules that define the possible interactions between the particles. It is still unclear if any combination of settings can lead to the emergence of real physical processes. Further research is needed to investigate the causal relationships that emerge with these options.