LEARNING IN AGENT-BASED MODELING OF ARTIFICIAL SOCIETIES

A Project in
Computer Science
by
Hans B. Royer

©1999 Hans B. Royer

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

July 1999
I grant The Pennsylvania State University the nonexclusive right to use this work for the University's own purpose and to make single copies of the work available to the Public on a not-for-process basis if copies are not otherwise available.

____________________________________

Hans B.Royer
SIGNATORY PAGE

We approve the project of Hans B. Royer

________________________________________
Thang N. Bui
Associate Professor of Computer Science
Project Advisor

________________________________________
Linda M. Null
Assistant Professor of Computer Science

________________________________________
Timothy A. Wahls
Assistant Professor of Computer Science
ABSTRACT

The study of artificial life, generally thought to have been fathered by John von Newmann, was initially of interest primarily to computer scientists and mathematicians. In the early Seventies, Thomas Schelling applied the concepts of artificial life to the social sciences in his study of urban segregation. At first the use of artificial life or agent-based computer modeling was restricted by the limited computing power of the day. As the computing power dramatically increased and programming tools became easier to use, more and more the concepts of artificial life were used by economists, biologists, sociologists, and others to study problems in their fields.

This paper describes an extension of the artificial society model Sugarscape. It attempts to explore the effects of adding the capability for agents to choose from among different behaviors instead of being locked into one behavior or controlled by a global rule set over the course of a simulation. This choice of behaviors is a weighted selection, where the weight a behavior has depends on previous experience. It will do this by comparing various attributes of the population, such as average population size, average wealth of the population, etc. from runs of the simulation without learning to runs of the simulation with learning.
# Table of Contents

Abstract.......................................................................................................................... iii  
Table of Contents ........................................................................................................ iv  
1  Introduction................................................................................................................ 1  
2  Overview of Sugarscape ......................................................................................... 3  
3  Overview of Erewhon............................................................................................... 5  
   3.1  Implementation ................................................................................................. 5  
   3.2  Phase One ........................................................................................................ 6  
   3.3  Phase Two ......................................................................................................... 10  
   3.4  Phase Three ...................................................................................................... 11  
   3.5  Phase Four ....................................................................................................... 12  
   3.6  Phase Five ....................................................................................................... 14  
4  Data Collection and Observations........................................................................... 16  
   4.1  Phase One ......................................................................................................... 16  
   4.2  Phase Two ......................................................................................................... 17  
   4.3  Phase Three ...................................................................................................... 19  
   4.4  Phase Four ....................................................................................................... 20  
   4.5  Phase Five ....................................................................................................... 23  
5  Conclusion................................................................................................................ 26  
References ..................................................................................................................... 27  
Appendix A  Glossary.................................................................................................. 28  
Appendix B  Future Enhancements .......................................................................... 29
1 INTRODUCTION

Artificial Life is the discipline that investigates the synthesis of life-like behaviors in computers, machines, etc. to study topics such as the origin of life, social organization, and evolutionary and ecological dynamics. Why would we want to synthesize life when real examples of life are available for study? Having only examples native to Earth presents problems to researchers. They are unable to distinguish characteristics that are universal to life everywhere and are unique to Earth. Having other examples of life would allow researchers to derive general theories of life. Just as synthetic chemistry (creating chemical compounds not found in nature) increased the understanding of chemical phenomena, hopefully, the synthesis of life will increase the understanding of the phenomena of life [7][11].

According to the introduction at the ISU Complex Adaptive Systems Workshop web site, many natural systems and increasingly many artificial systems as well are characterized by apparently complex behaviors that arise as the result of nonlinear interactions among a large number of components or subsystems. Examples of such natural systems include immune systems, nervous systems, multi-cellular organisms, ecologies, and insect societies. Artificial systems sharing this property include parallel and distributed computing systems, large-scale communication networks, artificial neural networks, evolutionary algorithms, large-scale software systems, and economies. Such systems have recently come to be known as Complex Adaptive Systems (CAS) [13].

According to the introduction at the Web Site for Agent-Based Computational Economics, Agent-based Computational Economics (ACE) is the computational study of economics modeled as the evolving decentralized systems of autonomous interacting agents. ACE researchers attempt to understand the apparently spontaneous appearance of global regularities in economics processes. ACE researchers attempt to explain how these global regularities arise from the bottom up by studying the repeated local interactions of autonomous agents channeled through socioeconomic institutions. ACE is thus a specialization to economics of the basic CAS paradigm [14].

First presented in Growing Artificial Societies, by Joshua M. Epstein and Robert Axtell, Sugarscape is the application of agent-based computer modeling to the study of social and economic phenomena. This modeling methodology had its start with John von Neumann’s work on self-reproducing automata. In the early 1971, Thomas Schelling was the first to apply the work to the social sciences in his studies of urban segregation. Currently, there are many examples in a variety of fields of the use of agent-based modeling to study various problems [4].

Sugarscape is also an example of an artificial society. Agent-based models of social and economic processes are also known as artificial societies. Artificial societies involve three basic ingredients: a population of agents, an environment, and a set of rules [4].

Another example of an agent-based model is Echo. Echo is a generic system model of an ecosystem. In Echo, evolving agents are placed in an environment with limited resources. The behavior of these agents and the environment are used to study the effects of biological evolution. This model demonstrates good qualitative agreement with studies of naturally occurring species [6].
A third example is the work being done by Eric Werk of the University of Aarhus in which emotion is added to the agents populating an artificial society that is modeled on Sugarscape. In Werk’s model, the autonomous agent, known as a peasant, has its behavior altered by a happiness function. There is no evolving or learning of the agents [15].

The subject of this paper, Erehwon, is an extension of the Sugarscape model. In Erehwon as in Sugarscape, agents populate an environment of varying levels of resources. The agents interact with the environment and neighboring agents.

Initially, Erehwon is a re-implementation of the Sugarscape model to verify that the behavior is similar to Sugarscape. After that, agents are given the ability to choose from among different behaviors instead of one behavior (a simple form of learning). The agents in Erehwon gather information about their environment and neighboring agents, then adjust their behavior using a simple learning model. The behavior of the Erehwon model with agent learning is compared to the model without agent learning to determine what, if any, effect agent learning has on the model. Erehwon is implemented in five phases as described below. Note: Phases One through Four are the re-implementation of Sugarscape.

- **Phase One** – Agents gather and consume one resource. Population size is maintained constant by replacing any dead agents with new randomly created ones.
- **Phase Two** – Agents gather and consume one resource. Population size is variable. New agents are created by mating.
- **Phase Three** – Agents gather and consume two resources. Population size is variable. New agents are created by mating.
- **Phase Four** – Agents gather, consume, and trade two resources. Population size is variable. New agents are created by mating.
- **Phase Five** – Agents gather, consume, and trade two resources. Agents also can alter their own behavior over the course of the simulation through a simple learning process. Population size is variable. New agents are created by mating.

Erehwon, being an extension of the Sugarscape model, is another example of an artificial society. The primary goal of the project is to examine the effects on the model of adding learning to agents. A secondary goal is to create software that can be easily extended to study other aspects of artificial societies and artificial life.
2 Overview of Sugarscape

A brief introduction to Sugarscape is presented. Those wishing greater detail should consult Growing Artificial Societies [4]. Only those parts that are re-implemented in Erehwon are discussed.

In the initial phase of Sugarscape, a population of agents is born into the sugarscape with vision and metabolism. Vision and metabolism represent physical attributes of the agents. Having higher vision means that the agent can see farther and is more likely to find the sugar (the resource) it needs to survive. Having a lower metabolism means that the agent burns less food when it moves. By conserving more of its sugar, the agent is more likely to survive. The sugarscape (or landscape) is a two dimensional array of areas. Some areas possess varying amounts of sugar. An agent looks around for sugar, moves to areas of higher sugar, and eats the sugar. When it moves, it burns an amount of sugar equal to its metabolism. If the agent burns up all of its sugar, it dies.

During the first phase, Axtell and Epstein observed that the behavior of the Sugarscape model closely followed that of natural populations of grazers. The overall fitness of the population improved as the least fit members of the population died of starvation. They also observed grazing behavior similar to natural populations, when they used a resource grow back rate that was less than the capacity of the site. This grazing is an example of a population behavior that is generated “from the bottom-up” (created from the nonlinear interactions among the agents). Attempting to recreate this behavior globally by giving each agent explicit instructions as to which area to graze at what time would result in a very complicated harvesting program, particularly, if the different qualities of the agents were taken into account.

The second phase of Sugarscape adds mating to the model. Now agents that die are not automatically replaced. New agents are created only when two agents mate. Agents mate only when they are fertile and are mature. The population size of agents can expand to greater than the initial population size.

In the second phase, the behavior of the Sugarscape model demonstrated an evolutionary process that closely paralleled that of natural evolution. Agents with high vision and low metabolism were more likely to survive and reproduce. Axtell and Epstein also commented on the speed of the evolutionary process by noting that the average values for metabolism and vision improved at a faster rate with mating than without mating. They also noted that average metabolism dropped faster than average vision rose.

The third phase of Sugarscape adds a second resource, spice, to the landscape to prepare for trading. Now, agents use a welfare function to determine to which area they will move. Agents now have a greater chance of death since they must maintain inventories of two resources. Consequently, the carrying capacity (the number of agents that an environment can sustain) is less than that of the previous phase. Other than the lower carrying capacity, the behavior of this phase of Sugarscape is similar to the second phase of Sugarscape.

The fourth phase of Sugarscape adds trading to the model. Agents can now exchange resources with other agents in a primitive form of barter. Trade increases the carrying capacity of the environment. This is the behavior of the model Sugarscape in this phase.
that is of primary interest in this project as it will be used to determine the effect of adding learning to the agents. Axtell and Epstein also used this model to study the behavior of de-centralized markets.

This was a brief overview of the Sugarscape model leaving out many areas that will not be explored in Erehwon. Areas such as culture, conflict, disease, etc. may be explored in future versions of Erehwon. Also, Axtell and Epstein did not explicitly call the divisions of their model phases. That was done for this project to allow easier comparison of Sugarscape and Erehwon.
3 OVERVIEW OF EREWHON

3.1 Implementation

The simulation, Erehwon\(^1\), is implemented in five phases using the Java 1.2 JDK and the Java Foundation Classes. One of the requirements decided upon was that the project should be portable between Windows NT and Sun Solaris. After examining several options (e.g., Java, C++ and Tcl/Tk), Java was chosen.

Initially, Erehwon was implemented using the Java 1.1 JDK and the AWT. Other than difficulties encountered with learning a new language, problems with the paint method were encountered. The problem was in repainting the window if the window was resized or minimized then restored. About the same time the second beta for the Java 1.2 JDK was available. The project was recompiled to bytecode. The same difficulties with the paint method were encountered. The AWT heavyweight components were removed and the JFC equivalent lightweight components were substituted. Although not as severe, difficulties with the paint method were still encountered. The problems were encountered on both Windows NT and Sun Solaris.

Since the release of the release version of the JDK v1.2 was being delayed, some of the earlier options were reexamined. One option, which initially seemed attractive, was the use of Tcl/Tk to provide the graphic interface and C++ to provide the simulation. Although a working version was of the Phase One was quickly implemented, it did not have some of the functionality desired. To implement that functionality would have required considerable time and effort. Plus, the interface did not present as attractive an appearance as the Java JFC.

About this time, the release version of the JDK had been released by Sun. The version that utilized the JFC was recompiled to bytecode. None of the problems encountered with the previous versions were present in the current version. However, Sun's implementation of the Java Virtual Machine on Windows NT appears to have problems with garbage collection. During runs of extended length (150 or more time increments), the program would crash due to an access violation. There were some minor problems on Suns that were not running Solaris 7. One example, dialog boxes to open or save files would not allow a user to enter a file name in the file name field by typing. No problems were encountered with Solaris 7. The decision was made to continue the project using the Java 1.2 JDK.

The implementation closely followed the description of the Sugarscape model given in Growing Artificial Societies. Any differences are noted in the individual phases. Until recently, the source code for Sugarscape was not available. Therefore, it could not be a simple matter of translating the Object Pascal and C code for the Macintosh to Java. According to Robert Axelrod \cite{axelrod2}, the source code is now available from Robert Axtell.

Note: Well into the project, Swarm, a software package for multi-agent simulation of complex systems being developed at the Santa Fe Institute, was discovered. Swarm is intended to be a useful tool for researchers in a variety of disciplines, especially artificial life. The basic architecture of Swarm is the simulation of collections of concurrently

\(^1\) The source code for Erehwon can be found at http://cs.hbg.psu.edu/~hxr100/erehown.html
interacting agents. Using this architecture, a large variety of agent based models can be implemented. Since it uses Objective C as its principle programming language, it was not immediately useful [8].

3.2 Phase One
In Phase One, the intelligent agents, also referred to as Dens, roam a randomly generated or user defined environment, also referred to as the Land, searching for and gathering resources (also referred to as Honey) in order to survive. The population of Dens (also referred to as the Populus) is maintained at a constant size by replacement of Dens that die. The Land is a two dimensional matrix of individual cells (also referred to as Acres). The overall simulation is known as Erewhon.

Erewhon
The overall simulation Erewhon is implemented as a class. Erewhon is responsible for running the simulation by keeping track of the current season. Each season, Erewhon will pass messages to the environment object, Land, and the population object, Populus, telling them to carry out their responsibilities. Erewhon also implements the user interface, which allows the user to set various parameters for the simulation and control the simulation. It also provides an animated view of the simulation as it progresses.

![Object Model of Simulation](image)
Overview of Erehwon

User Interface
Via buttons and pull-down menus, the user can select various options and change various parameters. In Phase One and subsequent Phases, the items of interest are:

- Choosing between a randomly created environment and a pre-defined environment.
- Creating a data file for a permanent record of the simulation.
- Saving a randomly created land.
- Changing the following initial parameters:
  - Number of seasons (iterations) the simulation is to run.
  - Initial size of the Populus (population).
  - The percentage of Acres that contain Honey (resource).

Also, there are displays for following information:

- A legend for amount of Honey an Acre contains.
- Initial average metabolism and vision of the Populus.
- Current average metabolism and vision of the Populus.
- Current size of the Populus.
- Current season.
- Two status displays.
  - Simulation Status (e.g., running, paused, etc.)
  - File Status (e.g., error loading file, etc.)

Vision and metabolism are attributes of the Dens (agents) comprising the Populus. Vision of a Den is how many Acres the Den can see along the X-Y axis. Metabolism of a Den is how many units of Honey the Den burns when it moves.

Running the Simulation
There are four buttons for controlling the simulations. The four buttons are

- Create – creates the environment (if a random environment is selected) and the initial Populus.
- Start – starts the simulation.
- Pause – pauses the simulation.
- Reset – resets the simulation.

Sequence of a Season (turn)
Each season has the following sequence. This starts as soon as the Start button is depressed.

- Dens (agent) move or gather Honey (resource).
- Dens are checked for starvation.
- Dead Dens are replaced by new Dens to keep the size of the Populus (population) constant.
- Unoccupied Acres regenerate Honey at a predetermined rate.
- The order of the Dens in the Populus is shuffled.

Acre
Acre is the class representing a site in the environment. In Phase One, the class Acre has data members for the amount of Honey present, the maximum amount of Honey that the site can support, the rate at which Honey is regenerated, the X and Y coordinates of the site, and a pointer to the occupant if present. Note: At most one Den may occupy an Acre at any instance in time.
The class also has the following methods to regenerate the Honey resource and methods to access most of its data members.

**Regenerating Honey**
In each season, every unoccupied Acre will regenerate the Honey available. The rate at which the Honey can regenerate can be selected prior to the start of the simulation. The default rate is one unit of Honey per season. Currently, there is no user option to change this.

This is a departure from the Sugarscape model. In Sugarscape, the grow back or regeneration of resources was instantaneous.

**Land**
Land is the class representing the environment. In Phase One, a Land object has data members consisting of a two dimensional array of Acre objects, and the dimension of the array. The two dimensional array is square and is implemented as a torus. That is agents are allowed to move from right/left to left/right and top/bottom to bottom/top.

The object also has methods to access an individual Acre within the matrix, to access the dimension of the matrix, and a method to send a message to all the Acres to regenerate their Honey resources.

**Den**
Den is the object for the intelligent agent. In Phase One, a Den object has data members for identification number, vision, metabolism, the amount of Honey carried, the capacity to carry Honey, pointer to the next Den object in the list, pointer to the Acre object that it is occupying, and a pointer to the Land object. There are also data members that are not utilized in Phase One. These are age, the initial Honey the Den possessed when created, pointers to the both parents (Den objects), pointer to a potential mate (Den object) and a flag indicating whether the Den has mated in the current season.

The object also has methods for moving, performing required actions (in Phase One, it is simply gathering Honey), dying, checking the validity of a move, and methods to access most of its data members. There are also methods that are not utilized in Phase One. These are methods that deal with mating and methods to access the data members that are not utilized in Phase One.

In Phase One, the Den acts according to the following rule: If on an Acre with Honey, gather Honey according to the rule for gathering, else move according to the rules for moving.

**Gathering Honey (Action)**
If the Den occupies an Acre with available Honey, it will gather as much available Honey from the site as it can carry. Since the capacity of the Dens to carry Honey is, for all practical purposes, unlimited, the Den will gather all the Honey on an Acre.

This is a slight departure from the Sugarscape model. In Sugarscape model, the carrying capacity of the agents is unlimited and cannot be changed. In Erehwon, the user may change the default maximum carrying capacity of the Dens prior to the start of the simulation.
Moving
If the Den is not occupying an Acre with Honey, it will move according to the following rules.

- Consume Honey equal to its metabolism.
- Move to the unoccupied Acre, within the range of vision, with the largest amount of Honey.
- If no available Acre with Honey is found, make a random move as far as the Den can see, in one of the four directions.
- If no unoccupied Acre is available make no move.

Honey is consumed by decrementing the amount of Honey carried. This amount is allowed to go to 0 or less, which means the Den will die of starvation.

Moving is accomplished by examining the Acres along the X-Y axis within the range of the Den’s vision. If an Acre is unoccupied, the amount of Honey is compared to the current maximum Acre’s amount of Honey, which is initialized to 0 at the start of every Den’s move. If the amount of Honey at the current Acre is greater than the current maximum Acre, it becomes the current maximum Acre. If the current maximum Acre is null after all the possible Acres have been examined, the Den will make a move in a random direction. It will first attempt to move the range of its vision. However, if that Acre is occupied, it will check the Acre that is closer and so on until it can make a move. If all the Acres are occupied, the Den makes no move, but the Honey is still consumed.

Starvation
Starvation occurs when the amount of Honey a Den carries drops to 0. The Den has died and will be removed from the Populus.

Populus
Populus is the class representing the population of the Dens (intelligent agents) in the simulation. The collection is maintained as a linked-list. In Phase One, a Populus object has data members for maximum size of the Populus, current size of the Populus, pointer to the first Den object in the linked-list, and pointer to the Land. There are also data members for the maximum values for some of the data members of the Dens.

The object also has methods for calculating the average vision of the Populus, calculating the average metabolism of the Populus, checking the Populus to remove any Den that has died, replacing a Den that has died, shuffling the order of the Dens, and methods to access the data members.

Checking Dens
Each Den is checked for death by starvation, by sending a message requesting to the Den requesting its condition. If the Den returns a message indicating that it has starved to death, the Den is removed from the Populus. After all Dens have been checked, new Dens are randomly created and placed in the Land to replace the Dens that have died.

Shuffling
In order to simulate a pseudo-simultaneous activity of the Dens, the order of the Dens in the Populus is randomized at the end of each season. Otherwise, Dens that were towards the head of the list would have an advantage not related to the fitness of the Den or the environment over Dens that occurred later in the Populus list.
3.3 Phase Two
In Phase Two, the framework is similar to Phase One, except that the size of the Populus is variable instead of constant. Instead of being automatically replaced, new Dens are created by the mating of two adjacent Dens. All other aspects of the simulation are the same as Phase One.

Erehwon
The only change needed to Erehwon, the simulation class, is in the sequence of the season. A segment needs to be added to allow mating of the Dens. The new sequence is

- Dens (agent) move or gather Honey (resource).
- Dens are checked for starvation.
- Dens look for mates and if one is found, attempt to mate.
- Unoccupied Acres regenerate Honey at a predetermined rate.
- The order of the Dens in the Populus is shuffled.

Acre
Unchanged from Phase One.

Land
Unchanged from Phase One.

Dens
The Dens in Phase Two must now, in addition to finding and gathering Honey, search for a mate, attempt to mate, and produce offspring. To begin the process of mating, the Den must be mature, which is defined as having at least as much Honey as it possessed when created.

Note: This threshold for mating will be raised during Phase Five.

Searching for a Mate
Each mature Den examines the four adjacent Acres in the X-Y axis for a mate. If there are more than one it will pick the most attractive among those that have not already mated. Attraction is the sum of the wealth, carrying capacity, vision, and metabolism.

This is a modification of the mating scheme in Sugarscape. In Sugarscape, Agents will mate with every adjacent mature Agent. At first, this may appear to be a significant difference. However, due to the transfer of resources from parent to offspring, Agents rarely mate more than once per turn.

Attempting to Mate
If a potential mate is found, there needs to be an empty space adjacent to either Den to place the offspring. If none is available, mating fails – no offspring is produced.

Producing Offspring
If mating is successful, to produce the offspring, the new Den randomly selects from either parent to get its vision, metabolism, and Honey capacity. The new Den also gets a
third of each parent’s Honey. Parents each lose half of their Honey. The lost Honey is to simulate the physical cost of mating.

Note: The mating cost will be removed during Phase Five.

**Populus**
Unchanged from Phase One, except the method to check each Den for death from starvation.

**Checking Dens**
This method will now only remove dead Dens from the Populus. It will not create new Dens to replace those that have died. New Dens are created by mating.

### 3.4 Phase Three
Phase Three adds a second resource, Nectar to the Acres. Now Dens must maintain an inventory of Nectar and Honey. If the inventory of either Honey or Nectar falls to 0, the Den will die of starvation.

**Erehwon**
No major changes to the Erehwon class.

**NectarAcre**
A new subclass of Acre, NectarAcre, is added. NectarAcre adds data members for nectar, nectar capacity, and nectar growth rate. It adds accessor methods for those data members. The method for regenerating resources is modified to support the new resource.

![Figure 3.2 – Object Model for Acre](image)

**Land**
Support for Nectar Acres is added

**NectarDen**
A new subclass of Dens, NectarDens, is added. NectarDen adds data members for nectar metabolism, nectar carried, nectar capacity, and initial nectar carried. Several methods in Den were modified to support the second resource, Nectar.

![Figure 3.3 – Object Model for Den](image)
Overview of Erehwon

Moving
As in Phase One and Phase Two, Dens that are on Acres with resources will not move
but will stay and gather available resources up to their carrying capacity.

Since there is a second resource, the Dens that move now must calculate which
unoccupied NectarAcre within its vision along the X-Y axis provides the greatest welfare.
It moves to that Acre.

Welfare
Welfare is based on which resource is most valuable to the NectarDen when it is looking
for a NectarAcre to occupy. The idea is to compare the time for starvation from Honey
and the time for starvation from Nectar. Whichever is less is the more valuable resource
and the NectarDen will give greater weight to that resource. The below formula is used
to calculate the welfare of a Den.

\[ W(h, n) = \frac{m_h}{m_h + m_n} \times m_h + \frac{m_n}{m_h + m_n} \times m_n \]

W is the welfare of the Den
wh is the Den’s wealth in Honey
wn is the Den’s wealth in Nectar
mh is the Den’s metabolism with respect to Honey
mn is the Den’s metabolism with respect to Nectar
m is the sum of mh and mn

This is the Cobb-Douglas functional formula[4][10]. To calculate the welfare of a
prospective site, add the resources of the site to the Den’s wealth. This is because we
assume that the Den effectively has unlimited carrying capacity.

Starvation
If either the Honey or Nectar inventory of a Den falls to 0, that Den will die of starvation.

Populus
Support for NectarDens is added.

3.5 Phase Four
Phase Four adds the ability for Dens to trade resources. Dens may now trade Honey for
Nectar and Nectar for Honey if the conditions for trading are met.

Erehwon
No major changes to the Erehwon class.

NectarAcre
As is Phase Three, this is now the NectarAcre class, a subclass of Acre.

Land
No major changes to the Land class.
TraderDen
A subclass of NecatrDen, TraderDen, is added. TraderDen adds no new data members. The methods for mating are modified to support TraderDens. The following methods are added to support trading: trading, make trade, calculate Marginal Rate of Substitution (MRS), and calculate price.

![Object Model for Den](image)

Figure 3.4 – Object Model for Den

Trading
Each adjacent NectarAcre is examined. If it is occupied, the TraderDen will attempt to make a trade with its Trader Den.

Make Trade
Dens trade according to the following rules.
- Each TraderDen calculates its Marginal Rate of Substitution (MRS). If the two MRSs are not equal, then continue.
- The direction of trade is as follows: Nectar goes from the Den with the higher MRS to the Den with the lower MRS. Honey goes in the opposite direction.
- Using the MRSs, the price $p$ is calculated as shown below.
- The exchange rate is calculated as follows.
  - If $p > 1$, then $p$ units of Nectar for one Honey.
  - Else, then $1/p$ units of Honey for one Nectar.
- The two trader dens will exchange Honey for Nectar and Nectar for Honey as long as they both feel they are improving their welfare. Trading is also stopped if the MRSs would crossover (i.e., the Den that began the trading with the higher MRS has its MRS go below the MRS of the Den that began the trading with the lower MRS). This will prevent two TraderDens exchanging Honey for Nectar, then exchanging Nectar for Honey, or the reverse.

Calculate Marginal Rate of Substitution
The Marginal Rate of Substitution of a Den is calculated as follows.

$$MRS = \frac{m_h w_n}{m_n w_h}$$

MRS is the Marginal Rate of Substitution of the Den

- $w_h$ is the Den’s wealth in Honey
- $w_n$ is the Den’s wealth in Nectar
- $m_h$ is the Den’s metabolism with respect to Honey
- $m_n$ is the Den’s metabolism with respect to Nectar

Marginal Rate of Substitution is the amount of one good, in this case Honey, that is required to compensate a Den for giving up an amount of another good, in this case Nectar\[4][10].
Calculate Price
The price for Honey is calculated according to the below formula. The price for Nectar will be in units of Honey. Depending on the direction of trade, the price for Nectar will be multiple of the price for Honey or a fraction of the price for Honey.

\[ p(MRS_A, MRS_B) = \sqrt{MRS_A MRS_B} \]

p is the price
MRS_A is the Marginal Rate of Substitution for Den A
MRS_B is the Marginal Rate of Substitution for Den B

The MRS and price are determined between at the time of the trade. Both the MRS and the price will be different for every interaction. It is not set by a global authority or function.

Overall, trading in Erehwon is considered a form of the Edgeworth barter process but is more decentralized in that the prices change with each round of trading. The Edgeworth barter process describes the mutually beneficial exchange of goods between two individuals [4][10]. Another similar bartering model among autonomous agents is the Albin and Foley model, described in Barriers and Bounds to Rationality, by Peter S. Albin [1]. Although it is a cellular automata (CA) model, it uses similar a similar scheme for trading as in Sugarscape and Erehwon.

Populus
Support is added for TraderDens.

3.6 Phase Five
Phase Five expands Phase Four by adding the ability for the Dens to alter their own behavior over the course of the simulation through a simple learning process.

Erehwon
No major changes to the Erehwon class.

NectarAcre
No major changes to the NectarAcre class.

Land
No major changes to the Land class.

ScholarDen
ScholarDen is a subclass of TraderDens. ScholarDen adds a data member for brain. The move method is modified. The following methods are added: move to food, move to mate, move to land, and learn.
Move
If the ScholarDen has enough wealth, it will choose from among the following behaviors: move to food, move to mate, and move to land. It will use a proportional selection scheme based on the current weights the ScholarDen gives to each behavior. The current weights are found in the brain, which is implemented as an array.

If the ScholarDen is not wealthy enough, it will choose move to food.

Move to Food
This is the old move method of the super classes of ScholarDen.

Move to Mate
The ScholarDen will count the number of ScholarDens in each direction along the X-Y axis to the limits of its vision and move in the direction that has the greatest number. It will attempt to move as far as possible (i.e., the range of its vision) but will move less than its vision if the target NectarAcre is occupied.

Move to Land
The ScholarDen will count the number of ScholarDens in each direction along the X-Y axis to the limits of its vision and move in the direction that has the least number. It will attempt to move as far as possible (i.e., the range of its vision) but will move less than its vision if the target NectarAcre is occupied.

Learn
Based on the ScholarDen’s experience of the previous season, the ScholarDen will update the brain to reflect that experience. If the welfare of the ScholarDen did not increase, it will increase the preference for the behavior to move to food. If the ScholarDen did not find a mate and was mature, it will increase the preference for move to mate. If the ScholarDen could not find food or mating failed because its neighborhood was too crowded, it will increase the preference for move to land.

Populus
Support is added for ScholarDens.
4 DATA COLLECTION AND OBSERVATIONS

As each phase was implemented, the simulation was run a number of times on various parameter sets. This was to allow collection of data, verify the implementation, and allow observation of the simulation and data collected. During each phase, the simulation was run a minimum of one thousand times. Each simulation was run for at least 250 seasons. The simulations in Phase One and Two were run for 500 seasons.

At each phase, the results were compared to those obtained by Epstein and Axtell in Sugarscape at similar phases. If the results were similar, this would validate that phase of Erehwon. This aspect only applies to Phases One through Four, as learning was not implemented in Sugarscape.

During each phase, incidental observations about the results, not directly related to the validity of the model, were also made.

4.1 Phase One

Phase One involved an invariant population size of Dens that looked for Honey and moved to Honey, made a random move if no unoccupied Acre with Honey was located, or consumed Honey if occupying an Acre with Honey. If a Den’s amount of Honey carried ever drops to zero or less, it dies. A new randomly generated Den replaces any Den that die.

According to Axtell and Epstein, the population should become fitter over the course of the simulation [4]. By this, the average metabolism of the population should be lower and the average vision of the population should be greater at the end of the simulation compared to their values at the beginning of the simulation. To compare the results from Erehwon against the results from Sugarscape, the simulation was run one thousand times for each of the sets of parameters listed in Table 4.1.

<table>
<thead>
<tr>
<th>Parameter Set</th>
<th>Percent Honey</th>
<th>Population Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>400</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>400</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>400</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other Parameters*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max Vision</td>
</tr>
<tr>
<td>Max Metabolism</td>
</tr>
<tr>
<td>Honey Carrying Capacity</td>
</tr>
<tr>
<td>Seasons</td>
</tr>
</tbody>
</table>

*common to all parameter sets

Table 4.1 – Initial Parameter Sets for Phase One

Overall, the results of these runs show the population becoming fitter. However, the results for a few individual simulations did not always show a fitter population with respect to vision. That is, the average vision of the population was less than the average vision at the start of the simulation.
The following table summarizes the results of the batch runs on the parameter sets.

<table>
<thead>
<tr>
<th>Parameter Set</th>
<th>Average Vision</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>Final</td>
<td>%D</td>
<td>Num &lt;</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3.498</td>
<td>3.852</td>
<td>10.1%</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3.502</td>
<td>4.155</td>
<td>18.6%</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3.500</td>
<td>3.921</td>
<td>12.0%</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AVERAGE METABOLISM</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initial</td>
<td>Final</td>
<td>%D</td>
<td>Num &gt;</td>
</tr>
<tr>
<td>1</td>
<td>2.501</td>
<td>2.185</td>
<td>-12.6%</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2.496</td>
<td>1.244</td>
<td>-50.2%</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2.495</td>
<td>1.411</td>
<td>-43.4%</td>
<td>0</td>
</tr>
</tbody>
</table>

Notes

% D  Percent change in parameter.
NUM < Number of simulations where the final parameter was less than the initial parameter.
NUM > Number of simulations where the final parameter was greater than the initial parameter.

Table 4.2 – Results of Running Erehwon on Phase One

An interesting observation is that the Populus became fitter in parameter set two than in parameter set three, even though there was more Honey Acres available. One reason this may have been was the larger supply of resources allowed less fit Dens a greater chance of survival. The reason that population did not become very fit in parameter set one is the scarcity of resources. Since there was a larger population of Dens than supply of Acres with Honey, even very fit Dens would die. Only those lucky enough to be created on a food square (not necessarily the fittest) would survive.

4.2 Phase Two

Phase Two involved a variable population size of Dens that gathered Honey, looked for Honey and moved to an unoccupied Acre with Honey, made a random move if no unoccupied Acre with Homey was located. If a move was made, an amount of Honey equal to the Den’s metabolism was consumed. If the amount of Honey that a Den carries becomes zero or less, the Den dies. This is the same as Phase One. The difference between Phase One and Phase Two is the replacement mechanism for Dens that die. Dens that are adjacent may possibly mate depending on their maturity. The population size is not fixed. It may decrease to zero or increase beyond the initial size. See the design section for a more detailed explanation.

Similar to Phase One, the population should become fitter over the course of the simulation. What is of interest in this phase is the carrying capacity of a particular environment. The carrying capacity of the environment is the level at which the population size levels out. As in Phase One, the simulation was run one thousand times for each of the parameter sets below.
Table 4.3 – Parameter Sets for Phase Two

As in Phase One, the runs did show the population becoming fitter. An interesting observation was the greater number of simulations where the average vision of the population declined during the course of the simulation compared to Phase One for parameter set one. This may have been due to the scarcity of resources and the lucky few that were initially created on those Acres with Honey surviving to pass on their genes. As in Phase One, the lucky Dens would not necessarily be the fittest Dens. However, overall the population is becoming fitter at greater rate than in Phase One.

Table 4.4 – Results of Running Erehwon in Phase Two

The other result of interest was the carrying capacity of the environment. Summarized in Table 4.5 is the final population size for each parameter set. The data confirms the intuition that a larger supply of Honey supports a larger population size.
Another observation is the distribution of the final populations in the parameter sets display in Figure 4.1. From this one may suppose that not only does a larger supply of Honey support a larger population, it also increases the chances of reaching the equilibrium population size.

![Population Distribution](image)

**Figure 4.1 – Population Distributions for Parameter Sets 1, 2, and 3**

The observation from Phase Two is that mating increases the rate at which the population becomes fitter. Though not commented on by Axtell and Epstein, several other researchers have observed the phenomena.

Since the population is becoming fitter and reaching an equilibrium value, this provides evidence that Phase Two has been implemented properly.

### 4.3 Phase Three

Phase Three builds on Phase Two by adding a second resource, Nectar. In Phase Three, a Den must possess a positive inventory of Nectar, in addition to maintaining a positive inventory of Honey. The Dens now move to the unoccupied Acre that will benefit them the most. Also, the Acres can now have Honey or Nectar, Honey and Nectar, or no resource. The Nectar resource acts just like the Honey resource in terms of resource regeneration. See the design section for a more detailed explanation.

Since the computation of the Den’s welfare significantly increased the running time of the simulation only 1000 runs of the following parameter set were performed.
After the runs, the following data was collected. The values, unless otherwise indicated, are the average of one thousand runs. The results are considered anomalous if the final average characteristic does not show improvement over the course of the simulation. For example, if the final average vision in a simulation is lower than the initial average vision, this would be counted as an Anomalous Vision.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>400</td>
<td>508.808</td>
</tr>
<tr>
<td>Average Vision</td>
<td>3.504</td>
<td>5.724</td>
</tr>
<tr>
<td>Average Honey Metabolism</td>
<td>2.499</td>
<td>1.005</td>
</tr>
<tr>
<td>Average Nectar Metabolism</td>
<td>2.499</td>
<td>1.004</td>
</tr>
<tr>
<td>Anomalous Visions</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Anomalous Honey Metabolisms</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Anomalous Nectar Metabolisms</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.7 – Results of Running Erehwon in Phase Three

According to Epstein and Axtell, the Dens should have a lesser chance of survival with two resources than with one [2]. This was verified by running ten simulations on a predefined Land with Honey and Nectar and running ten simulations on the same predefined Land with only the Honey. The averages of the final populations were compared. The average of the population with just Honey was 913.1. The average of the population with both Honey and Nectar was 778.4. This supports Epstein and Axtell’s claim and shows that implementation of Erehwon up to Phase Three is in agreement with Sugarscape.

### 4.4 Phase Four

Phase Three builds on Phase Three by adding the ability for Dens to trade the resources Honey and Nectar with other Dens. Trading will only occur between adjacent Dens and if both Dens decide that the trade will benefit them. See the design section for a more detailed explanation.

As in Phase Three, the added computation of trading significantly increased the time for a simulation to run. Therefore, only a thousand runs of the following parameter set were performed. The parameter set was the same as the parameter set in Phase Three to verify that trading would increase the carrying capacity of the Land.
Data Collections and Observations

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Honey and Nectar Metabolism</td>
<td>4</td>
</tr>
<tr>
<td>Vision</td>
<td>6</td>
</tr>
<tr>
<td>Initial Populus Size</td>
<td>400</td>
</tr>
<tr>
<td>Honey and Nectar Percentage</td>
<td>45</td>
</tr>
<tr>
<td>Seasons</td>
<td>500</td>
</tr>
</tbody>
</table>

**Table 4.8 – Parameter Sets for Phase Four**

After the runs, following data was collected. The values, unless otherwise indicated, are the average of one thousand runs.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population Size</td>
<td>400</td>
<td>506.323</td>
</tr>
<tr>
<td>Average Vision</td>
<td>3.499</td>
<td>5.691</td>
</tr>
<tr>
<td>Average Honey Metabolism</td>
<td>2.497</td>
<td>1.008</td>
</tr>
<tr>
<td>Average Nectar Metabolism</td>
<td>2.499</td>
<td>1.010</td>
</tr>
<tr>
<td>Anomalous Visions</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Anomalous Honey Metabolisms</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Anomalous Nectar Metabolisms</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4.9 – Results of Running Erehwon in Phase Four**

Figure 4.2 shows the distribution of the final population sizes for the runs. This includes runs from Phase Three. This data does support the claim that trade will increase the carrying capacity of the Land. This may be due the random creation of the environment. The trading methods were checked again for correctness in terms of code. To verify the correctness of the code the simulation was run several times while printing the results of each trade to the standard output. No trades were observed that violated the rules of trading.

![Population Ranges](image)

**Figure 4.2 – Population Distributions in Phase Four**
To see if the random environment may have affected the results, the simulation was rerun for both Phase Three and Phase Four using a predefined Land (see Figure 4.3). Since the population was quickly reaching equilibrium with this Land, only 250 seasons were specified instead of 500. For each of the final 150 seasons the population was recorded. The average and standard deviation of those 150 seasons were calculated. The simulation was run 5000 times without trading and 5000 times with trading.

The results, in Table 4.10, do not support the claim that trading allows an environment to support a larger population size. The results, both average and standard deviation are virtually identical.

<table>
<thead>
<tr>
<th></th>
<th>Nectar (No Trading)</th>
<th>Trading</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runs</td>
<td>5000</td>
<td>5000</td>
</tr>
<tr>
<td>Average Pop</td>
<td>777.07</td>
<td>776.81</td>
</tr>
<tr>
<td>Stand Dev</td>
<td>21.24</td>
<td>21.29</td>
</tr>
</tbody>
</table>

Table 4.10 – Additional Results of Running Erehwon in Phase Four

At this point, given the data and the code is correct, there may be a problem in one or more of the algorithms used for trading and price setting. Further investigation is in progress, including email to Robert Axtell requesting the source code for the original Sugarscape. Other possibilities include altering various parameters and observing the effect. Learning (Phase Five) will still be implemented, but the results will only be valid when compared to Phase Three to see if the population increases with learning.

An interesting observation that may be worth further investigation is the fitness of the population without trading compared to the fitness of the population with trading. The population without trading is slightly fitter than the population with trading. Also, there were no instances of the population having a lower final vision in the simulations without trading. There were three instances of lower final vision with trading. This may be due to trading allowing less fit members a greater chance of survival, which allows them more opportunity to pass on their less fit characteristics. The authors of Sugarscape did not comment on this.
4.5 Phase Five
Phase Five builds on Phase Four by adding two behaviors to the Dens and the ability to choose between the behaviors, look for food, look for a mate, and look for land. The Dens use a proportional selection scheme to choose from among these behaviors. Depending on the results of each season, the preference for one or more of the behaviors is increased. See the design section for a more detailed explanation.

Because of the inconclusive results from Phase Four, the results of adding learning to the population were compared to Phase Three of Erehwon. One thousand runs were made on random environments. After that, 1000 runs were made using a predefined environment. In addition to these runs, runs were made using lower resource levels for both learning and no learning. During these runs, the number of populations that recover were noted to see if there is a significant difference between learning and no learning.

The initial runs of 1000 on random environments and predefined environments produced the results noted in the table below. The initial population was 500 and the length was 500 seasons. For the random environment, the resource setting was 45%.

<table>
<thead>
<tr>
<th>Learning</th>
<th>Average Population</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random Environment</td>
<td>443.83</td>
<td>37.64</td>
</tr>
<tr>
<td>Predefined Environment</td>
<td>568.39</td>
<td>38.84</td>
</tr>
<tr>
<td>No Learning</td>
<td>507.01</td>
<td>21.21</td>
</tr>
<tr>
<td>Predefined Environment</td>
<td>777.10</td>
<td>21.19</td>
</tr>
</tbody>
</table>

Table 4.11 – Comparison of Population Levels
The results in Table 4.11 show that the average population with learning was less than that without learning. Also, of interest are the wider swings in the population with learning. At this point learning appears to adversely affect the carrying capacity of the environment. One reason for this may be that Dens that search for a mate do not gather resources but still burn resources when choosing to look for a mate. The other option, looking for land, was never selected. Even if it was selected, the Dens still did not gather available resources. Had any Den ever chose the looking for land option, it would not survive very long and would not likely have enough wealth to mate even if it did find a mate. However, the Dens that did not learn would likely have enough food to mate once having found a mate.

An incidental observation made during both runs was the short life expectancy of a Den after mating occurs. One reason for this may be the cost imposed on the parent Dens after mating. As stated in Phase Two, a child Den gets one third of the wealth of each parent; parents each lose an additional one sixth of their wealth for a total of one half of their wealth. Another reason may be the low threshold for mating. The threshold for mating is that the Den must possess as much wealth as it possessed when created. Dens that are just at the threshold and mate are very likely to die in the next season.

To see if this high turn over in Dens was having an effect on the results, the mating cost was removed, the thresholds for mating was increased, and the threshold to possibly
select another behavior besides looking for food was increased. The parents now only lost a total of one third of their wealth. The threshold for mating was increased to possessing an amount of Honey and Nectar that was three times the metabolism for Honey and Nectar, respectively. The threshold for allowing selection of alternate behaviors was increased to possessing an amount of Honey and Nectar that was five times the metabolism for Honey and Nectar, respectively. Also, the behaviors for looking for a mate and looking for land were modified to allow the Den to gather any resources present before moving. The runs on lower resources levels all started with an initial population of 500 and ran for 500 seasons. The resources levels varied from 10% to 35%.

Table 4.12 shows the results for the runs. The results are for only the final 250 seasons of each run. The average population is the average population for all 500 runs. The standard deviation is the average standard deviation for the 500 runs. The number of recovered populations is the number of populations with an average of 200 or more for the population size. The average recovered population is the average of those populations that recovered. This number is indicative of the carrying capacity of the environment.

<table>
<thead>
<tr>
<th>Resource Level</th>
<th>No Learning</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Avg</td>
<td>SD</td>
</tr>
<tr>
<td>10%</td>
<td>0.7</td>
<td>0.04</td>
</tr>
<tr>
<td>20%</td>
<td>25.6</td>
<td>2.8</td>
</tr>
<tr>
<td>25%</td>
<td>123.3</td>
<td>13.9</td>
</tr>
<tr>
<td>30%</td>
<td>330.3</td>
<td>25.2</td>
</tr>
<tr>
<td>35%</td>
<td>556.8</td>
<td>17.7</td>
</tr>
<tr>
<td>40%</td>
<td>669.4</td>
<td>12.2</td>
</tr>
</tbody>
</table>

NOTES:  
Avg: Average Population  
SD: Standard Deviation  
Rec: Number of populations recovering to equilibrium values  
Avg R: Average of the Equilibrium Populations

Table 4.12 – Comparison of Recovering Populations

As can be seen from these results, adding the behavior “look for mate” made a significant difference in the number of populations that recovered at lower resource levels. However, if the resource level was too low the population did not recover. This can be seen by the results for the 10% level of resources. And, once the resource level reached 40%, the populations recovered in almost all of the simulations.

Also observed were the average wealth of the Dens and the number of Dens created or born. These were only noted for the simulations that had populations that recovered. In the populations that did not recover, the Dens gathered enormous amounts of Honey and Nectar. Including these would have masked the affects that learning had on the average wealth of the population.
As can be seen from the results of Table 4.13, with the addition of learning the average wealth of the population was slightly lower than the population without learning in the resource levels ranging from 20% to 35%. Intuitively, this would seem to be due to more frequent mating in the simulation with learning versus the simulation with no learning. More frequent mating would lower the wealth of the Dens that were mating and, as a result, offspring would start with lower resources. Therefore the average wealth of the population would be lower. However, the results contradict this. In the resource level of 25%, there was an average of 7522 Dens created during the course of the simulations with no learning and an average of 6858 Dens created in the simulations with learning. Yet the average wealth of the Dens in the simulation with no learning, 8.4, was higher than the average wealth of the Dens in the simulation with learning, 7.1.

At this point, there are only two things that may be said about the simple learning scheme with any degree of confidence. One, learning improves the chances of populations recovering in simulations with lower resources levels, as long as the resource level is high enough to permit recovery. Two, learning imposes a slight cost in terms of personal wealth. The reason for this is unclear at the moment and warrants further investigation.
5 Conclusion

The results of running Erehwon in Phase One, Two, and Three, were consistent with results of the equivalent phases in Sugarscape. This allowed the comparison of the results from Phase Three to the results of running Phase Five. There was an unresolved problem with the results of Phase Four. The problem was the addition of trading did not improve the carrying capacity of the environment or change the size of the swings in the population level from season to season. This led to the comparison being made between Phase Three and Phase Five instead of between Phase Four and Phase Five.

The initial run of Phase Five did not show any gain in the carrying capacity of the environment for Dens with learning. Instead, there was a significant decrease in the carrying capacity of the environment. After adjusting the threshold for allowing different behaviors to be selected and the threshold for mating, the simulations were run, both with and without learning. Initially, it appeared that learning did not alter the results of the simulation, because the carrying capacity of the environment was the same for the simulations with learning and the simulations without learning. However, it was noted that number of populations in low resource simulations that recovered to equilibrium levels was greater for the simulations with learning than those without. It was also noted that the average wealth of the Dens in the simulation with learning was slightly lower than the average wealth of the Dens in the simulation without learning. Surprisingly, this did not directly relate to the number of Dens that were being created in the course of the simulation (i.e., the number of times mating occurred).

Based on the results, two things may be said about the learning scheme presently used in Erehwon with any degree of confidence. Adding learning increases the chances of populations recovering in low resource environments and learning imposes a slight cost to the Dens. Based on the results obtained throughout the project, Erehwon is a viable agent-based model (or artificial society) and that adding learning does affect the simulation, though not always in ways that were anticipated.

The importance of Artificial Life in the study of the various aspects of life increases as further research is done in the field. Even though there is much that can be done to improve the Erehwon model, hopefully, it will eventually contribute something useful to the field of Artificial Life and Agent-Based Computational Economics.
REFERENCES


APPENDIX A  GLOSSARY

Acre: The class that represents a space in the environment

Den: The class for the intelligent agent.

Environment: The two-dimensional array of spaces in which the agents interact with each other and the environment.

Erehwon: The class that is responsible for running the simulation. See Simulation.

Honey: A resource.

Land: The class that represents the two-dimensional array of spaces in the environment. See Environment.

Nectar: A resource.

NectarAcre: The class that represents a space in the environment with two resources (Honey and Nectar).

NectarDen: The class for the intelligent agents that makes use of two resources.

Populus: The class that represents the collection of intelligent agents.

ScholarDen: The class for the intelligent agent that is able to select from different behaviors and adjust its preference for each of the behaviors during the simulation.

Season: One turn or iteration in the simulation.

Simulation: The class that allows parameters to be set for the interaction between the population and the environment, starts those interactions, and displays the results of those interactions.

TraderDen: The class for the intelligent agent that is able to trade one resource for another with neighboring agents.
Correct the algorithm for trading.
Currently, the algorithm for trading does not appear to be correct. According to the Sugarscape model, with trading, the carrying capacity of the Land should increase over no trading. Also, wide swings in the size of the Populus were noted. Currently, in Erehwon, the carrying capacity and swings in the size of the Populus with and without trade are virtually identical.

Improve the appearance of the animation.
The animation needs to be smoother. The animation could be improved by painting after each Den moves, instead of after the season is complete.

Improve the usability of the interface.
The user interface could be improved by placing all the parameter settings in one dialog box.

Improve the flexibility of the simulation.
The flexibility of the simulation could be improved by allowing the user to set more parameters (e.g., mating threshold) at run time.

Add the ability to rerun the previous simulation.
The ability to rerun the previous simulation should be added. This would include monitoring of four or five Dens. Each Den would be assigned a unique shape. This would allow visual tracking of the Den in the Land. Also, a separate window could be opened that would show statistics for each Den.

Include real-time analysis tools.
Including real-time output of simulation data would help the analysis. For example, these could be in the form of line or bar graphs.

Develop a GUI tool for creation of predefined environments and populations.
Currently, predefined environments are created by hand using Microsoft Excel to create the layout, then saving the resulting file as a text file. It is no surprise that this is a tedious process. A GUI tool that would allow creation of the Land would make the process much faster, because the user could see the Land as it would appear in Erehwon during the process of creation.

Add other activities for the Dens to engage in.
Currently, the only activities the Dens engage in are searching for and gathering food, mating, and trading. Examples from Sugarscape, of activities to add would be conflict and cultural identification. Examples not from Sugarscape, of an activity to add would be farming.