

# A framework for studying the increase of complexity of interacting finite state machines

Lavoué F, Boschetti F, Ryan A, Grisogono A.

## DRAFT

### Abstract

In natural and artificial systems, both costs and benefits are associated with increased complexity. Here we investigate this tradeoff for simple machines in a discrete spatial resource field, where complexity is quantified as the entropy of the distribution of the states for that machine. The finite state machines can store different length histories, but machines can only increase in history length through reproduction. Notably, selection and fitness of the machines are intrinsic to the system's dynamics. For the resource field under consideration, we show that there is a peak complexity that maximises machine survival; that maximum complexity does not correspond to maximum history length; and characterise the dynamics of complexity for different history lengths.

## 1 Introduction

“All life is problem solving” [1]. This statement very much summarises our current understanding of biological evolution within a Darwinian framework and addresses the question of why Nature has produced structures of a complexity far exceeding the most complex engineering system we can envisage; organisms of increasing complexity can solve more difficult problems and consequently have an evolutionary advantage over simpler organisms. In turn, since organisms compete to share resources, their increasing complexity poses increasingly difficult problems to their competitors, thus requiring further complexity increases to survive<sup>1</sup>.

The above statement also underpins current approaches to numerical optimisation and Artificial Intelligence, which inherit ideas inspired by natural processes, like Genetic Algorithms [6], Swarm Intelligence [7], Neural Networks [8], etc. All these approaches place their emphasis on a population of agents and rely on the dynamics of the overall population in order to solve the problem at hand. In standard implementations, the individual components are simple and their structure/complexity fixed in time. Evolution, adaptation, learning and problem solving are achieved by the population of agents, not by the agents themselves and consequently the analysis of the underlying process is also devoted to the dynamics of the population, not of the individuals. Indeed, evolution and learning are said to arise in the population from the interaction among the individuals and the feedback provided by the environment (cost function).

In the above mentioned software tools, information processing is in a certain sense ‘purposeful’. The user has a specific problem to solve, represents it in the form of a cost function and then employs the population of agents to explore the peaks and

---

<sup>1</sup> For a nice description of the relation between complexity, problem solving, information processing and evolution see [4]

valleys of the resulting solution surface. The shape of the solution space (the world the agents live in) and the purpose to explore it are thus externally given. A large section of today's scientific community has a less purposeful view of Nature's working and of biological evolution itself. As nicely summarised in Kauffman [5], some autocatalytic chemical reactions happen to provide an avenue for replication, which is the most basic form of survival. Roughly put, this kick-starts evolution, that is, the process of selecting the most efficient and effectively reproducing agents. To be selected agents need to have survival advantage and since an efficient use of energy provides a survival advantage, they benefit from developing more complex structures to manipulate and exploit energy flows. This, albeit very simplified view, leads to an image of biological evolution whose dynamics is intrinsic and spontaneous and in which complexity becomes a selective advantage. Most important, purpose is self-referential and also intrinsic; all that matters is survival and replication, which become both the engine and the purpose of the dynamics.

Within this context lies Crutchfield and Görnerup's [2,3] pioneering work, in which they study the generation and dynamics of complexity in a population of agents. In their work agents are elementary units of information processing ( $\epsilon$ -machines, that is, finite state machines as defined in the Computation Mechanics literature [9]), which can interact with other machines. Their modelling environment does not have an explicit externally given purpose: if an  $\epsilon$ -machine arising from the combination of two parent machines can perform information processing then it will survive. Conversely, if the mechanics of information transmission is such that no information processing is possible, then the machines will die; selection and purpose are intrinsic.

Crutchfield and Görnerup [2,3] do not include a concept of space in their modelled world, they simply focus on the information processing arising in the population of machines. They study under what conditions the average complexity of the population can increase and under what conditions such increased complexity can be maintained. One of their main results is that the increase in the average complexity of the population is favoured by the low complexity of the individual agents; complex agents are not needed to generate complex population dynamics.

A natural question arising from their study is why in Nature individual agents, and not only populations, undergo a considerable increase in complexity via natural selection. A common speculative answer is that all agents we see in Nature effectively consist of much smaller units, and consequently can themselves be seen as ensembles of more elementary agents. Dismissing the circular question of what should then be considered as the basic 'unit' agent, in this work we borrow from Crutchfield and Görnerup's work and we ask under what conditions the individual agents can display an increase in information processing complexity and whether a relation between the increase in complexity of individual agents and of the population can be found.

## 2 Problem setting

We model agents representing simple organisms. The agents live in a 2D world, defined by a square grid. In order to model a finite world the edges of the grid are hard 'walls' which the agents can not cross. The grid consists of a 'resource field' which defines the amount of resource available at each location at the beginning of the

simulation (different resource distributions are used in our tests, see Figure 1). Agents forage their world for resources. They do so by moving in 4 directions (up, down, left and right) one grid step at the time, or choosing to not move and stay at the same location.

A first crucial feature of this work is the following circularity:

- agents need resources to carry out information processing; and
- agents carry out information processing in order to find resources.

As in Crutchfield and Görnerup's work, purpose is thus intrinsic in the survival of the agents; no externally defined purpose for the agents' action (and consequent evolution) is given.

Each move of an agent involves a cost and a potential reward. The cost is proportional to the information processing needed to decide what move to carry out, as will be discussed below. The reward is given by the amount of resource found at the location the agent moves to. A move is judged to be successful if the balance between reward and cost is positive. As in any reinforcement learning approach, successful moves are rewarded and unsuccessful ones are penalised.

The agents' decision process is made less trivial by two more features:

- the resource field is not renewable. The first agent which reaches a grid location can access up to a fixed amount of resource; agents reaching the resource at later stages can access only what resource is left;
- a fixed amount of resource reaches the grid at each time step (this is meant to model radiation from the sun). This amount of resource cannot be stored in the resource field but can be accessed by the agents. Consequently the action of not moving may also be successful.

While they forage for food, agents may meet at a grid location. When this happens they mate. Mating is the only process which allows for the information processing capability of an agent to increase (see Section 3.3 for details). We assume that information processing requires some sort of effort, thus an agent carrying out more information processing requires more resource in order to decide what move to take next. However, we conjecture, more information processing should empower an agent to take more 'informed' decisions (by better accounting for the information accumulated in its past history) and consequently should lead to a better model of the unknown environment. Thus, the questions we want address are:

- does this setting allow for an increase in complexity for individual agents?
- does the benefit arising from more information processing compensate for the increase in information processing cost?
- is an increase in complexity evolutionary useful?

### **3 The agents**

Our agents are approximations of the finite state machines defined in the Computational Mechanics literature [9]. They carry out 5 actions:

- 1) they forage by moving in the 2D grid;

- 2) they process information in order to decide how to move;
- 3) they store resources;
- 4) they mate;
- 5) they die if they run out of resources.

Here we briefly describe these different actions.

### 3.1 Movement

At any one time each agent needs to choose one out of 5 possible options: stay in place, move left, right, up, down; each option is given a symbol, from 0 to 4, respectively. Movements are chosen randomly. At the beginning of the simulation each agent has equal probability to choose any option ( $P_{0..4} = 1/5$ ). During the simulation successful moves (moves which result in an increase in the agent's resource level) are reinforced by increasing their probability to be chosen at the next step. Similarly unsuccessful moves are penalised by decreasing their probability. Over time, all moves are still available to each agent, but we expect successful moves to become progressively more frequent and unsuccessful moves to become progressively less likely to occur.

### 3.2 Information processing and agent's complexity

This is the crucial component of the simulation framework because it controls how an agent decides to move and consequently how it forages for the resource. This information processing is carried out by a finite state machine (associated with each agent) which includes one or more available states. A state comprises a number of moves and a probability distribution over the next moves. Following Crutchfield and Young's definition of statistical complexity [9], we define the complexity of a finite state machine (and of the associated agent) as the entropy of the distribution of the states for that machine.

At initialisation all machines are in State0. State0 includes the null history [\*] and a uniform probability distribution for the next move,  $P_0 = [p_{left} = .2, p_{right} = .2, p_{up} = .2, p_{down} = .2, p_{stay} = .2]$ . When a machine is in State0, it does not need to analyse its past moves and takes the next move fully randomly. Because it has only one state (and thus the probability of being in that state is 1), its complexity is zero.

During the simulation, an agent determines what state it is in by matching the history of the past steps it took with the moves stored in its machines.

#### Definitions:

- A *step* is the action of transitioning from one cell to another (0= "don't move", 1="move left", 2="move right", 3="move up", 4="move down");
- A *move* is a string of steps (symbols in the alphabet 0..4);
- The *future* is represented by the last symbol in the move;

- The *past* is represented by all symbols in the move which come before the future;
- A *history* is the string of last steps taken by an agent;
- The *history length* is the number of past steps an agent remembers, that is, number of symbols representing past steps which an agent keeps in memory;
- A *machine* is a set of moves, each move being associated with a probability of being chosen;
- A *state* is a set of moves within a single machine which share the same probability distribution over the future.

For instance, a move can be given by the string '111|2'. Here '2' is the future and '111' in the past. If an agent last's three steps have been 'move left', 'move up' and 'don't move', then its history is '130'. In this case the history length is 3. For computational reasons we limit the agent's memory to a maximum of 6 symbols.

If the last set of steps an agent has taken (history) matches a past in one of the machine's states, we say that the agent is in that state. Then, the agent chooses the next move according to the probability distribution which characterises that state. As an example, suppose an agent *A* has history length 1, has just moved right ( $A_{history} = 2$ ) and its machine is characterised by 2 states:

- *State1*, with past  $State1_{past} = [0,3]$  and probability distribution over the next moves  $State1_p = [p_{left} = .6, p_{right} = .1, p_{up} = .1, p_{down} = .1, p_{stay} = .1]$  and
- *State2*, with past  $State2_{past} = [2,4]$  and  $State2_p = [p_{stay} = .3, p_{left} = .1, p_{right} = .3, p_{up} = .2, p_{down} = 0]$ .

Then, the agent *A* is in *State2* (since  $State2_{past}$  includes  $A_{history}$ ) and will choose the next moves according to  $State2_p$ .

It is possible that during the simulation an agent's history does not match any of the pasts stored in the states of its machine. In this case the state the agent is in is chosen stochastically, with probability inversely proportional to the distance between the agents' history of moves and the pasts stored in the states.

### 3.3 Mating

Mating occurs when two agents meet at the certain location. This results in the generation of 2 more agents, whose machines combine the information in the parents' machines. Suppose agents *A* and *B* mate; 2 new agents *C* and *D* will be generated by combining the moves in *A* and in *B*. In order to keep the computational and memory requirement of the code reasonable, a maximum of 300 moves are assigned to each agent; also, in order to maintain diversity an agent will possess a minimum of 25 moves.

The moves of the child agents are obtained first by concatenating the moves of parent *A* with those of parent *B* and then by concatenating the moves of parent *B* with those of parent *A*. In order to concatenate move *m* with move *n*, the last symbols of *m* should match the *past* of move *n*; the future of the resulting move will then be the

future of  $n$ . As an example, suppose we have  $m=2|0$  (with probability  $m_p = 0.04$ ) and  $n = 0|1$  (with probability  $n_p = 0.05$ ). In order to combine  $m$  with  $n$ , we check whether the past of  $n$  (0), matches exactly the last step of  $m$  (0). Since it does, then  $m$  can be concatenated with  $n$ : the resulting move  $k$  will be  $20|1$ , with probability  $k_p = m_p * n_p = 0.002$ . Now we check whether  $n$  can be concatenated to  $m$ : the past  $m$  (2) does not match the future of  $n$  (1), therefore,  $n$  cannot be concatenated to  $m$ .

This method can be generalised to longer moves. It is important to notice the following:

- 1) by mating, the child machines increase the history length by 1 symbol;
- 2) because it may not be possible to concatenate certain moves, child machines may have either a larger or smaller amount of moves than the parent machines; and
- 3) once the mating is completed, the moves belonging to a machine are grouped into states and the machine complexity calculated.

### 3.4 Cost of moving

Suppose agent  $A$ , of history length  $A_{history\_length}$ , performs a move. We wish to assign to the move a cost proportional to the information processing required by the agent in order to choose what move to take. In particular, we define  $Cost = \alpha * A_{history\_length}$ , where  $\alpha$  is an a-priori set parameter. Basically the cost of the information processing is proportional to the agent's memory.

### 3.5 Resource storing

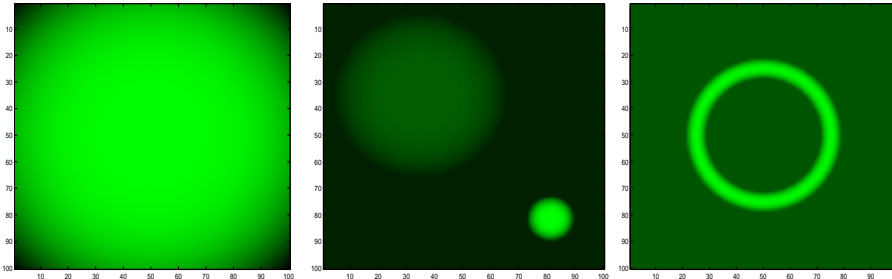
At the beginning the agents have no stored resource. During the simulation, each agent can store up to a maximum amount of resource  $Max_{res}$ .

When an agent reaches a location, it can store as much resource as is available, up to  $Max_{res}$ . The stored resource can then be used to fuel the information processing necessary to decide the next move.

## 4 Numerical results

We ran a number of simulations by employing three different resource fields, as shown in Figure 1. Each field is  $100*100$  pixels in size. Figure 1a shows a simple dome-like field, with maximum in the centre of the domain and was used merely to test the learning capabilities of the agents. Figure 1b mimics a basic test case for non-linear optimisation, in which agents may be attracted to a larger local maximum, thereby missing a narrower global maximum which would provide higher resource. Figure 1c shows a field in which the resource maximum is represented by a circular ring. Because this work does not directly address optimal resource exploitation, but rather dynamics of the agents' behavioural complexity, this last field is far more

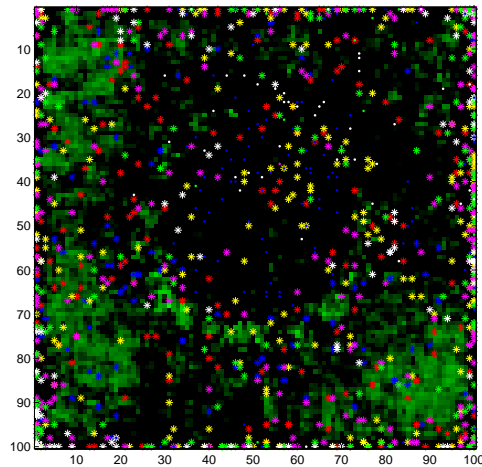
interesting. This is because once an agent has found the maximum resource by following a simple gradient, maintaining a high level of resource involves a fairly complicated sequence of steps (remember that the agents can move only vertically or horizontally, not diagonally). Finally, it is important to remember that the resource is not renewed, so the field depletes and constantly changes configurations according to the behaviour of the agents.



**Figure 1.** The three different resource fields used in our numerical simulation. In these and all following figures the resource field is represented by a 100\*100 pixel grid.

In all our tests we start with 200 agents. Agents mate and can replicate as explained in Section 3.3, and they die when they run out of resource. Consequently, the number of agents varies during the simulation. For computational reasons we limited the population size to a maximum to 1000 agents, that is, once this limit is reached new agents can be generated only provided others die. After extensive numerical testing on the resource fields in Figure 1a and b we established that the optimal values for the reinforcement learning module was to increase the probability of successful moves by multiplying them by 1.5 and decreasing them by multiplying them by .3. The other parameters used in our tests are  $\alpha=0.2$  (constant in the calculation of the cost of a move as in section 3.4), radiation from the sun = 0.1 and maximum resource a machine can store = 10.

In Figure 2 we can see a snapshot of a simulation after 1000 time steps on the resource field in Figure 1c. Agents are given a colour according to their history length (blue, red, yellow, purple, green, and white going from 1 to 6, respectively). Live agents are represented by a star, and dead agents by a dot.



**Figure 2.** Result of a simulation on the ‘ring’ resource field after 1000 time step. Agents are given a colour according to the length of their moves (blue, red, yellow, purple, green, and white going from 1 to 6, respectively). Agents which are still alive at the end of the run are represented by a star, the ones which died are represented by a dot.

A number of observations can be drawn from this image:

- 1) The resource field has been extensively harvested; some pockets of resources are left close to the corners of the domain and the original ring is barely perceptible.
- 2) Many machines are located close to the edges of the domain (remember that the edges represent walls which the agents can not cross). There are two main reasons for this to happen:
  - after the higher level of resource along the ring is exploited, the machines are led towards the border of the domain, where the remaining resources are still located.
  - Some machines may have undergone a loss of diversity in their available moves. When this happens, a machine will inevitably tend to follow a path which sooner or later will lead to a wall from which it won’t be able to escape.
- 3) The fact that the vast majority of the machines at the border of the domain are still alive seems to suggest that loss of diversity was not the main reason for their being located at the edge of the domain.
- 4) The vast majority of dead agents are blue (history length =1); 43% of blue agents died before the end of the simulation. Most of these agents died in the centre of the domain, not close to the border. These are ‘simple’ agents, which can remember only the last step they took in the past and can use only this information to plan the next move. This, roughly, equates to accounting for simple gradients and it is not surprising that these agents are unable to survive once the resource starts to deplete and the resource field becomes irregular.
- 5) 16% of white agents (history length=6) and 5% of green agents (history length = 5) also died before the end of the simulation. These are the agents with longer memory, that is, the agents for which information processing is most expensive. None of the other agents died. This seems to suggest that, for the field under analysis, moves of medium length give a reasonable compromise between the cost of information processing and the benefit arising from exploiting the information collected during the past exploration of the resource field.

#### 4.1 Complexity dynamics for groups of agents

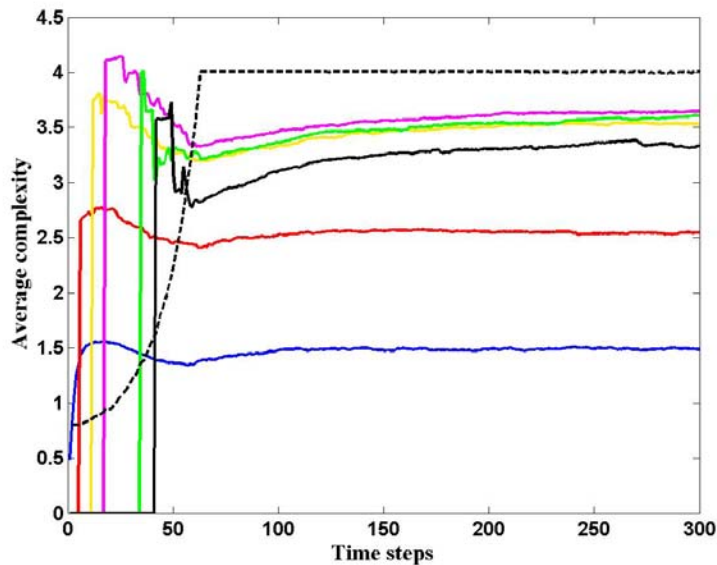
In Figure 3 we show a plot of the average statistical complexity of the agents, grouped according to their history length. A few patterns can be detected which were not necessarily obvious to expect:

- 1) each group of agents soon reaches a maximum complexity followed by a slight decrease. The most likely explanation for this behaviour is that at the beginning the machines try several different moves, some more successful than others and their complexity increases. After a while a few moves prove to

be more successful, they are chosen more often, and this leads to a reduction of variety and resulting decrease in complexity.

- 2) the complexity for each group starts to increase slowly again once the population size reaches the maximum and plateaux;
- 3) the relation between the history length and the complexity is not monotonic; the highest average complexity is given by history length of 4 (purple) while history lengths of 3 and 5 are very close (yellow and green respectively); the maximum move length (6, black in the plot) does not coincide with highest complexity.

The last point is particularly interesting; as we noted in Section 4, the optimal survival rate is obtained for move length of 2 to 4 and is also high for move length of 5. This seems to suggest that (with the exception of history length of 2) high complexity seems to favour survival.



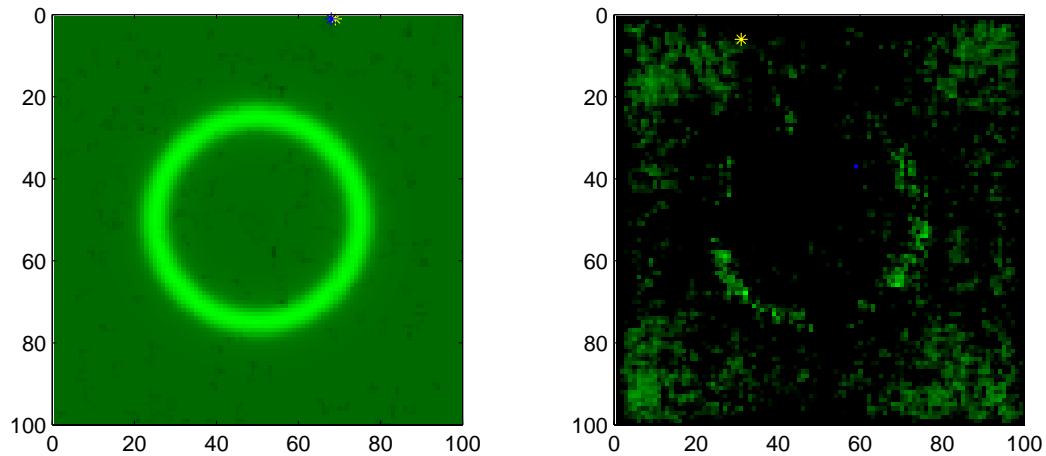
**Figure 3.** Average complexity for each group of agents. Agents are grouped according to the length of their moves using the same colour scheme as in Figure 2. The dotted line shows the number of agents during the simulation, starting from 200 (initialisation value) up to the maximum allowed of 1000.

## 4.2 Complexity dynamics of individual agents

In this section we analyse three individual agents in order to gain some insights into their specific behaviour and into the dynamics of their complexity. Obviously, since the agents' behaviour is stochastic, this analysis is only broadly indicative of the behaviour of the population as a whole.

In Figure 4a we see the usual resource field after 10 time steps. For clarity of exposition only two agents, one blue (history length =1) and one yellow (history length=3) are displayed, (the population size at initialisation was 200, as in the previous tests). The agents start in close proximity (the yellow agent was generated by the blue parent, at time step 10). In Figure 4b, we see the resource field and the two

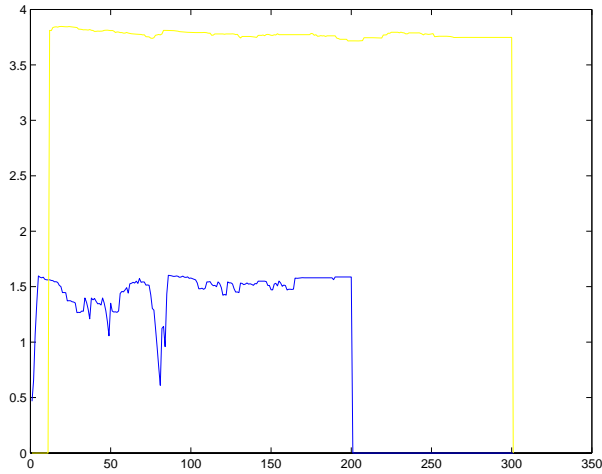
agents after 300 time steps. We can notice that the resource field has been extensively harvested by the population. The blue agent has died close to the centre of the resource field, which is almost completely depleted of resource. The yellow agent has managed to survive by foraging the resource field more extensively.



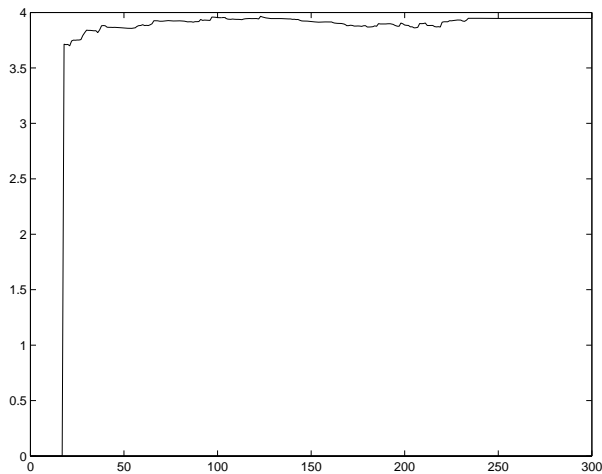
**Figure 4.** Two agents (blue and yellow, move length = 1 and 3, respectively) located in close proximity on the very top row at the 10<sup>th</sup> time step during a simulation (left). As in all previous runs, the simulation starts with 200 agents, not displayed for clarity of exposition. Resource field and agents' location are displayed after 300 time steps. The population of agents has harvested the resource field extensively. The yellow agent is still alive in the proximity of more resources, while the blue agent has died in resource depleted location.

In Figure 5 we show the dynamics of the complexity of the two agents. We can clearly see that the yellow agent has higher complexity and that the blue agent dies soon after time step 200. Of particular interest is the sudden drop in complexity of the blue agent between time steps 70 and 90. This drop in complexity coincides with the agents crossing the high resource ring. We can thus summarise the dynamics of the blue agent as follows:

- 1) at the beginning the agent explored its surrounding;
- 2) then, it learned and tuned into the main gradient;
- 3) it then aimed towards the top of the ring. At this stage the learning reinforces the use of mainly two moves (go down and go left), most other moves are not reinforced and this results in a complexity drop.
- 4) Once the peak of the ring is reached, the agents continue to use mostly the trained rules and consequently 'overshoot' by keeping on moving down and left towards a central area of low resource. At this stage the complexity is too low to master a different behaviour;
- 5) Once in a low resource area the previously successful moves are not useful anymore, their strength is reduced and the agent's complexity increases again. Now the agent explores its surroundings again, but soon the resource is depleted and the agent dies.



**Figure 5.** Complexity dynamics for the two agents in Figure 4.



**Figure 6.** Dynamic of the complexity of a yellow agent (history length =3) which crosses the ring towards time step 40. Unlike for the blue agents in Figure 5, no clear drop in complexity can be noticed.

The yellow agent in Figure 5 did not cross the high resource ring. In Figure 6 we analyse the dynamics of the complexity of a yellow agent which also crossed the ring towards time step 40. Unlike Figure 5, here we notice no clear drop in complexity. The longer history length, and the resulting higher complexity, allowed the yellow agent to maintain a richer set of available moves, and consequently, with no diversity loss, to keep on foraging the resources by following areas in which the resource had not yet been depleted.

## 5 Discussion and Conclusions

The main aim of this work was to extend the results presented by Crutchfield and Görnerup [2,3]. In particular:

- We provided a basic concept of space to our modeling environment.
- We extended the properties of the agents. In our model, agents do not only process information, but they also carry out actions (moving and foraging). As a result information processing is a means to an end, not an action for its own sake.
- Nevertheless, no external sense of purpose is given to the agents' action; agents process information in order to move, they move in order to feed and they feed in order to survive and process more information; this causal loop is self-referential and purely intrinsic.
- Agents start with full degrees of freedom available for their movement, in a most symmetric state, which results in zero complexity; future reduction of degree of freedom, and increase of structure and complexity is due solely to their interaction with the asymmetric resource field and with each other.

It is important to notice that our model is basically a closed system. The only exception to this is represented by a) the solar radiation and b) and the stochastic process used by the agents to choose the next action according to a given probability distribution. Since the solar radiation has low intensity, unable to sustain the agents population, our system is doomed to collapse once the agents have harvested all of the resource. As a result, we are effectively studying a transient process. While this is not a common procedure for agent based modeling, we believe this is particularly interesting and relevant to the analysis of several real world phenomena.

Our analysis focused mostly on the trade-off between cost and benefit of information processing. We assumed that storing more memory, and thus being able to learn more from previous experience (in our case from previous sampling of the resource field) involves a cost, since it requires a more sophisticated information processing hardware and internal resource which otherwise could be devoted to other uses. Consequently, the higher cost needs to be compensated by harvesting more resource; an agent with a more sophisticated information processing system can survive only provided it can store more resource than a simpler agent.

Our main results suggest:

- 1) that optimal performance (in terms of an agent's survival) is achieved by agents whose information processing cost is not too low and not too high;
- 2) that the relation between complexity and history length is not monotonic, the maximum complexity is achieved by history length of medium, but not maximum length;
- 3) there is an indication that optimal survival is obtained for highest complexity, but this conjecture is not definite and more experiments are needed to confirm it.

The final important result is that in our test, not only the overall population but also individual agents increase their complexity. This result has important potential implication for a large number of population-based algorithms. This leads naturally to the question of resource exploitation with direct applications to optimization problems. How this idea can be extended to search a potentially high dimension parameter space and how it can be employed to achieve optimal coordination among agents are questions worth exploring. An obvious immediate avenue for further

research includes testing the behavior of our population against standard non linear optimization tools like Genetic Algorithms and Swarm Optimization.

## 6 References

- 
- <sup>1</sup> Popper, K., 1999, All Life is Problem Solving, Publisher: Taylor & Francis, Inc.
  - <sup>2</sup> J. P. Crutchfield and Olof Görnerup, "Objects That Make Objects: The Population Dynamics of Structural Complexity", Proceedings of the Royal Society Interfaces 3 (2006) 345-349.
  - <sup>3</sup> Olof Goernerup and James P. Crutchfield, 2006, Hierarchical Self-Organization in the Finitary Process Soup, Santa Fe Working Paper #: 06-03-008
  - <sup>4</sup> Crutchfield, J. P. (1994) The Calculi of Emergence: Computation, Dynamics, and Induction. *Physica D* **75**: 11-54.
  - <sup>5</sup> Kauffman, S., 2000, Investigations, Oxford University Press.
  - <sup>6</sup> Goldberg, D. E., 1989, Genetic algorithms in search, optimization, and machine learning: Addison-Wesley Publ. Co., Inc.
  - <sup>7</sup> Mouser C., and Dunn S., 2004, Comparing Genetic Algorithms and Particle Swarm Optimisation for an Inverse Problem Exercise, The 12th Biennial Computational Techniques and Applications Conference, Melbourne, Australia.
  - <sup>8</sup> Picton, P., Neural networks, Basingstoke, Hampshire : Palgrave, 2000
  - <sup>9</sup> Crutchfield, J., and Young, K., 1989, Inferring Statistical Complexity, Physical Review Letters, 63, 105-108.