Simulating Adaptation to Environmental Change: Complexity and Organized Behaviour Within Environmental Bounds (COBWEB) *

Brad Bass
Adaptation & Impacts Research Group
Environment Canada at the University of Toronto, Institute for Environmental Studies
33 Willcocks Street, Toronto, Ontario, M5S 3E8 CANADA
Fax: 1-416-978-3884, Email: brad.bass@ec.gc.ca, http://www.msc-smc.ec.gc.ca/airg

Jeff Hill
Visual Infinity, Inc.
229 College Street, Suite 301, Toronto, Ontario M5T 1R4 CANADA
Email: jeff@visinf.com

Nina Suh
Division of the Environment, University of Toronto
33 Willcocks Street, Toronto, Ontario, M5S 3E8 CANADA
Email: ninasuh@yahoo.com

Abstract
The architecture, Complexity and Organized Behaviour Within Environmental Bounds (COBWEB), was developed to support an anticipatory approach to adaptation. COBWEB consists of a large number of autonomous agents, each a genetic algorithm, using different strategies to adapt to changing resource availability. Anticipatory genetic algorithms, that are Turing complete, as well as mutation are used to allow the agents to respond to a changing environment. The simulation has four attractors, which exhibit sensitivity to initial conditions, and the spatial patterns of the agents exhibit wide variation as well as local structure, which might indicate adaptive or anticipatory behaviour.
Keywords: genetic algorithms, anticipatory agents, learning, adaptation and Turing machine.

1 Introduction

The Framework Convention on Climate Change and subsequent accords have referred to the need to consider how various types of systems might adapt to climate change. Given the uncertainties associated with climate change scenarios, there are arguments for both reactive and anticipatory approaches to adaptation. Reactive arguments focus on letting adaptations occur as required due to the cost of some proactive adaptations and the uncertainties associated with specific impacts, particularly

when translated down to specific regions or locales. This approach also assumes that if the climate does change, these changes will unfold at a rate that will allow for the requisite adaptations. Anticipatory or proactive arguments focus on the risk of not taking action given the uncertainties, and acknowledge the potential for a rapid and discontinuous transition between the current and future climates.

Anticipatory approaches to adaptation require some form of predicting the future, and then using that future to change a broad range of government policies as well as organizational and individual behaviour so that they incorporate adaptations or the flexibility to be adaptable in the face of future climate change. For the most part, research in adaptation to climate change discusses the need for adaptation as well as specific measures for different regions or sectors. The architecture, **C**omplexity and **O**rganized **B**ehaviour **W**ithin **E**nvironmental **B**ounds (**COBWEB**) represents a different approach to studying adaptation. It is general simulation platform where autonomous and where anticipatory or adaptive agents act in a variable and changing environment.

The goal of COBWEB is to study the general characteristics and principles of anticipatory adaptation that might be applicable to wide variety of systems. The specific objectives of this phase of the project were to evaluate the behaviour of a simple system at the level of the agents and at the level of the system; to develop a method to simulate anticipatory behaviour; and to develop a software architecture that would be flexible enough to accommodate different research needs and different platforms. In this initial study the focus is on the behaviour of the system under different constraints in order to identify the emergence of anticipatory behaviour and its effectiveness under different constraints.

Anticipatory agents and anticipatory systems are defined as an agent or a system which contains a predictive model of itself or the environment at some future state and acts according to that prediction. Agents that manifest adaptive behaviour (Rosen 1985), which is incorporated in evolutionary computing (Bedau and Packard, 1992; Holland, 1993), can represent anticipatory behaviour. Bedau and Packard (1992) modelled multiple adaptive agents called Strategic Bugs, based on a set of rules, on a two-dimensional grid with a renewable food supply. Holland (1993) used genetic algorithms in the Echo model to represent agents in this environment and also introduced interactions amongst the agents as well as multiple resources.

COBWEB is a mix of these two approaches. It is very similar to the Bedau and Packard grid except like Holland, the agents are represented by genetic algorithms. Like Bedau and Packard, the agents consume one resource, although this has been modified in the next version of COBWEB. A cellular automaton models the resource growth, although resources can appear at random locations at any time with a small but fixed probability. As the location of resources is constantly changing, the environment is in a state of constant flux. Unlike the Echo model, the only interactions between agents are collisions, which require an expenditure of energy.

A similar approach to modelling anticipatory economic systems using classifier systems has been proposed by Rivero et al. (1999) and a classifier system was used for a single agent in an ecological context by Krebs and Bossel (1997). A classifier system representation of an agent uses many genetic algorithms or strategies, and auctions are

held to determine which strategy the agent will use at any one time. Lavigne (2001) has suggested a similar approach to modelling anticipatory behaviour using neural networks. The COBWEB architecture will allow agents to be represented by classifier systems, neural nets or other inference engines.

COBWEB agents are able to replicate if they have sufficient energy resources. Those agents that do not have strategies that allow sufficient consumption of resources do not replicate and are eliminated from the system. COBWEB can support several unique genetic algorithms and is able to evaluate several strategies at once. Hence the agents that are selected represent a system-wide response to the expected level of resources, the expected cost of movement and replicating and even the expected location of resources. Mutation is very important because at the system level, it allows for the experimentation with new strategies that may be necessary to survive in a changing environment.

To introduce anticipatory behaviour, defined as acting on a predictive model of a future state (Rosen, 1985), anticipatory agents were developed by allocating a small amount of memory, that is updated with new information after every action and can result in new behaviour. The full implementation is discussed in the following section. Mutation could also allow for latent strategies to re-emerge in the environment. If the environment changes making that strategy useful again, it is available to the agent, and through replication, to the system. This is akin to the function of biodiversity or any sort of societal information storage system, and potentially allows for anticipatory behaviour or adaptation to emerge at the system scale.

COBWEB has many of the features of adaptable and evolutionary behaviour that Rosen (1985) discussed in linking anticipation to adaptation. Specifically, it has genes, or genotypes, acting on a landscape, and the use of memory allows for the emergence of adaptive behaviour. Although Rosen used a biological metaphor of genotypes and phenotypes, his intent was to provide a generic description, which is the intent with COBWEB as well. COBWEB does not mimic any system in particular, and it would have to be modified for a specific application. Rather the intent is to examine the linkage between anticipatory agents and system behaviour in a variable and changing environment.

The output from the current version of COBWEB consists of the time step, the number of grid cells occupied by food, the number of agents, the average agent energy and the total energy summed over all of the agents. Despite the simplicity of the initial COBWEB platform, it still exhibits nonlinear behaviour, chaotic attractors, semi-stability (Bass et al., 1998) and multiple attractors. Although the most common outcome is a steady state cycling of food and agent populations representing a few successful strategies, there are initial conditions that will lead to the elimination of all the agents, unlimited growth of the agent population, and survival of all strategies. The spatial patterning of the agents is not predictable, but with the first attractor there are indications that pockets of local spatial structure emerge with some degree of stability.

2 Implementation of COBWEB

Unlike many other simulation models, COBWEB was implemented as a Java application primarily because Java programs can execute on nearly any host platform. While other environments share this cross-platform facility with Java, few are as capable at producing dynamic graphical output on as wide a variety of systems. Traditionally, programming languages tend to leave the question of interface to the platform, leading to a wide range of platform-specific user interface libraries. With Java, this interaction is standardized, leading to the ability to develop a visual presentation of the simulation which remains the same across multiple host environments.

COBWEB also makes use of the dynamic class loading functionality present in Java, so that new simulations can be added to the system without any modification to the driver code. The ability to embed Java applets in web pages allows for the presentation of results on the web with actual live simulations as diagrams. While it is true, that when using any interpreted language there will always be some performance penalty when compared to a compiled language, the performance of the Java language was very acceptable for COBWEB, even with early prototypes.

The architecture used to implement COBWEB is very much an object-oriented system. The application itself simply consists of a number of housekeeping and presentation classes, and the logic to load a data file. The classes that actually implement the simulation are referenced by name from the script file, which is loaded at start-up along with a configuration file that contains the values for various simulation parameters. This clear distinction between the application and the simulation has proved very useful in reducing the complexity of the implementation, and increases the potential flexibility of the system. The most basic class in the system is the Environment class, which implements the two-dimensional grid, upon which the agents interact, which contains rocks and resources. A non-deterministic cellular automaton drives the growth of the resource; any resource not consumed by the agents at any time step may reproduce to adjacent grid locations, with a probability that is set in the configuration file for the simulation.

Given an Environment class, an arbitrary number of different Agent classes may be defined. An Agent executes the allowable behaviours for an actor in the system, defining both the perceptual input and the possible actions that may be taken on any time step. It is possible for multiple heterogeneous agent types to be present in the same environment, in the same simulation, heterogeneity being defined by a different genetic code, which allows for competition between physically heterogeneous populations. Furthermore, an Agent class does not specify a policy regarding the decision of what action should be undertaken on each time step; this decision is deferred to an AgentController instance. Just as heterogeneous agents can populate an environment, any number of AgentControllers can control any class of Agent, though only one AgentController may drive a single Agent at any time. This allows for the direct modelling of competition between physically identical, but algorithmically or mentally different actors.

An AgentController is a pure representation of the intelligence of an actor. The interaction between the Agent and the AgentController is defined in a simple bitwise manner where the Agent supplies a fixed number of bits of input, and requires a fixed number of bits of output. This allows for the AgentControllers to be designed in a truly abstract way, so that the complex and error prone code required to implement various AI algorithms need only be written once and shared between multiple Agent types or simulations. Due to the abstract nature of the AgentController class, any number of abstract algorithms such as simple rule sets, neural nets, genetic algorithms or classifier systems can drive the actors' actions.

The primary AgentController class in the current versions of COBWEB is a genetic algorithm. The implementation of the genetic algorithm is straightforward, containing a table that directly represents a translation between the input and the output. When Agents replicate, this table is copied to their offspring, with a random chance for bitwise mutation. An interesting extension to the standard genetic algorithm that was used in COBWEB was to provide a limited memory to the AgentController. This was implemented by providing an additional number of bits of both input and output, beyond what is strictly required by the Agent. On the first simulation timestep, this additional input space is simply set to zero. However, the extra output is saved, and is used in this input space on the next timestep. Thus the AgentController has a fixed space where information can be passed to itself to change its behaviour at the next time step.

More formally, this allows the input to become an encoding of a future state into a model that is held in the agent's memory, which functions as a prediction and alters the behaviour agent. In this version, a 12-bit genetic algorithm, with a random allocation of one's and zeros, and a two-bit memory, define the agents. Of course, the complexity of the algorithm is limited in practice by the number of memory bits allowed to each AgentController, but the encoding of additional information is an interesting result in and of itself as it creates genetic algorithms that are Turing complete. The ability to process additional information is also one of the characteristics of a truly complex system.

The ideal solution for any agent is trivial — eat if possible, else move or else turn — but it allows for the development of a genetic algorithm library to drive the agents, as the base model for future development. The actual number of genetic algorithms is equivalent to the number of agents in the environment at any particular time step. In order to view the evolution of strategies a hashing procedure was used to create colour identifiers, based on how the actor moved in the environment.

3 Running the COBWEB Model

There are several components in the COBWEB environment. As previously mentioned, it includes agents that can move, consume resources, give birth, expend energy, die and potentially anticipate the future. The accumulated energy represents the "life force" of the agent. When this score drops below 0, the agent dies, and is removed from the simulation.

In the first version of COBWEB, birth occurs as soon as enough energy is accumulated. In the second version, a gestation period is introduced, thus introducing additional uncertainty into the simulation as an agent could die before giving birth. An agent is restricted in that it cannot see past one grid square nor can it communicate with other agents. The initial number of agents, the amount of energy required to accomplish any task, the initial energy available at birth, the mutation rate, and whether the memory is in use can be changed before running the program. The time steps are defined by every new appearance of the resource in a grid cell that was previously empty.

At the start of each simulation run, the user can specify several parameters such as the size of the grid, the number of agents, the number of stones, the amount of energy obtained from food and expended on each activity, and the rate of food growth. Both means of controlling the rate of growth, and hence the distribution of new resources, can be altered at the beginning of the program. The grid is two dimensional, but the platform is flexible enough to allow for a three-dimensional grid in the future. Stones, which are immovable objects that extract a certain amount of energy per collision, are also distributed randomly in the grid. The number of stones and the energy expended by a collision is chosen at the beginning of the program. The user can also specify whether the agents have memory and the rate of mutation.

4 Analysis of COBWEB

For most parameter values, the initial behaviour of the system through time oscillates between extremes for agent population and the available resources. At 200 time steps the agent population may decrease quite rapidly, sometimes falling to one individual, or the system may reach its steady state. This is the phase where the system eliminates most of the strategies. After 200 time steps three attractors emerged depending on the initial values of the parameters. The predominant attractor resembled the classic predator prey pattern in ecology where both the agent population and resources oscillated in a very narrow range in what resembles. The spatial patterns were much more chaotic with pockets of order emerging, that are stable or at least semi-stable (Bass et al., 1998). A second attractor resulted in the elimination of all agents by 800 time steps, and a third attractor was unlimited growth in the agent population until the agents filled up the grid.

The temporal behaviour of COBWEB was analyzed by adjusting the parameters individually and moving them together, and allowing the simulation to run for 3,000 time steps. Parameters were adjusted between a high, intermediate or low value. The intermediate values were set at a level that resulted in the stable predator-prey interaction and the extremes were set at arbitrarily high and low values. The parameters were increased and decreased by small increments until they upper and lower limits were reached on a fixed 75 x 75 two-dimensional grid.

The objective of these tests was to determine the number of attractors in COBWEB and the degree to which an attractor, not a specific trajectory of any one simulation, could be predicted from the initial values. Two sets of experiments were run with two different sets of initial parameter settings. The first experiment produced the first two

attractors. At the first attractor the steady state agent population usually hovered between 80 – 90 regardless of the parameter values.

Most of the initial parameter values drove the system towards the first attractor, which is easily detected at the system level, through the number of agents, and strategies that persist in the environment. When the system reaches its steady state, the genetic variety has been reduced such that the population is dominated by at the most four different strategies, though the exact balance between them is not predictable. In some simulation runs they four different types of genetic algorithms may be present in equal numbers while in others, one or two may dominate in the environment.

The first attractor was reached within a wide range of parameter values, and the agent population, energy and available resources usually varied within a very narrow range. The transition from the convergence on a steady state to the elimination of all the agents occurred quite suddenly with subtle shifts in the initial parameter values at the boundary or threshold of this range. While some of these threshold values, such as the number stones, are sensitive to the size of the grid, others clearly reflect the limits of the strategies that are available to the system. Although the mutation rate and anticipation offer the means to increase learning and vary the strategies, in this case, too much variation in the ancestral strategies could not be tolerated. In other words, the epigenetic landscape was severely constrained (Beer, 1981) due to the apparent stability, or lack of variety, in future resource distribution.

There were several parameters for which thresholds existed that overwhelmed the available strategies to the system and drove the system to the second attractor (Table 1). The agents were eliminated between 800 – 1000 steps when the number of stones exceeded 600. The agent population also died out when the resource rate of the cellular automaton was reduced to 50%, when the initial energy fell below the intermediate value, when the food energy was decreased to 100 units, and when the turn energy was set to 30. The impacts of other threshold values, such as mutation rate and resource energy, were unpredictable. For example, when the rate of mutation exceeded 0.06 the agents were eliminated in some simulation runs, but not in others, although at very high levels of mutation, all of the agents disappeared at before 600 time steps. There were other parameters, such as the energy required to breed, for which the equilibrium number of agents were reduced but not eliminated from the grid.

Table 1: High, Intermediate and Low Values of Critical Parameters (First Set of Experiments)

	Stones	Resource Rate	Mutation Rate	Initial Agent Energy	Resource Energy	Step Energy	Turn Energy
High	1000	2.0	2.00	2000	200	100	100
Intermediate	300	1.0	0.05	1000	100	5	5
Low	1	0.5	0.01	2	3	1	1

The second set of experiments were run with a different set of high, low and intermediate values. Table 2 indicates those parameters that were critical and Table 3 indicates some of the larger differences that were introduced between the two experiments. There were some notable differences and similarities between the two sets of experiments. In the second experiment, the average steady state agent population was 350, but the range was much larger, ± 100. In the first experiment, a dramatic cull of strategies and agent population occurred at time step 200, and this was visible both in the graph of population with time and on the two-dimensional grid. In the second experiment, the dramatic cull occurred at time step 200 but was only visible on the grid at time step 400. The third attractor only emerged in the second set of experiments.

Table 2: High, Intermediate and Low Values of Critical Parameters (Second Experiment)

	Resource Rate	Resource Growth	Mutation Rate	Resource Energy	Step Energy	Turn Energy
High	3.0	1.0	1.00	500	30.0	30.0
Intermediate	2.0	0.5	0.05	100	5.0	10.0
Low	0.2	0.2	0.01	3	0.5	0.5

As can be seen in Table 2, parameters such as the number of stones were no longer critical in terms of system viability. The system was sensitive to a threshold value of 0.5 for resource growth, the probability of new resources appearing at random locations, at which point all the agents were eliminated. At the other end, the average steady state agent population suddenly increased to 625 when food growth increased to 2. When step rock energy and turn energy were set to 0.5, the agent population did not reach a steady state, but instead continued to increase until the grid was filled with agents. When this simulation is allowed to continue, the system will crash, not due to resource scarcity, but due to a scarcity of space.

Table 3: Parameter Differences between the Two Sets of Experiments

Experiment 1 to 2 (± change)	Stones	Initial # Agents	Initial Resources	Initial Energy	Resource Energy	Step Energy
High	1000 – 2000	2000 - 1000	3000 – 2000	2000 - 3000	200 – 500	100 - 30
Intermediate	300 – 700	1000 - 100	1000 - 500	1000 - 2000	No change	5 - 10
Low	1 - 10	2 - 10	5 - 20	2 - 30	No change	1 – 0.5

In this set of experiments, increasing the mutation rate appeared to lower the agent population, but the pattern was not regular. However, as the rate was increased from 0.25 to 0.75, the average steady state population decreased from 1300 to 100, but the average energy remained slightly higher than 1000. Thus, although increasing the mutation rate appears to have a detrimental affect on the system in the long run, the changes in the initial parameters allowed a much higher degree of trial and error. When groups of parameters were moved in unison, the food parameters had the most consistent impact. Regardless of the other parameter values, when the food parameters were set at their low values, a number of the agents were eliminated from the grid. Raising the values of other parameters could delay the elimination, but the food parameters were the only set of parameters to consistently thwart the system's ability to adapt.

A preliminary set of experiments has been conducted by primarily increasing the amount of energy available at birth and lowering the cost, in terms of energy expended, of movement about the grid, resulting in a fourth attractor. This attractor allows for most, or at times even all, of the strategies to survive. Its emergence appears to be based on highly mobile agent that consumes a lot of the resource, replicates often and will occasionally replicate other strategies through mutation, which survive due to their high initial energy allocation. Large increments in the cost of movement will induce a shift to the second attractor, but this may occur with even small increments in this parameter. At certain threshold values, the emergence of either the second or the fourth attractor is not predictable, and small changes in other parameters may cause a shift from the fourth to the third attractor.

5 Discussion

The analysis of the temporal patterns in COBWEB revealed that the system is quite robust to the changes in resource distribution as well as changes to the initial parameter settings. However, the system's response to these settings is nonlinear as indicated by the sudden shift to a new attractor below or above certain threshold values. The success of any strategy, or type of genetic algorithm, is constrained by the values of the initial parameter settings. The initial values create a large window of opportunity in which some or all of the strategies are successful whereas just outside that window a different attractor may emerge.

At other times, exceeding the threshold values did not result in the elimination of the agents. The population might have decreased dramatically, but it recovered and the predator-prey pattern re-emerged over time. The specific attractor could not be predicted from the parameter values, these uncertainties might be indicative of the presence of chaotic attractors at certain threshold values.

It could be argued that the stable predator-prey pattern that emerges is representative of successfully anticipatory behaviour at the system level in that the strategies that survive represent models of decision making that are appropriate for survival. This is especially applicable when the initial parameter values for the simulation would usually result in the elimination of the agents, yet the agent population

is able to recover; the system has successfully adapted to the environmental constraints. Other strategies will re-emerge, but their elimination indicates a system-wide anticipation of a future where those strategies would not be conducive to maintaining the current agent population. Although COBWEB contains no higher order agents to direct the system's choices, emergent behaviours above the scale of the individual are often interpreted as system responses in a wide variety of disciplines.

The spatial patterns have not yet been analyzed in a formal manner, yet in many simulations the agents exhibit clustering patterns, often by colour group. They no longer traverse the whole grid in search of resources, but rather remain in a smaller domain with sufficient resources to meet their energy requirements. This suggests that if the system is at the first attractor, behaviours are emerging that narrow the search radius as long as resources appear in that section of the domain while other parts of the grid appear to remain unstructured or perhaps chaotic. These behaviours are consistent with what would be expected from anticipatory agents as well as observations in ecology (Hassell et al., 1991) and similar modelling studies (Kauffman, 1993). Although this behavioural modification may be a result of the anticipatory nature of the genetic algorithms, mutation may also play a role, although the clustering was more difficult to detect when the memory was not activated in the genetic algorithms.

The third and fourth attractors, unlimited growth in the agent population and the persistence of a wider range of strategies, emerge under certain conditions. The third attractor emerges when certain, but not all, energy expenditures were reduced below a critical threshold. For example, in the first set of experiments, when the energy requirements for movement and replication were sufficiently low, the steady state agent population at the first attractor was far below the average or the agent population would be eliminated. In this case, the rate of population growth exceeded the rate of resource production. In the second set of experiments, when the replication and movement energies were reduced to similar levels, the rate of resource production was sufficient to support an ever-increasing population, limited only by the size of the grid.

The nonlinear response to changes in initial parameter settings raises some interesting questions about anticipation. If the environment changes too drastically, will it overcome the system's ability to anticipate the future, i.e. to develop new strategies to ensure its survival? This question affects all applied research in adaptation, particularly climate change, in that many of the conclusions drawn form impacts and adaptation research are based on a gradual transition from one climate to the next.

The second question is the importance of anticipation. COBWEB agents are anticipatory, but it is not yet clear that it confers a distinct advantage when the best strategies for survival are fairly simple. The impact of memory in the individual agents was not discernible in these simulations. However, COBWEB was reprogrammed to allow only some fraction of the agents to have this memory. When the fraction was 85% or higher, the system eliminated those agents without memory. Below the 85% threshold, the system tended to favour those agents without memory. The next version of COBWEB allows for the memory to activated in only some of the agents, so that the importance of anticipation for survival can be evaluated under a wide variety of conditions.

Additionally, in this version, there are three types of resources and the simulation can be configured so that the agents' preferred resource confers more energy. The other major changes are that birth is not instantaneous, rather the agents are pregnant and, mutation cannot occur across resource preference groups, i.e. if an agent prefers the blue resource, it cannot give birth to an agent prefers the yellow resource. Initial runs with this version, indicate markedly different agent behaviours and a range of different outcomes, and the system takes much longer to reach this stable pattern. Initial testing with the same parameter settings suggests that the groups that are eliminated and the time required for elimination cannot be predicted in advance. However, as agents can be modified according to their resource preferences, the next version of COBWEB will allow a much richer set of experiments, including a direct comparison of agents with and without the capacity for anticipatory behaviour.

One of the hallmark features of COBWEB, that provides the basis for simulating anticipatory behaviour, is the memory that has been appended to the genetic algorithms. These agents are now able to encode input, perform arbitrary computations on the input, store intermediate results and generate output in the form of behaviour. Hence the genetic algorithms are isomorphic with a Universal Turing Machine (UTM), or Turing complete in an uncertain environment; essentially any algorithm, which can be implemented in any deterministic language, could mechanically be translated to this genetic algorithm just as it could to a UTM. Thus, in principle, this genetic algorithm can run any algorithm that is a model for anticipatory behaviour, although in practice it is limited by the size of the space that is allocated for its memory.

6 Future Research

There are several future research directions that will be explored over the next few years. These directions involve analysis of spatial patterns, analysis of the attractors, response to changes introduced during the simulation, additional environmental complexity and additional agent complexity. The spatial analysis will utilize both qualitative and quantitative measures to assess the significance of the clustering that has been seen to emerge at later time steps in the simulation. The analysis of multiple attractors will be continued, specifically at those threshold values that produce the second, third and fourth attractors.

Additional complexity will be added to the agents to allow for the emergence of different strategies, through the addition of breeding, communication, tenure of resources and a larger memory, or capacity for anticipation. Additional complexity will be added to the environment by introducing resource scarcity. As the ability to adapt to rapid environmental change is of major concern in climate change research, future versions of COBWEB will allow parameter changes during a simulation as well as at the initialization of each run. These modifications will provide a basis to explore the Law of Requisite Variety, only variety can absorb variety (Beer, 1981), and it will be interesting to explore how much variety is needed at the agent level to cope with additional environmental variability.

The COBWEB platform can be extended to experiments involving multithreading, running the simulation on multiple platforms. This would be accomplished through partitioning the world into 'processor blocks', where each n x n block of the world is a separate process. To facilitate this experiment, the borders between processes could be made more explicit making it expensive for agents to move across these borders. Ultimately, each agent could be modelled as a process with a separate process to run the environmental simulation and agent interaction. Although this would require synchronization at every time step, it would provide for a truly complex agent in a multiple agent world allowing for agents with substantially larger memories, on the order of 1024 bits.

7 Conclusions

COBWEB is a simulation platform for exploring questions about anticipation and hence adaptation to environmental change. It allows for a multiple agent simulation, penalties for making a mistake and the preservation of unused strategies. The system is able to learn and adapt, in an evolutionary manner, selecting the most viable strategies and the optimal number of agents. The system can also fail to adapt and the agents die, or maladapt leading to overpopulation.

Anticipation was embedded at the agent level through mutation and through providing each genetic algorithm with additional but unspecified storage space, allowing for a Turing complete computational model. COBWEB is still simple enough that a 'best' learning strategy does not have to be imposed upon the system; Darwinism accomplishes that task. Instead of explicitly defining what is best, which may corrupt the system, the best are those that replicate. Other strategies are stored for later use or become extinct. At smaller scales in the domain, the agents appear to modify their spatial behaviour, which may indicate a capacity to anticipate the location of a resource. Future versions of COBWEB will allow for a richer set of experiments and a comparison of agent success with and without anticipation.

Acknowledgements

The authors would like to thank the reviewers for their comments on a previous version of this paper and acknowledge the support of the Adaptation and Impacts Research Group, which provided the initial resources for the programming, the Division of the Environment at the University of Toronto for providing student support and Ian Burton, Roger Hansell, Jin Kang and Robert Medieros who provided assistance along the way.

References

Bass Brad., Byers Ralph E. and Lister Nina-Marie (1998) Integrating Research on Ecohydrology and Land Use Change with Land Use Management. Hydrological Processes 12, pp. 2217-2233.

Bedau Mark A. and Packard Norman H. (1992) Measurement of Evolutionary Activity, Teleology, and Life. Artificial Life II. Edited by C. G. Langton, C. Taylor, J. D. Farmer amd S. Rasmussen, Published by Addison-Wesley, pp. 431-461.

Beer Stafford (1981) Brain of the Firm, 2nd Edition. John Wiley & Sons.

Hassel Michael P., Commins Hugh N. and May Robert M. (1991) Spatial Structure and Chaos in Insect Population Dynamics. Nature 353, pp. 255-258.

Holland John (1993) Echoing Emergence: Objectives, Rough Definitions and Speculations for Echo-Class Models. Working Paper 93-04-023, Santa Fe Institute.

Kauffman Stewart (1993) Origins of Order: Self-orgenetic algorithmnization in selection and evolution. Oxford University Press.

Krebs Friedrich and Bossel Hartmut. (1997) Emergent Value Orientation in Self-Orgenetic algorithmnization of an Animat. Ecological Modelling 96, pp. 143-164.

Lavigne Frédéric Denis S. (2001, in press). Attentional and Semantic Anticipations in Recurrent Neural Networks. International Journal of Computing Anticipatory Systems.

Rivero Sérgio Luis de Medeiros, Storb Bernd Heinrich and Waslawick Raul Sidnei (1999) Economic Theory, Anticipatory Systems and Artificial Adaptive Agents. Brazilian Electronic Journal of Economics 2.

Rosen Robert (1985) Anticipatory Systems. Pergenetic algorithmmon Press.