Simulated Autonomous Agents Utilizing Instincts and Behavior Learning: Orgs in Orgland

George Chrysanthakopoulos & Robert J. Marks II
University of Washington, Department of Electrical Engineering, Seattle, WA 98195
georgioc@microsoft.com marks@ee.washington.edu

Abstract

This paper introduces a simulator of interacting agents wherein an artificial agent, dubbed an Org, can learn and accumulate experience built upon initially endowed primordial instincts. Building on these instincts, the Org develops experience, the ability to detect behavioral patterns and discover physical constraints through self-reflection. Emphasis is placed on the concurrent use of long-term planning and reactive behavior. The interaction of the Org with its environment takes place in a three dimensional world where pursuit-evasion games are conducted with other Orgs. The Org is an autonomous agent capable of learning its environmental constraints and self-organizing knowledge and experience. The long-term goal of this project is to create a generic artificial agent, used as an unsupervised problem solver on a diverse set of environments.

1 Introduction

The field of artificial life has many examples of autonomous agents that exhibit basic learning, behavior and self-organizing properties. Biology, developmental psychology and evolutionary theory [6, 12, 13, 14] have been used as a source and inspiration of methods and concepts already found in the natural world. Independent of the underlying theory used to create the artificial life form, the human experimenter should ensure, at a minimum, that the agent has the tools inherent in its design to learn and evolve unsupervised. For the organism to be successful, self-organization of its accumulation of knowledge, experience and perception must occur.

This work presents a simulator of autonomous entities (referred to *Orgs*, from the word *org*anism) that can learn behavior, observe their own actions and thus learn external constraints. The Org can self-organize and efficiently accumulate knowledge and use primitive, abstract behavioral patterns (instincts) as powerful goal achievement tools. The advantages of combining reactive behavior with long term planning become obvious when observing the evolving behavior of the Org.

Instincts are shown to be a valuable tool for guiding unsupervised learning. Orgs are shown to learn environmental constraints and physical laws by observing their own actions in response to the environment and other Orgs. The knowledge self-organization program in each agent successfully compresses information, which is subsequently used to generate decisions utilizing a fuzzy inference engine.

1.1 Virtual Environment

Orgs are placed in a three-dimensional bounded environment (*Orgland*). **Figure 2** provides a snapshot of the environment and the perspective of the same world by two Orgs (one is marked "Aggressive" the other "Evasive"). Instead of using two-dimensional opaque surfaces (the equivalent of walls), reflective surfaces were used. The use of these reflective surfaces is described in section 1.11. The boundaries appear as mirrors to an approaching Org - large flat surfaces reflecting the Org's image back to its sensors. No other objects are reflected. In this sense, the mirrors are special in that they possess a unique property relative to the Org.

1.2 Perception

The level and sophistication of perception is a critical component of an intelligent being. Perception is a compressed, transformed version of sensory stimulus - information efficiently translated for learning and memory. The underlying sensory and learning architecture is constantly influenced by *what* is sensed. What is perceived in the past changes the way we perceive things in the future [3, 4, 5, 9]. From the perspective of Orgs, Orgland consists solely of perception of objects or events created by the simulator's environment generator. It is the perception and cognitive processes specific to each entity that creates the world in which the Org lives. If an Org can't sense, speculate, theorize or guess about the existence of another entity,

0-7803-4869-9/98 $10.00©1998 IEEE

then that entity, from the perspective of the Org, does not exist. When the Org's perception or cognition expands, a previously non-existent event or object can become available. In this study, the learning mechanism remains unaffected by new knowledge. However the effect new stimuli have on the Org changes as the Org accumulates "knowledge".

1.3 Internal representation

Independent of form and method, internal representations of external information, both in time and space, are considered essential for developing unsupervised agents. Many different types of such representations, closely coupled with the learning mechanism, have been proposed. Orgs use a simple classifying architecture that serves the purpose of accumulating knowledge and subsequently using that knowledge for decision making and *speculation*. In addition, perception changes as the entity experiences. Knowledge and experience in this context is any of the following.

- Sensed objects and their properties (color, size, etc.)
- Observed state transitions of an object or alternatively observed object(s) behavior
- Observed spatial configuration of multiple objects
- Desired self-state transition and realized state transition differences
- Speculation and correlation of existing knowledge, to create new knowledge that has not yet been experienced.

1.4 Instincts

Instinctive behavior is common in the animal world. Nature has found a way of genetically coding basic behavior or reaction to environmental stimulus that are essential survival tools. In the evolutionary computation sense, the transformation of acquired knowledge during a lifetime can be passed to future generations in the form of genetic information. This is the known as the Baldwin Effect and an important basis for this work [1, 2, 11, 12, 16]. This theory claims that organisms capable of certain traits that enable them to learn faster will have a higher chance of propagating their genes. Then through several generations, this advanced learning ability will be coded in the genes allowing future organisms to acquire experience and learn a certain behavior a lot quicker.

The Org model utilizes abstract behavioral rules present in the simulated entities, at birth - prior to their exploration and interaction in the simulated environment. Those abstract rules in each Org are used as the equivalent of instincts in nature. By coding instincts in the Orgs, the designer initializes the agent with meaningful rules instead of relying on a random primordial state. By choosing the instincts, the experimenter can influence the agents to fulfill specific goals or solve certain problems without interfering after their activation. The abstract initial knowledge is then refined through learning. The agents are thus truly autonomous. In this implementation, the Orgs were configured with the following instinctual properties.

1. If an external object type is evasive, attempt a large decrease in separating distance. If no long term goal corresponding to another evasive object exists, make this object the current long term goal
2. If an object is aggressive, attempt a large increase in distance.
3. If an object is stationary, attempt a small increase in distance
4. If an object is unknown, attempt a small decrease in distance between it and self.
5. If no long-term goal exists and no evasive object is detected, select a random point in the environment and make those points the long-term destination/goal. This instinct drives the Org's exploration of the environment.

The choices above are intuitive and selected only for the pursuit-evasion application of the Orgs presented here.

1.5 Sensory interface

The simulated entities presented here can sense the following properties of objects in their virtual world.

- Distance
- Velocity
- Color
- Size

A more advanced sensation, also generated by the sensors, is the sense of "type". The following types are used.

1. If an object is getting closer to the Org, it is considered aggressive.
2. If an object is retreating, it is considered **evasive**.
3. If an object has been observed only in the current iteration so temporal differentiation can not be performed, it is labeled as **unknown**
4. If the object is in steady state since the last iteration, it is considered **static**.

The Org uses its experience, knowledge and instincts to produce a state transition output from its current sensory input. *The state transition output is a vector describing the change in value of each property of the Org.* The center of decision making, perception and learning is referred to as the *core*. A state transition is not limited just to position change, but could mean that the

Org changed its size or color in a chameleon fashion to evolve closer to a desired state. This state can be either an internally generated long term self-state or can be a distant position in the environment close to a desired object. In the latter case, the state is the aggregate of the sensed properties of the external object.

1.6 Simulated sensor characteristics

The sensors in this simulation have dynamic ranges dependent on the property (or attribute) they are designed to sense. A color sensor, for example, will have a much greater dynamic range than a texture sensor. In the case of the image sensor, the range is dictated by the actual size of the object being observed. As an object approaches, its apparent size increases.

Noise is added in each sensor reading, proportional to the distance from the sensed object. This introduces uncertainty in the sensory information and tests the Org's performance in a noisy environment. In similarity with the real world, all Org sensors have limited resolving power.

1.7 Static versus volatile properties

As the Org observes external objects over time, it can determine whether a property is volatile or not. In this simulation environment, color is one property that remains constant through time, enabling each Org to uniquely identify previously seen objects. Size, velocity and position are all volatile properties, changing throughout the simulation. The role of each property is described below.

Property	Volatility	Role
Color	Static	Unique for each object.
Size	Volatile	Determines sensory range. Perceived size changes with distance
Distance	Volatile	Collision detection, used for describing a reaction (section 1.4)
Velocity	Volatile	Determines the *type* property of an object (section 1.5).

Table 1. Sensor Properties

1.8 Long-term planning and reactive behavior

Insects and the artificial life examples that model them, exhibit reactive behavior [6, 13, 14]. Stimuli from the environment cause an immediate reaction by the organism. Higher cognitive functions are not involved in this type of sensory-motor coordination. Creatures capable of decisions based on experiences and knowledge however, can endure an unpleasant task in the short term in order to achieve a very desirable result in the future.

The simulated entities presented here exhibit both reactive and long-term behavior. During each simulated time step, a reactive output is calculated thereby applying the instinctive knowledge to the sensory input. In addition, the entity produces a long-term output by applying its current long-term goal to its experience, knowledge and instincts. The following cases describe how the final objective is selected.

1. *Desirable objects sensed with no current long-term goal present* – If an evasive object is sensed, this object becomes the long-term goal.
2. *No desirable objects sensed with no current long term goal* – If the sensory interface does not detect any desirable objects and there is no internal goal state (an internal goal is a self-state that the organism wants to achieve), then instincts will dictate what the long term goal should be.
3. *Desirable object sensed, different than current long-term goal* – If the current long-term goal was an instinctive desire generated in the past, due to lack of an external desirable stimuli or future self-state, then the currently sensed desired object will become the new goal replacing the current long term goal.

When a long-term goal exists, the *core* is used to generate a state transition output (section 2.5), using the long-term goal and the current sensory stimuli. Both reactive and long-term outputs are then combined through a weighted average. The weights are dynamically generated, with a bias towards the long-term output. The reactive output is the average of the *Core Output* for all sensed objects. The following formula describes the transition vector calculation.

$$\Delta_{t+1} = w_{lt}\,\Delta_{\text{long term}} + w_r\,\Delta_{\text{reactive}}$$

where w_{lt} and w_r are the weights associated with the long-term transition vector $\Delta_{\text{long term}}$, and the reactive transition vector Δ_{reactive}. Instincts or experience can modify the weights according to the current sensory stimulus or internal state

1.9 Learning external object behavior

Creatures with advanced cognitive functions can correlate several sequential actions that occurred in the past and group them together as behavioral patterns. The term ***iteration*** is used throughout this section to describe the smallest time interval in the simulation. Time is

discrete so all observations made by an Org are made in one iteration.

A behavioral pattern is a segment, of a sequence of state transitions. That is, a behavioral pattern $\mathbf{f_t}$ is a mapping $\mathbf{f_t} : T \rightarrow A_t$ where $T = \{ 0,1, \ldots, n\}$ and $\mathbf{A_t}$ is a set of state transitions E_{t-n}, E_{t-n+1}, ..., E_t of the same observed object and $f(k) = E_{t-n+k}$ for $k = 0, 1, \ldots, n$. The Org observing the state transitions of an external object must somehow determine if the transitions follow a pattern and, if so, decide when the pattern is sufficiently described.

To fully capture behavior, an object must be sensed during its entire set of state transitions that constitute a behavior set. If the object moves far away at step *t* and can no longer be sensed, then whatever set was being compiled, will be the same set A_{t+i} used in all *i* subsequent Org iterations until the external object is perceived again. Even after an object is sensed again at step *k*, new state transitions will not necessarily be added to the set of observed state transitions. The object has to perform a sequence of *N* state transitions, were $\mathbf{N = max(2, |A_k|/2)}$, where $|A_k|$ is the cardinality of A_k, matching a previously seen sub-sequence in set A_k. The previous formulation imposes the limitation on the size of A_k (greater or equal to two) before it can be considered a behavior set. If the state transition at step $k+N$ is not previously seen, then the state E_{k+N} is added to the behavior set A_{k+N} . A behavior set A_i is considered complete when the external object reaches a steady state (all *N* state transitions $E_{i-N}E_{i-N-1}\ldots$ before step *i* are the same).

When an object with a previous known behavior is sensed, its state transitions - treated as events - will be compared to previously known behavior. If these events match a sub-sequence of a behavior, the Org will react to the learned behavior and not the immediate state transition. As a result, the action taken by the Org is in anticipation of a future state.

One of the limitations of the above method is that complex behavior patterns can not be detected. As an example, consider the case where an external object shows aggressive behavior after it has been approached. Then it stops moving, reaching what the Org considers a steady state and as a result a behavior set is completed. When the Org leaves the vicinity of the previously aggressive object, the object start chasing the Org again and collides with it. Since this new behavior was sensed after the Org had classified the previous behavior, its is now considered a new behavior set. However, the external object result in this collision only because of the initial approach by the Org. So this is just the continuation of a complex behavior that started many iterations before.

1.10 Learning constraints using self-reflection

The Orgs are placed into a three dimensional world (Orgland). Before a temporal iteration is completed, each Org, based on its long-term goal(s), perceived events, observed objects and any internally, generates experimental rules in the form of a state transition vector $OrgState_i$. This transition vector is then passed through a *constraint filter*. This filter models the laws of Orgland; physical and artificial constraints that should be applied whenever an Org decides to change its state. The output of this filter is a modified state transition vector that reflects the effect of the environment. **Figure 1** describes the role of this filter in forming new knowledge. The next state of an Org is calculated by summing the filter output with the current state of the Org, using the formula:

$$\text{OrgState}_{t+1} = H_{\text{out}} / 2 + \text{OrgState}_t,$$

where H_{out} is a vector composed of the changed Org property values after filtering. As an example, if an Org collides with an object, the Org will experience an inelastic collision and its velocity will be affected. The Orgs learn the constraints of their virtual world as they experience them. By comparing their new (filtered) state transition to the desired (filter input) state transition, the Org can learn what effect, if any, the environment had on the state transition vector. The filter input and output is then used to form a new knowledge element. Then when the Org encounters a boundary or another object it can make decision affected by the collision it experienced in the past

This idea of comparing intended actions with what actually occurred allows the Org to learn the physical laws of its environment by observing their effect on its own state. By doing so, the Org can eventually accumulate enough knowledge to create an internal "model" of a physical law.

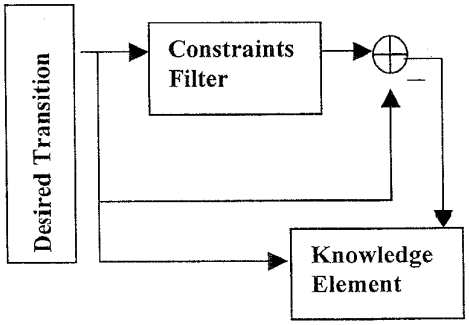

Figure 1. Learning environmental constraints.

1.11 Mirrors

Reflective boundaries are used in Orgland. They act as virtual boundaries and, at the same time, as an effective method for Orgs to learn their external image. The Org will observe an object that looks and behaves identical to it, but it is only two-dimensional. Since the Orgs have the luxury of knowing their current state, by observing the mirrored image, they can learn they exist and occupy space. This might seem trivial to humans, but being aware that what you see in the mirror is actually a reflection of your own body, is a difficult cognitive step. This work does not claim that this step has been made - it just presents a simple way that could make such a feat easier.

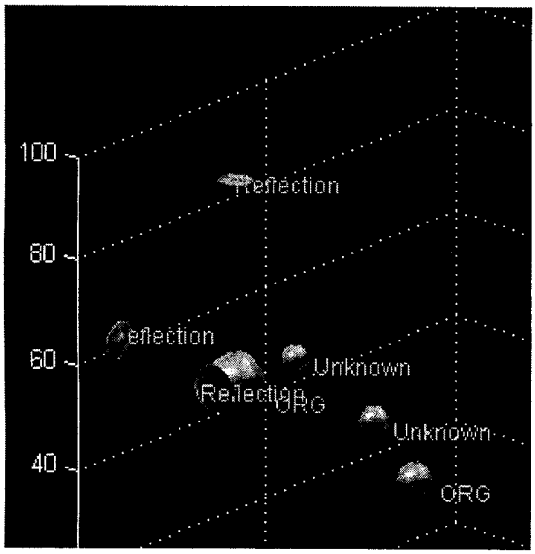

Figure 2. The flat objects on each virtual wall, are Org reflections. The reflection size increases as the Org gets closer to the mirrored boundaries.

2 Software Algorithm and Data structures

The theoretical concepts presented here were developed into an object oriented software algorithm and data representation. Since the algorithm and data structures are an essential part of this theory, the central learning and decision-making mechanisms (referred to as the *Core*) are presented through their software implementation. All software was written in MATLAB 5.

2.1 Learning

The Orgs can learn from their perceptions and their environmental constraints. Before knowledge can be presented to the *Core*, it has to be represented properly. The container used to carry information is a software construct called *KnowledgeElement*. This container has an *Effect* component and a *Condition* component. The *Effect* describes the state transition that the *Condition* will cause to the Org if it was the only input presented to the *Core*.

The *Condition* could be any of the items mentioned in section 2.1 (*e.g.* a sensed object or a collision with a boundary). In addition, the current reactive output is stored - linked with the item used as the condition. The justification for this is that the same stimulus will have a different effect under a different circumstance and internal state. If, for example, the Org is being chased, the *Core* produces a large negative reactive output. This means that the Org is sensing something instinctively undesirable. If, at this situation, the Org senses a desirable object, its course of action will be different than if it was under no threat.

The *Effect* component of a *KnowledgeElement* can contain active scalar and vector properties. If a property is active in the *Effect*, a change in the same property in the Org's current state results.

Both the *Effect* and *Condition* components of the *KnowledgeElement* have a property vector that describes which features are present in the information they are carrying. The following table describes which properties are active for each class of perceived information.

Perception Type	Active Features
External Object	All properties within their respective sensor range (Section 1.3)
Behavior	All sensed properties of the initial state
Environmental Constraints	Properties modified in Transition Vector

Table 2. **Perceived Information Properties**

After a *KnowledgeElement* is prepared, it is presented to a classifying algorithm that groups elements with the identical property vectors together. Each group is a construct called *KnowledgeBin* and can have up to a predefined number of elements with no duplicates allowed. A group of *KnowledgeElements* is not a cluster since the membership criterion has nothing to do with the values of its members. The criterion is the number and type of properties of each member.

After a *saturation threshold* is reached, the classifying algorithm will merge the new member with the most similar existing member of the *KnowledgeBin*. The Euclidean distance between the vectors formed by concatenating the active properties of *KnowledgeElements* is used to establish similarity. The above method limits the knowledge stored in the Core and concurrently *self-organizes* already present knowledge.

Old knowledge that is no longer used will be gradually replaced with new information thereby eliminating the need for time decay in the knowledge of each simulated entity.

2.2 Decision making

For each *KnowledgeElement* with a valid *Condition* component and no *Effect* component, the *Core* is required to generate a reaction vector. First, all existing *KnowledgeClusters* are searched for matching property vectors. If the property vector does not match (the number and type of active properties is not the same), a *KnowledgeBin* that has a subset of the active properties of the input will be used. For each element of a compatible bin, a decision output (similar to a rule) is generated. Using the *Condition* component of each cluster element, the concatenated values of all active properties generate the condition portion of the condition/decision vector.

For vector properties, there is an additional value associated with the vector. This value is an abstraction of the vector and represents a fuzzy set. Since vector properties (such as velocity) are only used in the *Effect* component, it is sufficient for this implementation to represent them with a single fuzzy set. Thus, for representing the effect of large increase in velocity, the scalar representing the velocity property would be the fuzzy set LARGE_POSITIVE.

The consequent vector, *Y*, in the fuzzy inference engine is generated in a similar fashion using the *Effect* component. Concatenating vectors *X* and *Y* forms rule *R*. For each knowledge cluster a rule base is formed from the rules that were generated from the cluster members. This ensures that each rule in the rule base has equally sized vectors.

After a rule base is created, a simple list of fuzzy if-then rules is then used to produce an output vector [13]. The fuzzy system uses Gaussian membership functions with supports being the antecedent *X* or consequent *Y* of each rule R_i. This means that each rule has its own membership functions thereby eliminating the need of heuristically dividing the input and output domains for each property. The system uses a center-average defuzzifier with sum-product inferencing.

After a fuzzy result has been generated it is encoded back to the *Effect* component of the input *KnowledgeElement*. This is now treated as the reaction of the *Core*, to the *Condition* component of the input element. The following pseudo-code describes the algorithm for decision making, using a sensed object as the input:

- Find all bins with a membership criterion equal to or a subset of the *Condition* properties in this object.
- *For each bin*
  - ✓ Generate a rule base with a rule describing each member of the bin.
  - ✓ Create an unknown input vector from the subset of *Condition* properties supported by the bin used to generate the rule base. Present the input vector to fuzzy system utilizing current rule base and store fuzzy output
- Average the fuzzy outputs from each bin and use them to generate an *Effect* component, for current object
- Create new *KnowledgeElement* using the sensed object as the *Condition* component and the fuzzy result as the *Effect component*.
- Use fuzzy results to create a reaction vector towards the sensed object.

3 Simulation results

The following section describes one simulation run, with two Orgs placed in the same virtual environment, interacting with each other and their surroundings. The author does not claim that the following simulations prove that the theoretical concepts presented here are correct. This work is in its initial state and more focus has been given in formulating and presenting the theory behind the artificial agents, than creating a proper experimental setup for testing the performance and correctness of its concepts.

An example of the paths taken by two Orgs is shown in **Figure 3**. An interesting result of the instincts used here (section 1.4) is that if an object type change occurs during two consecutive iterations, on an Org being chased, the current long-term goal of the aggressive ORG is affected. So if an object exhibits evasive behavior for *N* iterations, and is being chased by another ORG, and then at the *N+1* iteration the Org turns aggressive, the ORG in pursuit will stop going after the object and start avoiding it. This is illustrated in **Figure 3**. Shown is the result of one of the ORGs managing to capture the ORG coming from the upper corner of the plot. At iteration ***18***, both ORGs are engaged in pursuit – evasion behavior, with the lower ORG being the aggressor. At iteration ***36*** they came into contact, since the upper ORG was trapped in a corner and had to trace the boundary of the environment. At iteration ***42*** however, the victim turned into the aggressor, forcing the Org originating from the bottom part of the plot, to start heading back to its origin. At iteration ***48***, the upper Org in pursuit of the lower Org can be seen. Their roles are reversed. The graphic objects representing the Orgs (at iteration ***48***) were intentionally omitted, to make the paths more visible.

Observing the knowledge organization of the Orgs, after an iteration run of approximately 50 steps, it was found that instinctive knowledge was unaltered and occupied its original bins. Most of the observed objects were occupying the same cluster since they all have the same number of active properties. The bin members representing moving objects, had accumulated multiple state transitions for that object. Recognizing behavioral patterns in external objects is a difficult task and has not been verified with the current implementation. In the simulations presented here none of the two Orgs exhibited a fixed behavior pattern.

The knowledge classification scheme and the use of a saturation threshold works well and efficiently represent large amounts of knowledge. For simulations with two interacting Orgs and 10 static external objects, events or objects observed in the past were merged with existing knowledge keeping the number of bins and bin elements to a maximum of five, The bins found after the simulation presented in **Figure 3**, were as follows:

- Two bins contained instincts,
- One bin contained the all the sensory stimulus generated by static objects and reflections. The objects were used as the *Condition* component, with the reaction they induced to the Org being the *Effect* portion of each *KnowledgeElement*
- One bin contained only one element with knowledge acquired when one of the ORGs collided with a boundary.
- The last bin also had one element, describing the other Org in the environment. Although another Org looks like any other object, it exhibits aggressive or evasive behavior. The single element of this cluster contained numerous states, describing a behavior.

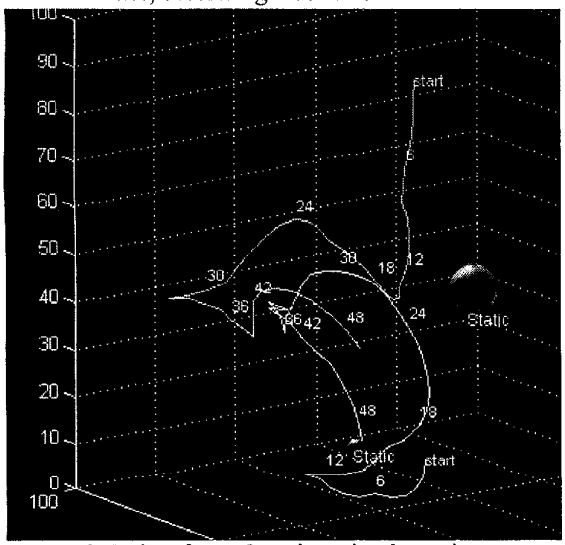

Figure 3. Paths of two Orgs in a simple environment.

The reflections were merged into one element for each boundary, demonstrating the ability of the self-organizing knowledge structure to compress information. It did not seem, however, the Org had made the association that the reflections were external objects generated by its own image falling on the boundaries. This indicates that in order to for a simulated agent to learn something, it has to have some information *on what to look for.* The authors believe that such information can come from instincts, communication with other objects in the environment, or *speculation.*

An animated movie of two simulations (including the one above), are available at http://cialab.ee.washington.edu/Marks-Stuff/presentations.htm

4 Research Issues and Future Work

There are several key research issues this work attempts to solve. Several others need to be addressed the future. The following are common in the Artificial Life and Computational Intelligence Field and an attempt to address them was made by this thesis:

1. *Internal representation and classification of knowledge* – This work presents a clustering/self-organization algorithm that classifies varying input size vectors. Separate groupings are created for an input vector of a given size. Thus this method has the flexibility of representing variable sized rules and then combining multiple rule-bases for generating a common decision vector (section 2).
2. *Selecting the optimal initial rules/embedded knowledge* – The use of abstract initial rules (instincts) that are constantly refined throughout the use of an Org is the approach chosen here for introducing heuristics into an autonomous agent. The fact that such heuristics can even be utilized is very useful. Choosing the optimal set of instincts for a give environment and desired behavior is important. However it is not critical for the success of an Org. Even so, more work needs to be done to discover methods of automatically generating instincts for a given Org application.
3. *Invariance to changing background while observing events and objects* – The ability to discern a previously seen object or behavior, even if the surroundings have changed, is one of the goals of this work. The learning mechanism presented here classifies objects independent of their surroundings but this mostly due to the design of the sensors and virtual world used in this simulation. So the Org's sensor ability to recognize discrete objects is most

similar to that of radar and very unlike real-time image capture and processing. Demonstrating that the Org's knowledge classification is invariant to a changing background is planned for future research.

4. *Escaping local-minima* - The use of both a reactive (short-term) and long-term decision components serves the purpose of avoiding situations were the algorithm reaches a deadlock trying to achieve a goal. Further proof that this is indeed the case with the Org algorithm needs to be demonstrated.

Many implementation alternatives of Orgs in Orgland are in need of exploration. The following extensions are currently planned or in progress.

- Communication between Orgs will be implemented so the concept of *tutors* can be introduced. Tutors are Orgs who serve the purpose of relaying useful information to another agent without human intervention.
- More properties will be added to the simulated agents so they have a more complete set of sensory and motor functions.
- A speculation facility will be added. Here, the ORG takes combinations of existing *KnowledgeElements* and creates new untested knowledge.
- The use of concurrent neural networks will be explored as an alternative to the current knowledge classification schemes.

5 References

1. Anderson, R.W., 1995, "Genetic mechanisms underlying the Baldwin effect are evident in natural antibodies", in Evolutionary Programming IV: The Edited Proceedings of the Fourth Annual Conference on Evolutionary Programming, edited by J.R. McDonnell, R.G. Reynolds, and D.B. Fogel, pp. 547-563. Cambridge, MA: MIT Press.
2. Baldwin, J.M. 1896, "A new factor in evolution", American Naturalist, 30, 441-451.
3. Barlow, H. B., 1972, "Single Units And Sensation: A Neuron Doctrine for Perceptual Psychology?", Perception, Vol. 1, pp. 371-394.
4. Barlow, H., B., 1990, "Conditions for Versatile Learning, Helmholtz's Unconscious Inference, and the Task Of Perception", Vision research, Vol. 30, No. 11, pp. 1561-1571
5. Braitenberg, V., 1989, "Reading The Structure of Brains", Network, Vol. 1, pp. 1-11
6. Chialvo, D. R., Millonas, M. M, "How Swarms Build Cognitive Maps", The Santa Fe Institute.
7. Di Paolo, E. A., "An investigation into the evolution of communicative behaviors", Cognitive and Computing Sciences, University of Sussex
8. Dorigo, M. *"Genetics-Based Machine Learning and Behavior-Based Robotics: A New Synthesis."*. IEEE Transactions on System, Man, and Cybernetics, Vol. 23, No. 1, January /February 1993.
9. Gutfreund, H., Toulouse, G., 1994 'Biology and Computation: A Physicists Choice", Advanced Series in Neuroscience, Vol. 3
10. Hebb, D. O., 1949, "The Organization of Behavior", New York: Wiley.
11. Hightower, R., Forrest, S., Perelson, A., (1995) The Baldwin effect in the immune system: learning by somatic hypermutation. In L.J. Eshelman (ed.) Proceedings of the 5th Intl. Conference on Genetic Algorithms, Morgan Kaufmann, San Francisco, CA. pp. 184-192 (1995).
12. Hinton, G.E, Nowlan, S.J., 1987, "How learning can guide evolution", Complex Systems, 1, 495-502.
13. Lambrinos, D., Scheier, C. (1996) "Building Complete Autonomous Agents: A Case Study on Categorization." Submitted to: IROS'96, IEEE/RSJ International Conference on Intelligent Robots and Systems, November 4-8, 1996, Senri Life Science Center, Osaka, Japan.
14. Mataric, M. J. (1994), "Interaction and Intelligent Behavior", MIT EECS PhD Thesis, May 1994, MIT AI Lab Tech Report AITR-1495.
15. Rutkowska, J. C., "Can Development Be Designed? What we May Learn From the Cog Project", Cognitive and Computing Sciences, University of Sussex
16. Turney, P., Whitley, D., Anderson, R. (eds), "Evolution, Learning, and Instinct: 100 Years of the Baldwin Effect", a Special Issue of Evolutionary Computation
17. VanLandingham, H., Chrysanthakopoulos, G. *"Data Driven Fuzzy Systems for System Modelling"*, IEEE Systems, Man, and Cybernetics Conference, October 1995, Vancouver, B.C