Persistence, Search and Autopoiesis

Oliver Sharpe
School of Cognitive and Computing Sciences, University of Sussex
Brighton BN1 9QH, UK. ++44 1273 748 253
olivers@cogs.susx.ac.uk

Abstract

The purpose of this paper is mainly to introduce the concept of 'persistence' into the discussion of dynamical systems and Artificial Life. First it is shown how persistence can be used to help understand the behaviour of evolutionary search algorithms. Then the notion of persistence is discussed in relation to the types of dynamical systems that will be capable of modelling autopoietic systems. It is argued that the first models of emergent autopoiesis will either require an exponential growth in the number of particles involved in the model or it will be based on some kind of search algorithm that approximates this growth.

1 Introduction

Persistence is a characteristic that distinguishes between different types of objects in our world: alive or dead, stable or unstable. The fact that some objects are able to persist, both in static and dynamic ways, and others are not appears to be a cornerstone of why interesting things happen in our universe. Hence, the class of dynamical systems in which persistence plays an important role should be a very interesting class of systems.

This paper is mainly a position paper rather than a results based paper. It looks at the nature of search in terms of persistence and then at how the notion of persistence of objects has implications for dynamical systems capable of supporting autopoiesis. It will be assumed that the reader already knows a fair bit about dynamical systems (Kauffman, 1993), search algorithms (Goldberg, 1989) and autopoiesis (Maturana & Varela, 1980).

2 Understanding the dynamics of search

The common understanding of search is that it is 'looking for something'. However, for describing a kind of dynamics, this kind of definition will not do as it relies too much on intention and ignorance. What if no one is interested in the solution? What if we already know the best solution? When a genetic algorithm (GA from now on) is run on a problem wont it always be doing search, independent of why it is run and what the user already knows about the problem? So for describing such dynamics we need to look at another kind of definition. A definition that is independent of the human user, a definition that can look at the dynamics of a given system and determine whether it is or is not search.

A first attempt at such a definition would be the fact that all search involves some kind of generate and test rhythm. However, a simple algorithm that generates a random genotype and then tests it, without saving the best found so far, is not a search algorithm, as even if by chance it generated and evaluated the best point in the solution space, it would not keep it but would carry on in its random waste of time. Search is about actively choosing and keeping certain states of the system. Search is about the persistence of certain types of sub-states of the system, the active persistence of those states. In fact, generally the sub-states that are persisting are only replaced by sub-states that will persist for longer. Hence a possible operational definition of search could be:

Search is the accumulation and maintenance of persistence.

From any random initial state a search process will first 'find' states that will persist and then it will 'keep' those states. However, to properly understand such a definition we need to have a clearer notion of persis-

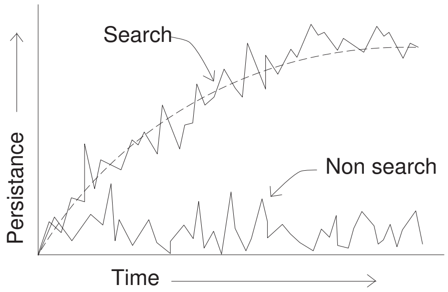

Figure 1: Distinguishing between search like and non-search like processes

tence. So let's look at a definition of a kind of persistence and see what it shows us about search.

3 Measuring Static Persistence

So how could we measure the level of persistence in the state of a GA over time? In this paper we are only going to look at a very simple measure of static persistence. Although it is simple, this measure already illuminates some interesting behaviours of the systems to which we will apply it. Only a few examples are given as the main purpose is to demonstrate the potential uses of looking at persistence rather than to support this particular measure of persistence. For clarity of writing I will refer to the sub-state of a state as being a particle of the state. This is so that instead of having to constantly refer to the state of the sub-state of the state, I can instead simply refer to the state of a particle. A GA has a population of individuals, each with a genotype. So the state of the GA at any given time is the state of the population. If we output the state of the GA after every generation then we can get a trace of the history of the state of the GA. If the GA has binary genotypes then such a trace could start off something like the following:

```
00110000100000111011 10010010010110001111 . .
10101011011001001111 10111011011001101111
10101011011001001111 10101011011001001111 . .
.                                        .
.                                        .
```

Each locus of each of the individual's genotypes is a state holding 'particle' of the system. So a GA running with a population of 20 individuals each with genotypes of length 10 bits long would be a dynamical system with 200 binary state holding particles.

We are going to look at a measure of the amount of static persistence within the state over time. So this is the degree to which the states of the particles remain fixed over time. Hence the degree of persistence of the whole state is linearly dependent on the degree of persistence of each particle's state. So, how can we measure the persistence of a particle's state, especially as a function of time. The simple measure that we will look at here takes the persistence of the state of a particle at a given time to be the percentage of the whole time for which the particle has been in its current state. So this is a measure that is knowledgeable of the whole trace of the GA rather than being calculated independently for each time step. This makes sense, as persistence is a relation over time, but unfortunately this means that it cannot be easily used as an online measure of persistence.

So we can define the algorithm for measuring the static persistence of a trace as follows:

- 1) Give the algorithm a trace, T, of length L time steps with dimension N. (by dimension I mean that there are N particles within the state)
- 2) Associate with each particle the % time for which its current state persisted. (this will be order 2L for each particle so O(N*L) in total)
- 3) Add all of the persistences of each particle at each time point to get the static persistence of the state at each time point: (this will be order N for each time point L so again O(N*L) in total)

So measuring static persistence of an N particle system over L time steps is O(N*L). We can write some pseudo code for such an algorithm as follows:

The major variables:

```
T[N, L] = trace of process.

P[N, L] = persistence of each state of
          each particle.

S[L]    = static persistence measure for
          the trace over time.

start    (variables for calculating length
finish    of persistence of a state)
```

First calculate each particles static persistence:

```
for (i = 0; i < N; i++) {
  start = 0; finish = 0;
  while (start < L) {
    while ((T[i, start] == T[i, finish])
```

```
            && (finish < L))
        finish++;

      for (j = start; j <  finish; j++)
        P[i, j] = (finish - start) / L;

      start = finish;
    }
}
```

Then calculate the whole state's static persistence:

```
for (t = 0; t < L; t++) {
  S[t] = 0;
  for (i = 0; i < N; i++)
    S[t] = S[t] + P[i, t];
  S[t] = S[t] / (N * L);
}
```

3.1 Examples of measuring static persistence

So now that we have defined a measure of persistence, let's look at what it 'sees' when looking at the traces of various different dynamical systems, first of all the dynamics of some stochastic search algorithms on NK landscapes (Kauffman, 1993).

3.2 Dynamics of stochastic search on NK landscapes

The three search algorithms that we will look at are a simple stochastic hill-climber, a multi-start bit flipping hill-climber and finally a GA.

The GA used is very simple, using a population of 20, a mutation rate of an average of 2 bit flips per locus per generation and uniform cross-over. Selection is implemented as steady state 3 player tournaments. The best two individuals make an offspring that replaces the worst of the three. The NK landscapes are a set of tunably rugged abstract search spaces widely used in the theoretical GA literature. The graphs presented here for all of the search algorithms were run on NK landscapes with $N = 50$ and $K = 4$ using next-door neighbours.

All of the graphs are presented with the data of the fitness performance over time as well to compare the traditional way of understanding how the GA is performing. As the fitness values have no particular meaning on an NK landscape, except for their ordering, the fitness values were scaled for greater clarity. It is important to note that the measure of persistence used here is not biased in either direction of time. If the same data is supplied to the measure but in reverse order (the search trace backwards) then the resulting per-

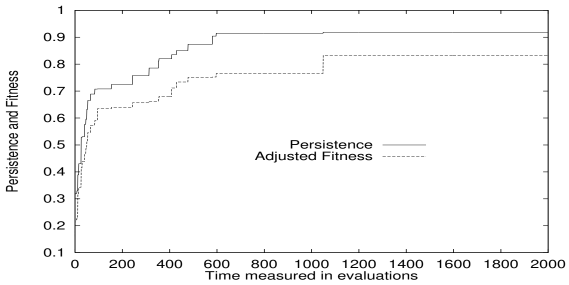

Figure 2: Stochastic hill-climbing on an NK landscape with $N = 50$ $K = 4$

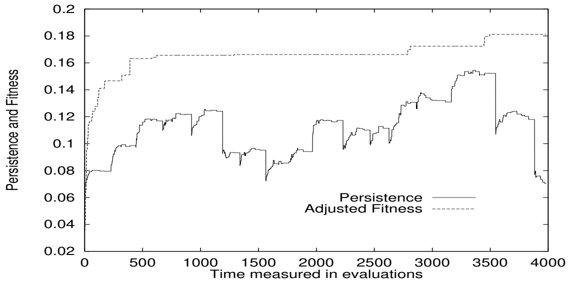

Figure 3: Multi-start hill-climbing on an NK landscape with $N = 50$ $K = 4$

sistence graph is also in reverse order, but otherwise identical. It is also worth restating that the persistence measure used has no knowledge of the fitness values of the different genotypes in the population. It only gets fed the binary strings of the genotypes.

3.3 The dynamics of non-search algorithms

As a comparison to the dynamics of the stochastic search algorithms here we will also look at the what the same persistence measure 'sees' when looking at the dynamics generated by a few other well known types of dynamical systems, and then the differences and similarities will be discussed.

3.4 Discussing the Results

Figures 2 to 5 are of the stochastic search algorithms on the NK landscapes. Figure 2 is of the stochastic hill-climber and clearly shows how the level of persistence, as measured by this static measure, increases rapidly as the search of the hill-climber finds fitter solutions in the early time steps of the trace. Once the hill-climber has reached the fittest point it will reach, the persistence also soon levels off to the highest level of persistence. Notice how changes in fitness are often

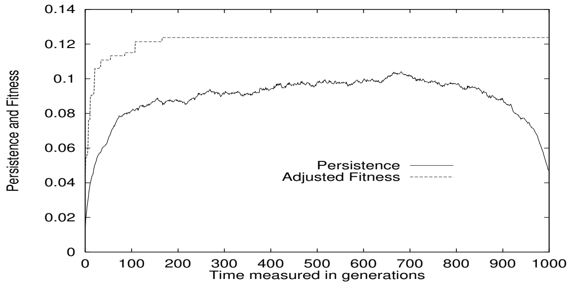

Figure 4: GA with mutation rate 2.0 on an NK landscape with $N = 50$ $K = 4$

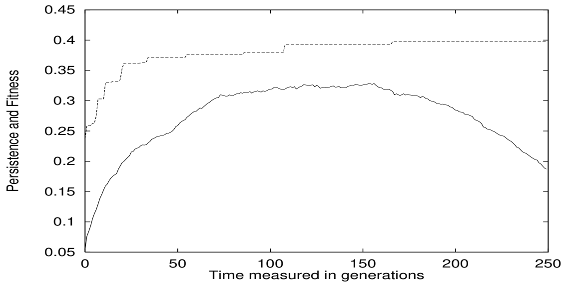

Figure 5: Exactly the same trace as in figure 4 above but only 'seen' by the persistence measure for the first 250 generations

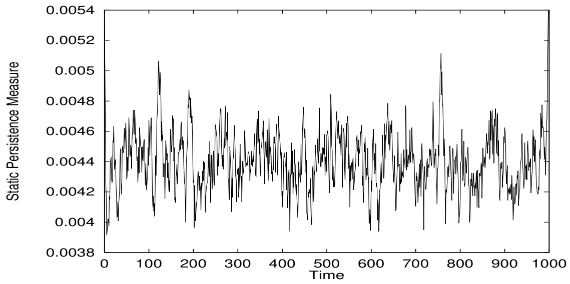

Figure 6: A random trace

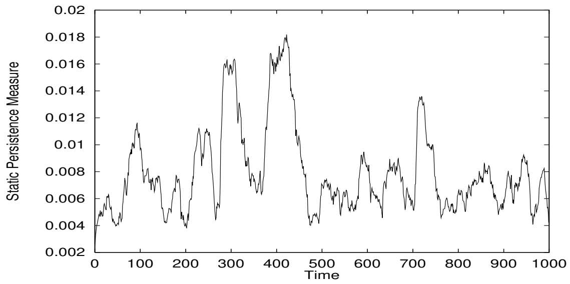

Figure 7: Asynchronously updated 1 Dimensional CA

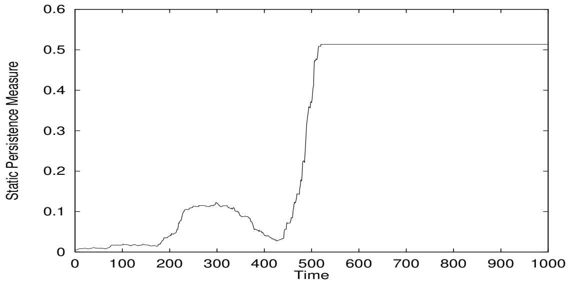

Figure 8: Asynchronously updated 1 Dimensional CA

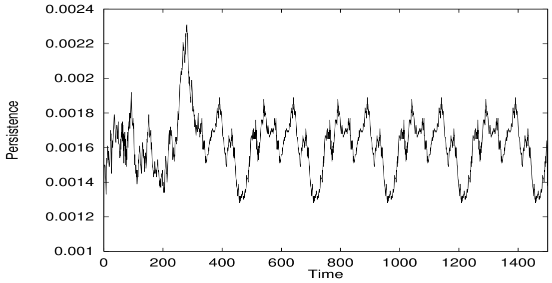

Figure 9: Synchronously updated 1 Dimensional CA

matched by changes in the persistence level.

Figure 3 shows the persistence trace of a multi-start hill-climber. Each of the sharp downwards drops in the persistence level is where the algorithm has re-started its search. Despite these jumps, an underlying trend of generally increasing persistence can still be seen.

Figures 4 shows a trace of a GA being run for a long time on the NK landscape. As the population converges around high levels of fitness, so too does the persistence level of the whole state. It is interesting to note how towards the end of the persistence trace the persistence level drops off quite considerably. This is because this measure of persistence can only tell if a given state has persisted for a long time within the length of the trace. The measure has no way of knowing how persistent a given state would have been if the trace had been longer. Hence, any change of state towards the end of a trace automatically leads to a lower persistence level regardless of how well the new state could have persisted if given the chance.

Figure 5 is of the exact same trace as figure 4, but only the first 250 generations have been given to the persistence measure. The comparison between the two clearly shows the problem that this persistence measure has with the ends of traces as it cannot know how persistent the new states at the end of a trace will be.

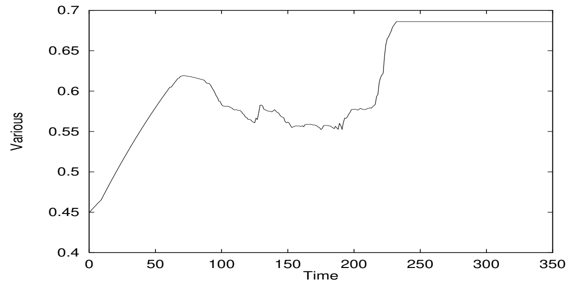

Figure 10: Game of Life

However, it is still possible to see that the dynamical system being measured has pushed up the level of persistence. This is quite impressive considering that the persistence measure gets no information about the fitness of the population.

Figure 6 shows the trace of a completely random trace, and clearly the persistence measure is showing nothing but randomness. Figure 7 shows the persistence trace of a 1-dimensional asynchronously updated cellular automaton with 100 cells, each with a possible of 5 different states. The graph is clearly different from that of the random trace, but also it is clearly not a search process. Figure 8 shows what the persistence trace looks like when the same 1-D CA gets stuck at a point attractor. Figure 9 shows a synchronously updated version of the same 1-D CA when it gets into a cyclical attractor. Figure 10 shows the persistence measure applied to the trace of Conway's Game of Life 2-D cellular automaton (Berlekamp, Conway, & Guy, 1985) when a glider (figure 12b) collides with another cyclic object (figure 12a) roughly at time 60 causing the two to disintegrate completely by roughly time 240. The main persistence being measured here is the persistence of the 'OFF' or white cells of the CA.

So this persistence measure has clearly highlighted some very different features of some very different dynamical systems. Obviously much more work needs to be done, but these examples are meant to show why persistence is an interesting characteristic of dynamical systems that should be explored further. In particular, persistence has been used here to show an objective difference between those systems that are displaying a search like dynamics and those that are not. It is worth noting that any system that 'finds' its point attractors is likely to give a trace of increasing persistence.

Indeed the notion of persistence is heavily related to that of attractors. An attractor is a state from which the system cannot leave, the ultimately persistent state. So, in a way, a measure of persistence is a measure of the degree to which a given state is an attractor. It is interesting to note that for modelling evolution, we do not really want a dynamical system that gets completely stuck. We don't want attractors, but rather we would like temporary attractors that persist for a while before evolving into something new. It is with these thoughts in mind that I first became interested in the notion of persistence. Having shown how persistence can give us some new insights into the dynamics of search processes, let's move on to looking at persistent objects.

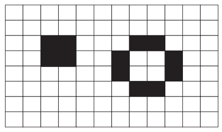

Figure 11: Static objects in Game of Life

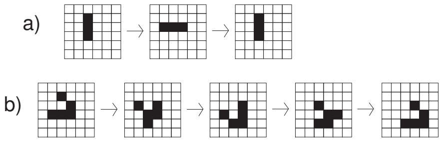

Figure 12: Cyclic objects in Game of Life

4 The Persistence of objects

So far, the static persistence measure that we have looked at measures the persistence of the whole system's state. Next I want to discuss the persistence of objects *within* the dynamical system. For the Game of Life CA there are some static objects (figure 11) whose persistence could easily be measured with a modification to the static persistence measure used earlier. However, the more interesting objects in the Game of Life are the objects, such as gliders (figure 12b), who not only persist in a cyclical way but also persist over different holding particles or 'cells' in CA terminology. The object is a relational object that moves across the grid of cells in the same sense that a wave travels through the water. Let's call objects of this type cyclically persistent relational objects (CPROs). We will use the persistence of CPROs to help motivate an argument about the kinds of dynamical systems that could be used to model emergent autopoiesis.

4.1 Modelling Autopoietic Systems

So we'd like to understand in which types of dynamical systems it is likely that autopoiesis can be inves-

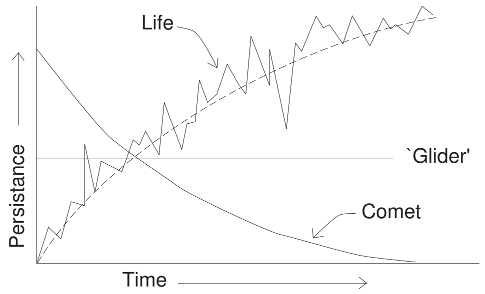

Figure 13: The different styles of persistence of a comet, a 'Glider' and life

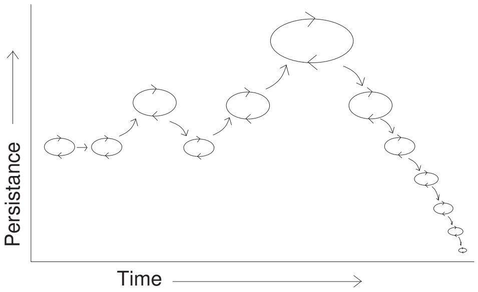

Figure 14: A Markov Chain model of persistence as a random walk will predict a finite time before extinction if the deleterious mutation rate is greater than the advantageous mutation rate

tigated. The 'virus of matter' that is life seams to be slowly increasing its ability at persisting in a generally persistence unfriendly universe. Comets and rocks break up into smaller and smaller rocks, slowly disintegrating (figure 13). The continued existence of rocks on earth is supported by the continuing creation of huge rocks, mountains, by plate tectonics which recycles the old material to make new rocks. So it is not the rocks themselves that enable rocks in general to persist on earth. In contrast, organisms before disintegrating give birth to other organisms that are almost identical to the parent. In this way mouse objects maintain the existence of mouse objects in a cyclical manner that involves only the mice. So we can crudely see a mouse as a version of a CPRO. Not only over its life time does the mouse CPRO remain a mouse whilst being 'held' by different particles, but also any offspring mice are, in a sense, continuations of the same relational pattern, the same CPRO, but in a different set of particles.

So, to look at which types of dynamical systems could support the emergence of an autopoietic system, we can first assume that it must at least be able to support CPROs. Furthermore, for the autopoiesis to have 'emerged' I mean it must have evolved itself into existence rather than being placed there by the designer. Hence the CPROs must not only be able to persist over time, but they must also be free to increase their ability at persisting, free to evolve. So let's have a look at the stability of CPROs within dynamical systems.

4.2 Markov Chain models

Given a CPRO, when some environmental effect on the CPRO occurs, let's call it a mutation to the CPRO, then this alteration can either make the CPRO more stable, equally stable or less stable. Let's say that 51% of the mutations are deleterious and 49% are advantageous, then a simple random walk Markov chain model (Syski, 1992) can predict the expected extinction time for such a CPRO. Once the CPRO has reached the point of having no ability at persisting then it will disintegrate and will not exist any more (figure 14). The Markov chain model predicts that given a set of such CPROs, there is an expected finite time after which all of them will have disintegrated assuming that the chances of mutations occurring to them does not change. As the majority of dynamical systems will have a much larger percent of mutations being deleterious to CPROs this shows the precarious position of CPROs in such dynamical systems. Irrespective of how the CPROs come in to being, they are inherently unstable on average in the long run. This is exactly the case of rocks in our universe.

So in such a dynamical system, to get a CPRO to be stable in the long run requires considerable thought on the part of the designer of the system. In the Game of Life, CPROs are generally stable so long as they are not interfered with. Usually when they are interfered with they either break up into smaller CPROs or they completely disintegrate. One way to ensure that a dynamical system has an increase in persistence would be to set up the system in such a way that, given a particular starting state, the CPROs that occur are likely to have mutations to them that will result in better persistence. This is a tall order and it suggests a set of rules for the system that are designed with the different levels of persistence in mind. I cannot help but imagine that in such systems persistence will increase, but only up to the point that the designer has built into the system. To get ever more complex forms of persistence will require the designer to put ever more complex rules into the system to ensure that that the likely mutations to CPROs will on average increase and not decrease their persistence.

However, in a system where deleterious mutations to

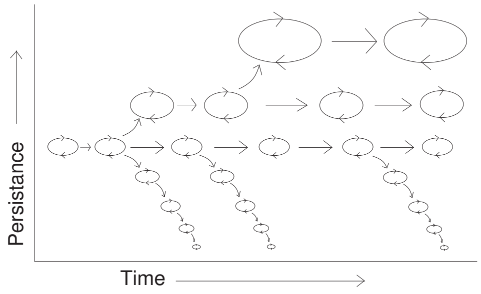

Figure 15: Replication is at the heart of a *persistence ratchet*. This useful dynamic comes at the cost of exponential growth in the number of objects

objects are more likely than advantageous mutations (in terms of their ability to persist as individual objects) then for those objects to continue to persist over very long periods of time, let alone to increase their ability at persisting, they must either be the product of a process that will actively make new objects to replace the old ones or they must be self-reproducing.

4.3 Replication as a Persistence Ratchet

So what is really needed to enable a general increase in persistence to occur is some equivalent of replication. Replication is at the heart of ensuring that persistence at least does not decrease, and therefore has a chance that it will occasionally increase (figure 15). Replication acts like a ratchet on persistence. Once we have replicating cyclically persistent relational objects (RCPROs) then we are in business. The only other thing that we need is to have the rate of replication being larger than the rate of mutation. So now we are in the position where given a simple RCPRO, its numbers will grow exponentially. As we know that the rate of mutation is less than the rate of replication we can guarantee that even if all of the mutations that occur are deleterious this type of RCPRO will still persist. Hence if nothing else, the level of persistence in the system will not go down. Replication has given us a ratchet on persistence. Now if an advantageous mutation occurs to a RCPRO, however unlikely, once it occurs it too can take hold in such a way.

With this persistence ratchet in place, persistence in the system will generally increase with time. So any system that is capable of supporting such a persistence ratchet and therefore capable of supporting autopoiesis, will also be exhibiting the dynamics of a search process as defined earlier.

A weaker form of the persistence ratchet guarantee is that the number of deleterious mutations that are likely to occur is less than the number of replications that are likely to occur in a given time. If most of the mutations that are likely to occur will be neutral then it is relatively easy to have this weaker guarantee holding true. So with RCPROs it is possible to have ever increasing levels of persistence even if the chance of a given RCPRO mutating into a more persistence RCPRO is small.

The only cost of all this replication is that the dynamical system must potentially support exponential growth in the number of instances of RCPROs. For example, long periods with mainly neutral mutations will result in such exponential growth. Although finite resources in the system could potentially give rise to a form of co-evolution between the RCPROs, there must be sufficient room in the system to ensure that the replication ratchet is definitely in place. As the number of deleterious mutations in most systems is likely to be large compared to even the neutral mutations, therefore the growth rate must be similarly large. So, to get autopoietic systems to emerge in a dynamical system not only must the system support RCPROs but also it must be able to support them growing at a rate that is fast enough to preserve the persistence ratchet. The danger here is that this means that the number of particles in the system will be growing, potentially in an exponential fashion. Hence, even if we are generous with the algorithm's complexity and say that it is O(n) where n is the number of particles, then as the number of particles increases over time we will have a decrease in the performance of the algorithm, potentially it will slow down at an exponential rate. This is not good news. It renders this kind of model of autopoiesis effectively intractable.

If we want to study autopoiesis we have to find tools with which the study is a tractable proposition. In the next section I make the argument that traditional generate and select search *algorithms* can be used as an approximate model of these search *processes* that occurs in dynamical systems that can support persistence ratchets and hence emergent autopoiesis. They make the modelling tractable by removing the exponential growth whilst maintaining the persistence ratchet.

4.4 Search as an approximate model of Exponential Growth

The only way out of this exponential slow down is to somehow make an approximation of such a system by trimming the exponential growth to reduce the number of calculations that are necessary, thereby enabling the simulation to be feasible. The most important

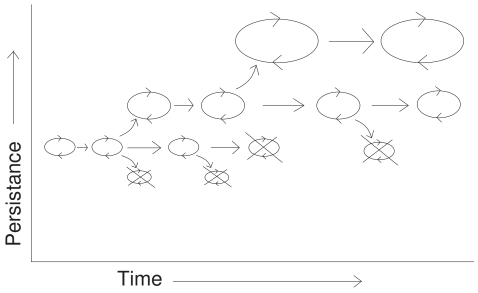

Figure 16: Using a search *algorithm* can enable a tractable approximate model of the search *processes* that occurs in dynamical systems that can support persistence ratchets.

thing that must be preserved in this approximation is the persistence ratchet as this is the key to why the system was interesting in the first place. The best way to achieve both of these goals is to remove from the system RCPROs that have just had deleterious mutations, and thereby remove all of the calculations required in modelling their decreased persistence (figure 16). In this way all of the attention of the model can focus on RCPROs that maintain the same level of persistence or higher levels of persistence, hence preserving the persistence ratchet. Alternatively we could keep only the best N RCPROs. This requires some measure that can be applied to RCPROs to see which are the ones that have a higher than average ability at persisting - a higher fitness. Those with lower fitness are removed from the simulation as they are unlikely to lead to higher levels of persistence and are therefore considered to be a waste of CPU cycles. Hence we have an approximation of a system with exponential growth that works by concentrating the attention of the simulation on only those RCPROs with higher than average fitness - a search algorithm.

As with any approximate method, such a search processes cannot capture all of the subtleties of the system that it models. The quality of the approximation will depend heavily on the quality of the fitness function that assesses the ability of RCPROs at persisting. If this fitness function is very accurate then the approximation should be effective. The difficulty is that a good fitness measure would be able to assess the *ability* of a given object at persisting which is much harder than simply observing how well on object persisted in the past. However, even the a poor fitness function based on past persistence performance of objects would be able to preserve a persistence ratchet and thereby give us an approximation of the persistence ratchet achievable through exponential growth.

So this is my argument for why a search algorithm is likely to be the first type of engine for a dynamical system used to model autopoiesis. This is not because I think it would be the best way to model autopoiesis, nor is it because autopoiesis will require a search process and therefore why not use a search algorithm. The argument is simply saying that for the time being, a search algorithm will be the only way of modelling a dynamical system with a persistence ratchet that can support emergent autopoiesis in a tractable way.

5 Conclusions

This paper has introduced persistence as a very interesting characteristic of dynamical systems. It has been shown how a persistence measure applied to the whole state of a dynamical system can show us whether or not it is displaying search like dynamics. The persistence of objects within dynamical systems has then been used as the basis for an argument about the kinds of dynamical systems that could be capable of supporting emergent autopoiesis. With this argument came the idea of the persistence ratchet that is necessary to support increasing levels of persistence in generally persistence unfriendly dynamical systems. This persistence ratchet can either be manifested as exponential growth in the number of objects involved in the system or by a search algorithm to approximate this growth.

Future work will try to look for dynamical systems that can support RCPROs as well as static objects, but where most objects do not last forever. For this work a measure of cyclic persistence will have to be developed. It should be capable of measuring the persistence of objects within the state and not just the level of persistence of the whole state.

Reference

Berlekamp, E., Conway, J., & Guy, R. (1985). *Winning Ways. For your mathematical plays.*, Vol. 2: Games in Particular. Academic Press.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Kauffman, S. (1993). *Origins of Order.* Oxford University Press, New York.

Maturana, H. R., & Varela, F. J. (1980). *Autopoiesis and Cognition: The Realization of the Living.* Reidel Dordrecht.

Syski, R. (1992). *Passage Times for Markov Chains.* IOS Press.