

MONASH UNIVERSITY

Agent-based model of bumblebee foraging

FIT3036 Computer Science Project

Michael Nguyen

mhngu10@student.monash.edu.au

Based on Simulation Design (2012) by A.G Dyer, A. Dorin and K.B. Korb

Student Statement

- I have read the university's Plagiarism Policy and Procedures
<http://www.policy.monash.edu/policybank/academic/education/conduct/plagiarism-policy.html>
- I understand the consequences of engaging in plagiarism and collusion as described in University Statute 4.1. Part III – Academic Misconduct
<http://www.monash.edu.au/pubs/calendar/Statutes/Statute04.html#Heading110>
- I have taken proper care of safeguarding this work and made all reasonable effort to ensure it could not be copied.
- I acknowledge that the assessor of this assignment may for the purposes of assessment, reproduce the assignment and:
 - provide to another member of faculty;
 - and/or submit it to a plagiarism checking service;
 - and/or submit it to a plagiarism checking service which may then retain a copy of the assignment on its database for the purpose of future plagiarism checking.
- I certify that I have not plagiarised the work of others or participated in unauthorised collaboration when preparing this assignment.

NAME: Michael Hieu Nguyen **STUDENT ID:** 22042962 **DATE:** 23/05/2012

Contents

1.	Introduction.....	3
2.	Project Plan.....	3
2.1.	Overview	3
2.2.	Risk Analysis	3
2.3.	Resource Requirements	4
2.4.	Project Components.....	5
2.5.	Schedule	6
3.	External Design	7
3.1.	Parameters	7
3.2.	User Interface.....	8
3.3.	Statistics Output.....	9
3.4.	Performance.....	9
4.	Internal Design	10
4.1.	Software Architecture	10
4.2.	World	11
4.3.	Agents	12
5.	Test Plan and Report	13
5.1.	Introduction	13
5.2.	Tools	13
5.3.	Test Report	13
6.	Interpretation and Analysis	14
6.1.	Stay Strategy Analysis	15
6.2.	Learn Strategy Analysis	16
6.3.	Switch Strategy Analysis.....	17
6.4.	The effects of plant density.....	18
6.5.	The effects of T1 and T2 distribution	18
6.6.	The effects of the number of bees in a hive	18
6.7.	The effects of flower reward chance	19
6.8.	The effects of the switch time.....	19
6.9.	Experimental Results.....	19
6.10.	Interpretation	32
6.11.	Conclusion.....	35
	References.....	35
A.	Netlogo Simulation Source Code	36

1. Introduction

Computer models can help us understand our world better by simulating behaviour based on certain rule sets that we observe. In particular, it excels in biology because it can track the seemingly random nature of flora and fauna and allow us to extrapolate from that information to predict future trends.

This project deals with bumblebees and their relationship between flowers and their hive. Bees set off from their hive to harvest nutrients from flowers and return it to the hive. In nature, no 2 hives and its bees are the same and it can be observed that among hives certain bees exhibit different behaviour when tasked with foraging for nutrients.

Created and run inside computer software NetLogo, this model aims to provide reasoning for these phenomena. Its goal is to observe how bees interact with flowers based on a variety of parameters and compare different foraging techniques to see which one yields greater results for the hive. In doing so we hope to explain why hives sometime contain bees of different strategies.

This project is based off “Simulation Design” (2012) by A. G Dyer, A. Dorin and K.B. Korb that outlines much of the necessary biological constraints and sets up how bees interact with flowers, plants and the hive for this model.

2. Project Plan

2.1.Overview

The goal is to write a computer model simulation that shows how bees forage for nutrients and how different strategies for these bees help or hinder the hive. These bees are capable of learning what flowers reward them which affect what flowers they frequent most. Bees forage normally (fly around, find flowers, harvest, return to hive) for a certain period of time, before switching their strategy. The 3 main strategies to be implemented are “Stay”, “Learn” and “Switch” – all of which are detailed in “Simulation Design” (2012). Meaningful data is to be collected that will allow us to make conclusions about the bee foraging process.

NetLogo (<http://www.netlogo.com>) will be used to write and run the simulation. NetLogo is an agent-based programming language that is ideal for computer modelling because of its simple syntax and in-built graphical displays.

Though the model is to adhere to the general scenario outlined in Simulation Design (2012), much of the creativity and processes are up to our own choosing.

The project is scheduled to be completed in roughly 11 weeks. At the end of those 11 weeks the computer model written in NetLogo, an accompanying workbook, a test report on validation, and a final report analysing the findings of the model will be delivered on a CD.

2.2.Risk Analysis

Unfamiliarity with NetLogo:

The agent-based programming paradigm of NetLogo and its specific language construct is unfamiliar and unlike any way to approach programming so far experienced.

Unfamiliarity in the Integrated Development Environment (IDE) may lead to ideas and concepts for the model not being fully realised. Although extensions exist for NetLogo that allow it to work with more familiar languages, there appears to be very little support and documentation behind a lot of them.

Risk Reduction Strategy:

NetLogo includes a vast library of models that simulate all kinds of behaviours and interactions. The library will serve as the foundation to understanding the syntax and paradigm of NetLogo. There is detailed technical documentation on the NetLogo website that will aid in understanding NetLogo

2.3.Resource Requirements

NetLogo Software v5.0 (<http://ccl.northwestern.edu/netlogo/>)

The model is to be written and run in NetLogo, an open-source piece of software that is both an integrated development environment (IDE) and an agent-based programming language. NetLogo is used because it is suited for modelling a complex system such as our model and provides built-in tools for displaying graphics. Version 5.0 is required as the simulation makes use of the latest plotting tools.

Compatible Computer

The latest stable release of NetLogo (version 5.0) runs on a wide variety of computers and operating systems provided Java 6 or higher is installed, including Windows XP and greater, Mac OS X 10.4 and greater and Linux. Alternatively, the NetLogo model can be exported into a Java web applet that can be run on any system with a compatible web browser and Java installed.

Source Code Management

Like all large software based project, good source code management (SCM) is paramount to success. Though SCM is best suited to teams and this project is to be completed by an individual, one feature of SCM is version controlling which allows those working on it to revert files back to its previous state in case something goes wrong. It also provides an online repository for the project code to be pulled and worked on multiple systems.

My source code management tool of choice is GitHub.com (<http://www.github.com>), an online repository that uses “Git” as its distributed revision control. The project will be hosted in a private repository that will be inaccessible by the public.

2.4. Project Components

The entire project has 4 deliverables and certain tasks that need to be done before the tasks can be completed. The main tasks can be summarised as so:

1. Create the design document outlining goals, schedule and initial design.
2. Learn and familiarise NetLogo and agent-based modelling.
3. Implement model from design doc to NetLogo.
4. Add to workbook detailing work completed simultaneously.
5. Write test reports to validate model.
6. Write final report on findings of model

The Design Document is both an outline of the project and a “bible” of sorts to consult on. It sets out what is required, why it’s to be done and how it’s to be done. This document should be regularly consulted while implementing the model to ensure that the project stays on track.

Learning NetLogo is an important step. Familiarity with programming is already a given, but NetLogo is an entirely new language and new programming paradigm that must be learnt before the model can be implemented.

Implementation of the model can only begin when task 1 and 2 are sufficiently completed. The task will involve realising the concepts of the design doc in NetLogo code.

The workbook is a document that is completed simultaneously alongside implementing the model – its purpose is to serve as a log that records and rationalises all decisions made in the model. Choices that differ from Simulation Design (2012) are included with reasoning as to why that decision was made.

The Test Report is documentation that performs various tests on our completed model to ensure accuracy and stability. Various software testing methodologies including white-boxed and black-boxed testing such as specification-based testing will be used to detect software bugs and inaccuracies.

The Final Report is the culmination of all the work completed thus far – it is a report that analyses the model’s findings and extrapolates that to make meaningful assertions.

2.5.Schedule

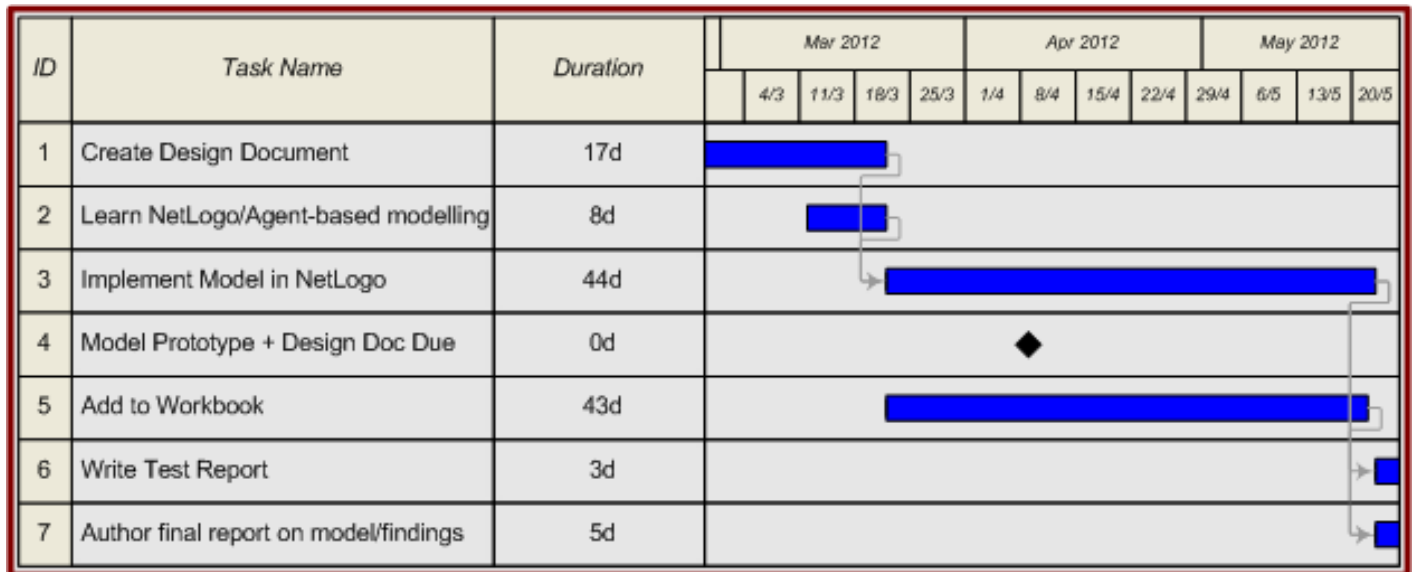


Figure 2.4 – Gantt Chart showing project schedule over 11 weeks

The project was introduced on 28/02/12 and is expected to be completed by 25/05/2012, leaving roughly 12 weeks to complete all deliverables.

The beginning 4 weeks will be spent learning about agent-based modelling and why it's an important tool for modelling behaviour. Time will also be spent on getting familiar with the NetLogo syntax and the agent-based programming paradigm.

Work on the actual implementation of the model will properly start in the week of 26/03/2012 and last for at least 5 weeks.

The design document along with a working prototype of the model is to be presented on 3/04/12. The working prototype will most likely feature movement of the bees on a grid and returning back to the hive.

From there on, the implementation phase will continue on while also working on the Workbook, documenting the entire process.

The week starting 08/05/2012 is when the final demonstration is to take place. The model should be sufficiently implemented at this stage and all code should be wrapped up. The demonstration will most likely take place somewhere in the Faculty of IT, Monash University Clayton Campus supervised by Kevin Korb.

The following final week is when the final report should be written discussing and analysing our findings through the model. It will discuss what we found, what we expected and any interesting emergent phenomena that arose with certain conditions.

3. External Design

3.1.Parameters

Density of Plants

Density of Plants can be configured via a slider in the model that determines how many plants there are on the grid. The slider ranges from 20% to 40% of the grid with increments of 1%. It was determined that the range of 20% to 40% allowed for reasonable testing.

Number of Bees

The modeller will have the ability to vary the number of bees in a hive from 3 to 99 with increments of 3. This can be changed on a slider in the program. The reason for increments of 3 is to allow for an even distribution of strategies for when the user chooses the “Even Split” scenario.

Initial preferences for bees for T1 and T2 flowers

The simulation allows for different starting preferences for all bees for T1 and T2 flowers independently. Both parameters are expressed via a slider that can range from 0% to 100%.

Reward chance for T1 and T2 flowers

Each flower type has different percentages dictating if they offer a reward or not. Simulation Design (2012) originally states T1 begins with 90% and T2 10%. This simulation however allows these percentages to be altered for both T1 and T2 flowers in the form of sliders. However, given that one flower type is initially rewarding and the other not, the range of percentages for T1 go from 51% to 100% and 0% to 49% for T2. These sliders will switch when the second phase starts.

Max nutrition for bees

Simulation Design (2012) states a bee completes one full bout when it reaches 100 nutrients and begins returning to the hive. This model allows for this value to be changed and affects when the bees return their current nutrients to the hive. This is changeable via a slider.

Distribution of T1 and T2 flowers

Plants can either be T1 or T2 plants. This can be determined by a slider that alters the occurrence of T1 flowers as a percentage. As so, choosing 50% for this slider says that of all the patches, roughly 50% are T1 and 50% are T2. Choosing 80% says 80% are T1 and 20% are T2.

Configurable Switch time

After a certain amount of time, the bees enter in a second phase taking on the scenario chosen. This specific switch time can be configured via a slider and can range from 100 ticks to 50000 ticks, with increments of 100.

Strategy scenario when second phase starts

A dropdown box allows the user to select what scenario they want to occur when the second phase starts. The options are those outlined in Simulation Design (2012) and are “All Switch”, “All Stay”, “All Learn” and “Even Split”.

3.2. User Interface

The User Interface is designed to be simple. Using the graphical tools provided by NetLogo, we can configure a UI that is familiar to user of NetLogo and those with a basic level of computer literacy. The screenshot below shows the final UI for the program.

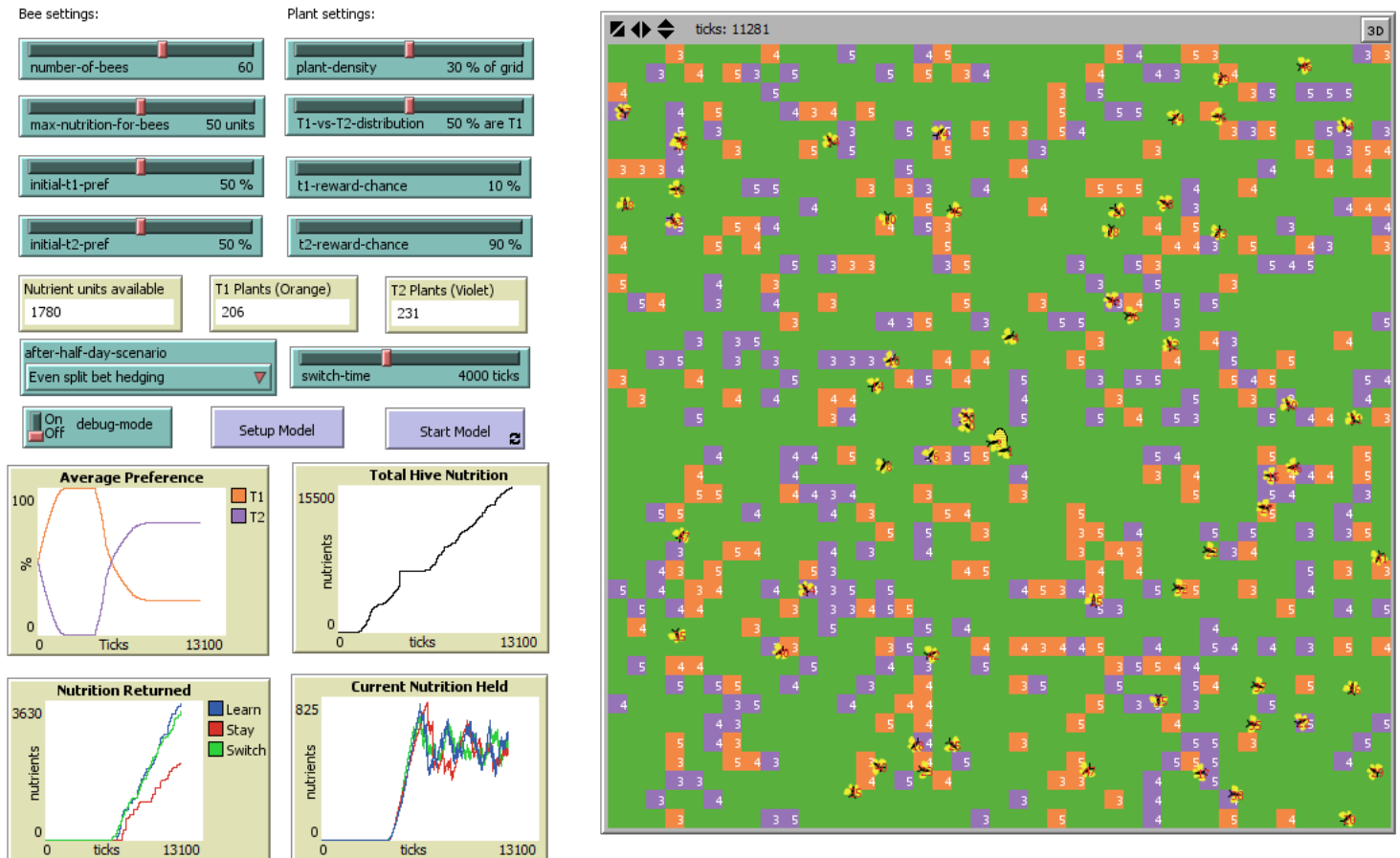


Figure 4.2: User interface of simulation

3.3. Statistics Output

Total Number of Nutrition Returned for the Hive

A simple time plot of the number of nutrients bees return to the hive, irrespective of any strategy.

Nutrients returned per strategy

A time plot comparing the number of nutrients returned by bees of each strategy. In the initial phase, all bees have the same strategy but after that, the individual strategies are introduced and it is paramount to see which strategy is the most effective.

Flower preference

A time plot of the average preference for the two flower types T1 and T2 over time.

Current Nutrition Held

A time plot that plots the current amount of nutrients held by all bees of a certain strategy. This plot will increase when a bee of a certain strategy picks up nutrients, and decrease when the nutrients are returned to the hive.

3.4. Performance

The simulation should be able to run smoothly on most modern day computer systems, but with an increased bee and plant count, the program will become much more processor intensive but its impact to the end user is fairly negligible for reasonable values of bees and plants.

For a further analysis into the performance, we'll look at the time complexity. The `go` method which continually runs, invokes the `refresh-flowers` method that queries all patches to check if they are full of bees, so at the very minimum the code complexity using Big O Notation is $O(M)$ where M is the number of plants.

`go` then performs various operations on all the turtles (bees). One of the larger methods called under the turtle context is the bee movement which has to call on all previous flowers that each bee has visited to check whether or not it should land. This is the most complex the code can get (in terms of time), $O(N*Z)$ where N is the number of bees and Z is the number of plants the bee has visited.

We can however infer that Z , the number of plants the bee has visited, is directly determined by M , the number of plants on the field since the more plants there are on the field, the higher the number of previous plants the bee has visited. Thus, we can conclude that the complexity of this simulation is $O(N*M)$ where N is the number of bees and M is the number of bees.

Of important note is the debug mode feature implemented. When toggled on, the performance of the program takes a large noticeable hit as it starts to print out thousands of debug commands. These debug commands describe all actions in the simulation such as what bee has landed on what flower, how many nutrients are being held by what bee etc. For normal use and where the seeing the information visually through plots and graphics, the debug mode is not recommended.

4. Internal Design

4.1. Software Architecture

The agent-based programming paradigm in NetLogo by design does not have the flexibility to employ many of the conventional software engineering architectures like the Model View Controller. As a result, the structure of the software is fairly straight forward as agent-based programs go. Figure 4.1 shows the structure and attributes of agents and methods in the software.

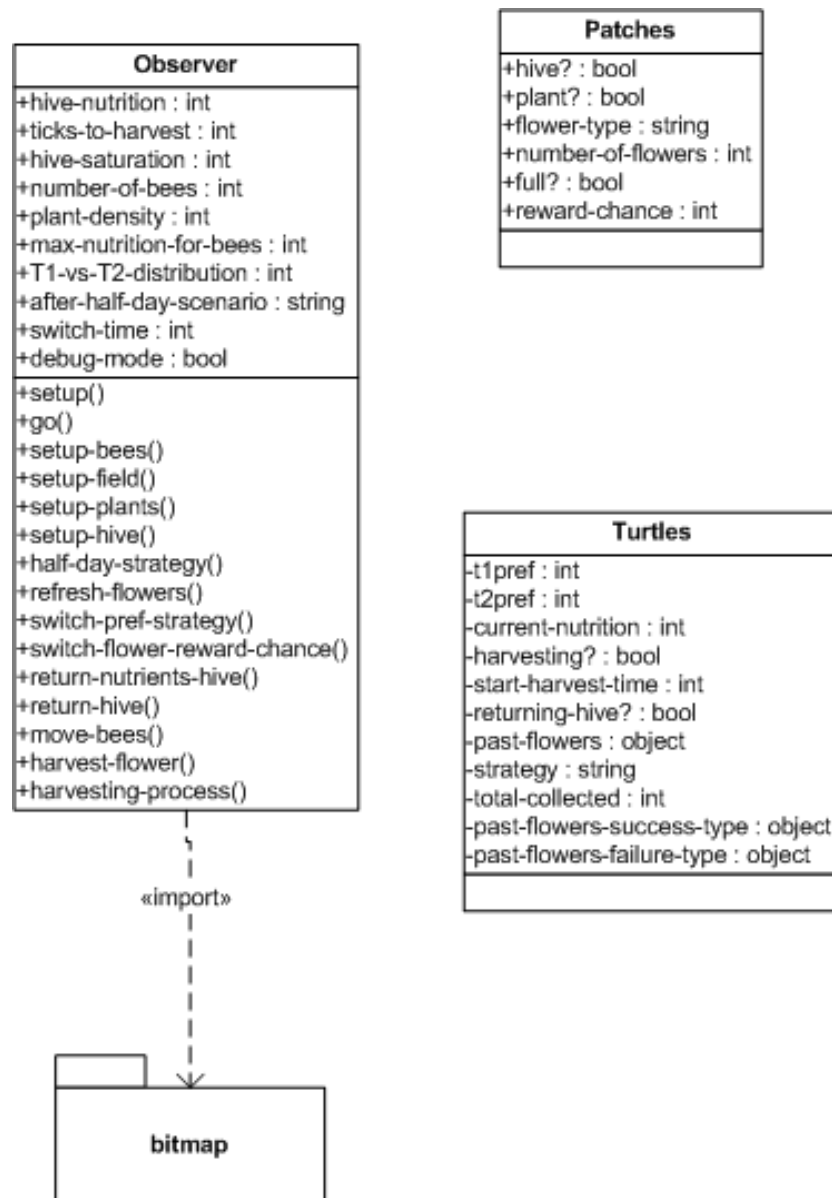


Figure 4.1: Internal design

4.2. World

In NetLogo, all methods are run from the observer context and as such the agents (patches and turtles) have no methods of their own. Below are the global variables and methods.

- **hive-nutrition**: An integer amount of nutrition the hive currently has
- **ticks-to-harvest**: Integer for how long it takes in ticks for a bee to complete harvest.
- **hive-saturation**: An integer representing the max saturation of nutrients for the hive.
- **number-of-bees**: Integer for the number of bees in the hive.
- **plant-density**: Integer that controls how many plants there are.
- **max-nutrition-for-bees**: Integer for the max number of nutrients bees can hold.
- **T1-vs-T2-distribution**: The percentage of flowers that are of type T1.
- **after-half-day-scenario**: String of what scenario we want to recreate.
- **switch-time**: Integer for the number of ticks it takes before phase 2 starts.
- **debug-mode**: Boolean to see output in Command Centre output.
- **setup()**: Method that sets up all the variables etc.
- **go()**: Method that continually runs.
- **setup-bees()**: Method to set up all the bees.
- **setup-field()**: Method to set up the field.
- **setup-plants()**: Method to set up all the plants and flowers.
- **setup-hive()**: Setup the hive including placing the bitmap image.
- **half-day-strategy()**: Method that assigns all bees new strategies once the switch occurs.
- **refresh-flowers()**: Method to allow/disallow other bees onto full flowers.
- **switch-pref-strategy()**: Strategy for "switch" bees to learn off their previous successes.
- **switch-flower-reward-chance()**: Switches the flower reward chance/
- **return-nutrients-hive()**: Return nutrients from bee to hive.
- **return-hive()**: Move bee towards the hive.
- **move-bees()**: Method that moves bees around map and performs appropriate checks of whether or not bees should land.

Bitmap Library

Additionally, the model imports the bitmap extension library which allows an external image to be placed onto the model. It is used to place an identifiable image of a bee hive in the centre of the field.

4.3. Agents

In NetLogo, the primary objects are represented as agents. Agents can be either patches or turtles.

Patches are immobile agents. In this model, patches represent multiple objects which are then identified by specific Boolean variables.

- **hive?** : A Boolean to specify if the patch is the hive.
- **plant?** : A Boolean to specify if the patch is a plant.
- **flower-type** : String of what sort of plant/flower the patch is.
- **number-of-flowers** : Integer for the number of flowers on this patch.
- **full?** : Boolean to represent whether the number of bees currently harvesting on this flower are greater than the number of flowers it currently has.
- **reward-chance** : The reward chance for this patch if it is a plant.

Turtles are mobile agents. In this model, the turtles represent the bees. The bees contain the following variables:

- **t1pref** : Integer representing the bees preferences for T1 flowers.
- **t2pref** : Integer representing the bees preference for T2 flowers.
- **current-nutrition** : Integer for the amount of nutrition bee currently is holding.
- **start-harvest-time** : Integer for the tick timer at which the bee starts a harvest.
- **returning-hive?** : Boolean to indicate whether a bee is in transit returning back to hive.
- **past-flowers** : A list of the bee's previously visited flower's co-ordinates.
- **strategy** : String for the current strategy the bee is currently employing.
- **total-collected** : Integer for the total amount of nutrition the bee has returned to hive.
- **past-flowers-success-type** : A list of the past flower types that have rewarded the bee.
- **past-flowers-failure-type** : A list of the past flower types that did not reward the bee.

5. Test Plan and Report

5.1.Introduction

The test report included details the various testing methodologies and results used to ensure correct functionality of the supplied bee foraging model.

The approach to testing comes with many difficulties, many of which originate from the agent-based programming paradigm. Unlike conventional object oriented languages, agent-based programming at least in NetLogo, does not allow for the construction of testing methods to ensure all components are functional and does not have a large set of tools to accommodate automated testing.

As such, during development a combination of white box testing and black box (but not grey-box testing) was used. Since the simulation is based off Simulation Design (2012), many of the tests are completed to a specification-based standard through acceptance testing to ensure our implementation is working to the description. Stress testing is used to explore the structure and stability of the code base.

5.2.Tools

The NetLogo IDE comes with a variety of tools to help debug and test.

It has an inbuilt syntax checker and a compile-time debugger that can among other things pick up errors that involve calls from an incorrect context (for example, setting a turtle's attribute from a patch context). The inclusion of these tools clear out all of the agent-based bugs that may have arisen through inexperience.

NetLogo provides a useful monitor that allows programmers to inspect the attributes of an agent and its attributes during run time and was used extensively to monitor correct behaviour of flowers and bees. Most if not all tests performed used the agent monitor.

5.3.Test Report

Rigorous testing of the 3 main strategies; "Learn", "Stay" and "Switch" showed that the strategies were working to the description. For these strategies to prove successful shows that the 4th scenario, "All Learn" – a combination of all 3 strategies and an important part of this model to compare strategies is working as expected. The stress test showed that the implementation of the model was running efficiently such that even with all settings and parameters maxed out, a modern computer could run the simulation adequately. For further details on the testing, refer to the included test report documentation. In conclusion, the test reports guarantee the proper functionality of our implementation and we can thus be ensured the findings, hypotheses and conclusions proposed by this report are correct.

6. Interpretation and Analysis

The ultimate goal in creating this model is to observe given a set of parameters how bees exhibiting different strategies can effectively harvest nutrients for the hive. The best strategy is the one that contributes the most nutrients to the hive the fastest – is it a hive where all bees are of a single strategy from Stay, Learn and Switch or is it a mixture of all 3 strategies? Does one strategy outperform another strategy, but only in the short term? How do all the parameters affect different strategies?

This is what the model ultimately seeks the answer. The following experimental tests and analysis explores all of the factors that affect the outcome of each scenario of bee foraging.

Before we begin, it's important to realise the implications of the parameters set so that we can better understand how each strategy might be affected. Reiterating sections 4.2 and 4.3, the important parameters are:

- **number-of-bees:** The number of bees in the hive and that are active in this simulation.
- **max-nutrition-for-bees:** The amount of nutrition the hive collects before the model ends.
- **initial-t1-pref:** The starting preference for T1 flowers for a bee.
- **initial-t2-pref:** The starting preference for T2 flowers for a bee.
- **plant-density:** The percentage of the entire grid that is plants. Simply, the amount of flowers there are on the field.
- **T1-vs-T2-distribution:** The distribution of T1 flowers among all of the plants. As such, 50% on this parameter indicates 50% T1 and 50% T2 flowers.
- **t1-reward-chance:** The reward chance of T1 flowers.
- **t2-reward-chance:** The reward chance of T2 flowers.
- **switch-time:** The time it takes before the second phase begins where the flowers switch their reward chance.

Note:

- The small spike on the hive nutrition returned plot is when the switch occurs where all bees return their nutrients to the hive at once and then switch their strategy.
- The terms “plant” and “flower” are used interchangeably and mean the same thing in the context of this model – a plant of type T1 has 3-5 flowers of type T1, both of which are on a single patch in the simulation.

6.1.Stay Strategy Analysis

The Stay Strategy bees are perhaps the simplest of the three; their insistence to stay with a flower type if they develop >70% preference for it proves to work against their favour when the flowers switch reward chance. Tables 6.9.4 and 6.9.5 demonstrates how much weaker this strategy is compared to the others running under the normal conditions.

The Stay Strategy bees have 2 obvious outcomes after the switch occurs: to have >70% preference for a flower or to have <70% preference. On scenarios where all bees start with 50% preference for both T1 and T2 flowers and the flower reward chances are drastic (90% vs 10% for T1 and T2 respectively), it can be seen that what flower they stay with will drastically affect their success. It is obvious that the worst case for the Stay bees is to develop a >70% preference for T1 plants with 90% reward chance only to keep sticking with T1 after the switch when it has become 10% rewarding.

Table 6.9.12 demonstrates a different scenario where T1 has a 51% reward chance and T2 has a 49% reward chance, after the switch T1 will now have a 49% reward chance. This scenario is not as detrimental to the Stay bees as they no longer stay with a flower that only has a 10% reward chance. A bee with >70% preference for T1 that rewards them 49% of the time is not as detrimental to their success as the previous scenario where T1 is originally 90% and T2 is originally 10%.

Thus, the **difference** in reward chance between T1 and T2 is vital and determines the success of the Stay strategy.

It seems the Stay Bees can only have it bad or worse – is there a scenario that they can do well in? The scenario that Stay Strategy bees succeed the most from is actually when they're not Stay Bees. What that is exactly is when the time to switch rewards is low enough for the bees to not accrue >70% preference for T1 flowers. This ensures that there are no bees that prefer T1 flowers even after it becomes non-rewarding.

As such, by putting a low enough switch time, it effectively turns the Stay Strategy into the Learn strategy as the Stay bees continue learning increasing/decreasing their flower preferences in the same manner as the Learn bees and as in the time before the switch occurred (+1% for a flower that rewards it, -1% for flowers that don't reward it). Table 6.9.13 shows that by simply reducing the switch time from 4000 ticks to 100 ticks there is a 30% improvement in the time it takes to harvest 30,000 nutrients.

Thus we can see the effectiveness of the Stay Strategy is at its greatest when either:

- They develop >70% preference for a flower type that is 51% rewarding which becomes 49% after the switch.
- Have a short enough switch time so that the bee does not develop >70% preference for a flower type and continues learning normally for T2.

Stay strategy Bees **at its best performing in optimal conditions can only perform as well as the Learn Strategy.**

6.2.Learn Strategy Analysis

Bees on the Learn Strategy demonstrate a basic cognitive learning ability by adjusting their preference based on whether they succeed or fail harvesting nutrients. It is also the strategy employed by all bees prior to the reward switch. Table 6.9.3 demonstrates how the Learn strategy operates under the normal control scenario. The results show nothing surprising – their rate of harvesting starts off a little slow after the switch but returns to the rate just before the switch, which is to be expected as they adjust their preference to the new rewarding flower type.

Looking at the average flower preference, it initially looks to both increase T1 and decrease T2 at the same linear rate and vice versa for when the switch occurs. This is consistent with our understanding of the Learn strategy as we have to remember that when a learn bee encounters a rewarding flower it increases its preference for that by 1% as well as decreasing its preference by 1% for the other flower type.

Like the Stay bees, the difference in reward also affects the learn strategy quite a lot as well. If the difference between the 2 reward chances is small then the learn strategy is not very effective, table 6.9.12 shows a scenario for Learn where the 2 reward chances are very close and thus the average flower preference changes very little and hinders the nutrition harvesting.

In essence, the Learn bees are the most consistent of the 3 strategies – they can adept to any situation in their favour but within time. Despite its simplicity, the learn strategy is actually one of the most effective. Table 6.9.4 shows that in the default scenario the learn bees are second only to the Switch strategy and even then the difference is very small between the two strategies. This (surprising) outcome can be attributed to the Learn Bee's consistency; they can and eventually will attain 100% preference for the rewarding flower if the reward chance difference between the two flowers is large enough.

6.3.Switch Strategy Analysis

Bees under the Switch strategy perhaps have the greatest sense of cognitive learning – by realising they have had 4 failures of one type and 4 failures of the other flower, the bee is capable of inverting its preference. From the onset, it seems this is the most effective strategy since these bees are the quickest to adapt to a change. In a scenario where T1 is 90% rewarding and T2 is 10% rewarding, in order for a Switch bee to switch after the second phase, all it needs is 4 T1 failures at 10% and 4 T2 successes at 90% (not in a row) of which are extremely likely. Figure 6.3.1 is a snapshot of when the switch occurs and shows how fast bees on the Switch strategy can adapt to their situation compared to all other strategies.

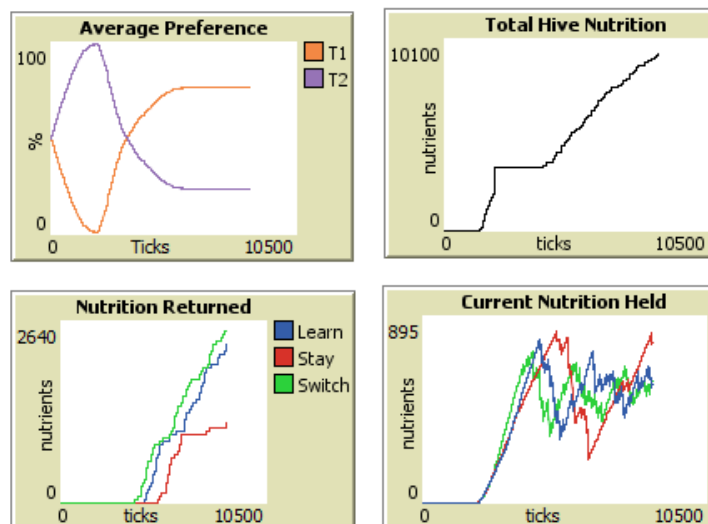


Figure 6.3.1: Early snapshot showing early lead by Switch Strategy

However, upon closer inspection – the switch bees are only useful in switching if the difference in reward chance of T1 and T2 is very big. If T1 and T2 reward chance are very similar after the switch, then the Switch bees are effectively useless because they will be switching their strategy over and over. If the reward chances for T1 and T2 are 51% and 49% respectively, then upon the reward switch the Switch bees will continually be switching their preferences because it's very likely with those reward chances it will discover 4 failures on T1 and 4 success on T2 and then very immediately discover 4 successes on T1 and 4 failures on T2. Therefore, switch bees are very indecisive and cannot get used to the fact that T2 is now the rewarding one.

Figure 6.3.2 below and Table 6.9.12 shows that it stays like this in the long run as the switch bees effectiveness diminishes when the reward chance difference between the 2 flowers is very low.

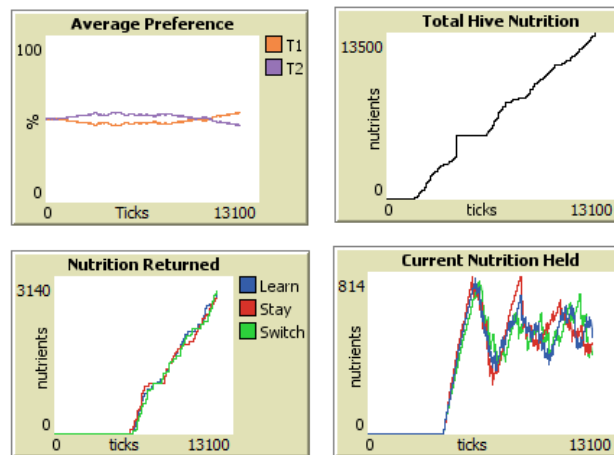


Figure 6.3.2: Long time snapshot of Switch strategy where flowers reward chances are similar

6.4. The effects of plant density

Plant density is the number of plants there are in the field. It may seem obvious that the more plants there are on the field that would be an increase in performance across all strategies since there are more flowers to go around but we must also remember that these flowers affect the learning of the bees both ways. If there are more flowers that may affect the bee's preference in a negative way that may potentially skew the performance of the entire strategy. As such, it is important to analyse the effects of plant density. Tables 6.9.8 and 6.9.9 explore what happens when our default control scenario changes from 30% plant density to 20% and 40% of the grid.

6.5. The effects of T1 and T2 distribution

In the default control scenario of 30% plant density, there is roughly 450 plants. Of those plants, they can contain 3-5 flowers of a particular type T1 or T2. Our model allows us to vary the distribution of T1. By default, the distribution is 50% T1 and T2 flowers making it such that in the control scenario, there are roughly 450 plants, made up of roughly 900 T1 flowers and 900 T2 flowers. Because many of our strategies differentiate between T1 and T2 plants, it will be useful to experiment with our T1 distribution parameter. Would 10% of the flowers being T1 affect how fast some strategies harvest flowers? Tables 6.9.8 and 6.9.9 explore the effects of an uneven flower type distribution on a strategy by looking at what happens if there is a 20% T1 and 80% T2 distribution scenario and 80% T1 and 20% T2 distribution scenario

6.6. The effects of the number of bees in a hive

The effects of the number of bees in a hive may seem obvious – the more bees, the faster the nutrients can be harvested. But one important thing to realise is there are a limited number of flowers on the field and don't allow any other bees to harvest it if it's currently full (where the number of flowers on the plant equals the number of bees currently harvesting it). There is also a brief time where the bees themselves stay on the flower and harvest before moving off increasing the chances of bees flying over a flower, deciding to land but is refused because it is full. As such, it is important to analyse the importance of the number of bees in a hive.

A number of tests were performed using the default control scenario with a modified number of bees. Tables 6.9.10 and 6.9.11 show the tests and results performed with 30 bees and 90 bees

respectively, compared to the regular 60 to observe if there was anything surprising regarding the number of bees and a particular strategy that is outside of the obvious assumption.

6.7. The effects of flower reward chance

Simulation Design (2012) talks of switching rewarding flowers to become non-rewarding flowers and vice versa but what if the line between what defines 'rewarding' and 'non-rewarding' was less distinct as 90% and 10%?

All the strategies outlined in this model in some way work off the percentages of reward for the flowers and in turn that affects the direction their strategy takes – learn bees learn off successful flowers, stay bees can only accrue >70% in the allocated time before switch if they encounter many success and switch bees could potentially be switching multiple times if there is no consistent reward/no-reward situation.

Perhaps the most important parameter of all is the reward chance for T1 and T2 flowers but less so what the actual values are and more so the difference between those two values. What if the flowers had reward chances instead of 90% and 10% for T1 and T2 had 51% and 49% respectively? Would that drastically change the landscape of all strategies? Tables 6.9.5 and 6.9.12 shows multiple runs of where the reward chance is 90% and 10% vs 51% and 49% for T1 and T2 respectively.

6.8. The effects of the switch time

The time in ticks it takes to enter the second phase where flowers switch reward chances is an interesting parameter that on the surface would seem to only affect the Stay strategy but further testing shows that it may perhaps benefit all strategies. The switch time controls what the second phase starts off with initially, in particular the starting flower preferences for T1 and T2. The default scenario of 4000 ticks provides ample time for bees to develop 100% preference for T1 while a short switch time of 100 ticks provides bees with very little time to develop any major preferences.

Table 6.9.13 shows the effects on the stay strategy specifically, while Table 6.9.14 explores the effects of a small switch time on the other strategies.

6.9. Experimental Results

All the experiments performed were performed under the following control scenario unless explicitly stated otherwise.

- **number-of-bees:** 60
- **max-nutrition-for-bees:** 50
- **initial-t1-pref:** 50%
- **initial-t2-pref:** 50%
- **plant-density:** 30%
- **T1-vs-T2-distribution:** 50%
- **t1-reward-chance:** 90%
- **t2-reward-chance:** 10%
- **switch-time:** 4000 ticks

By staying with this control scenario we can then change just 1 parameter to compare and contrast its effect to the model.

Table 6.9.1: All Stay Strategy, 3 runs

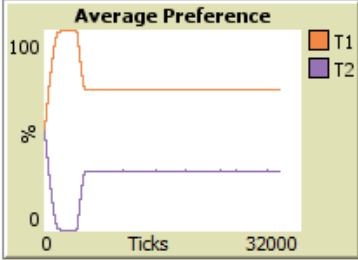
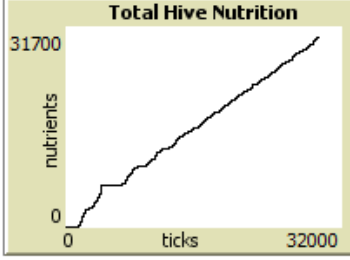
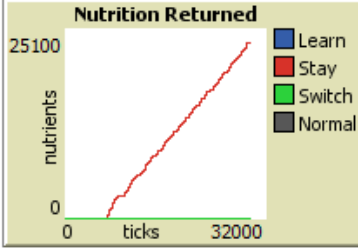
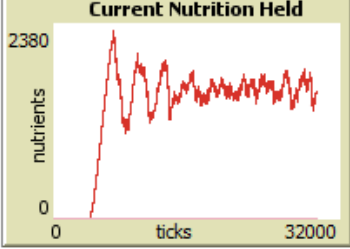
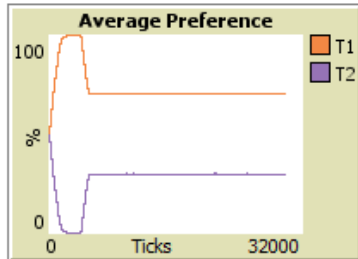
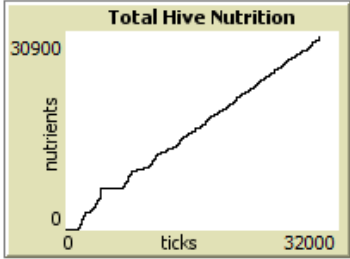
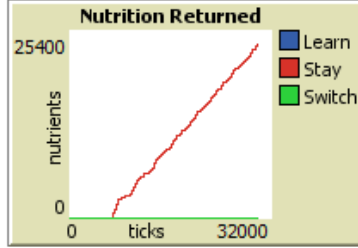
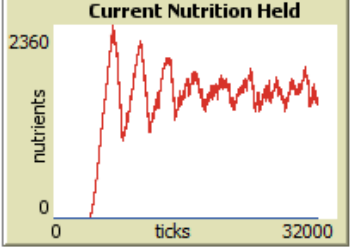
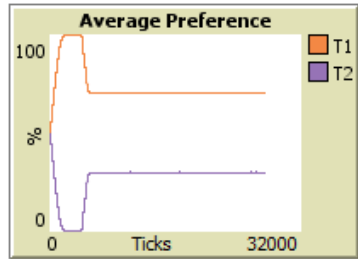
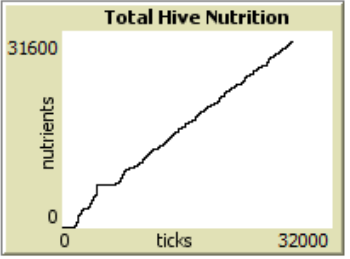
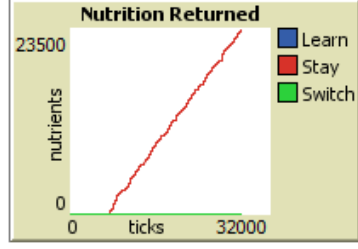
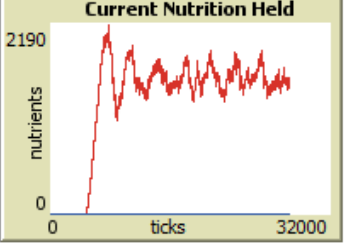
Output		Ticks to harvest 30,000 nutrients
 	 	29388
 	 	29834
 	 	27369

Table 6.9.2: All Switch Strategy, 3 runs

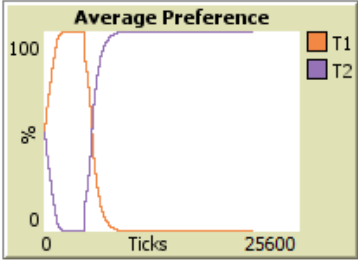
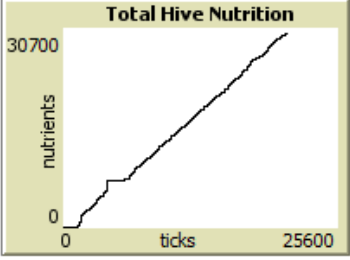
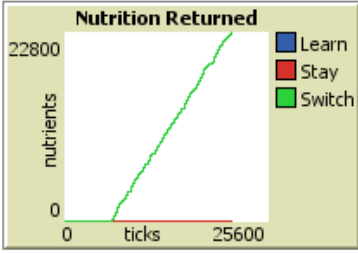
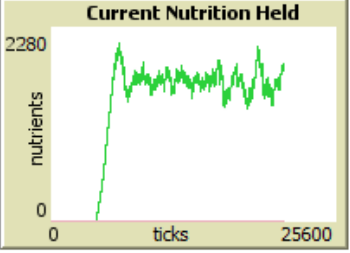
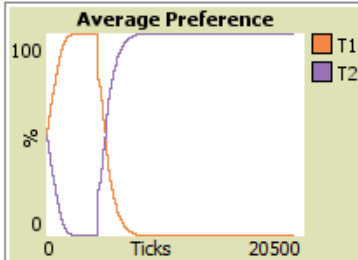
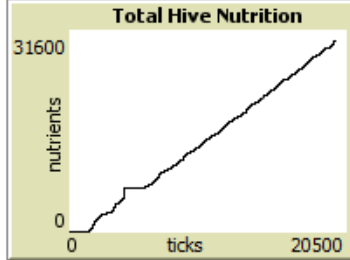
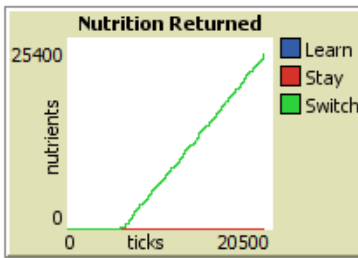
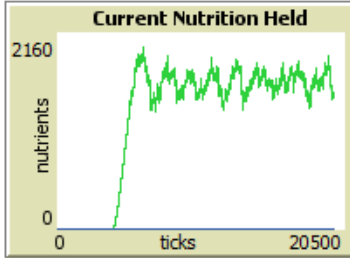
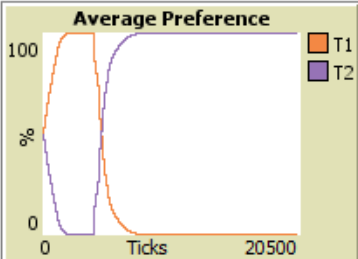
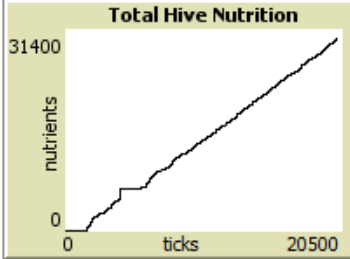
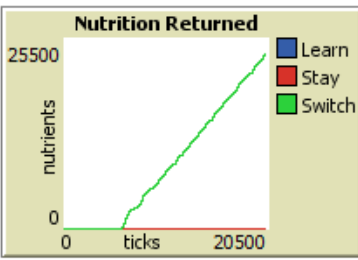
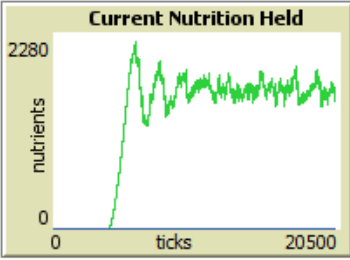
Output		Ticks to harvest 30,000 nutrients
 	 	20926
 	 	19706
 	 	20172

Table 6.9.3: All Learn Strategy, 3 runs

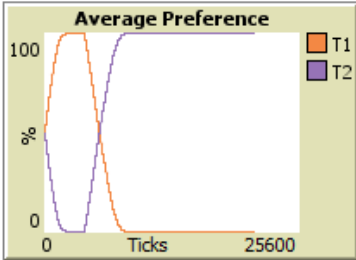
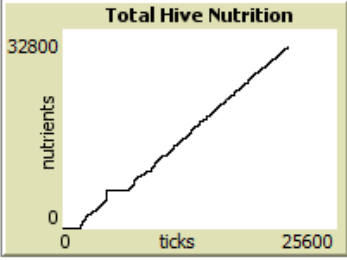
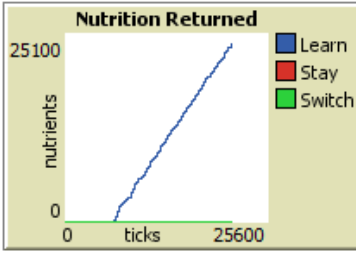
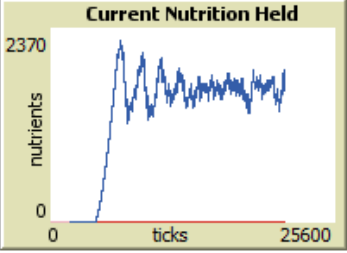
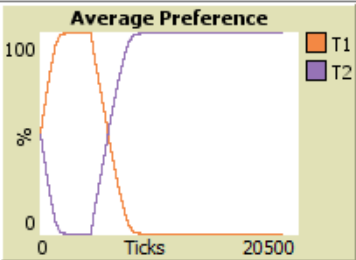
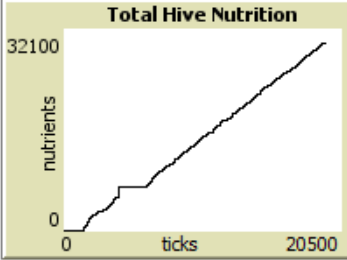
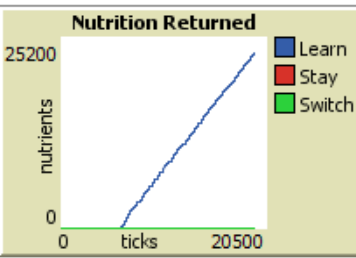
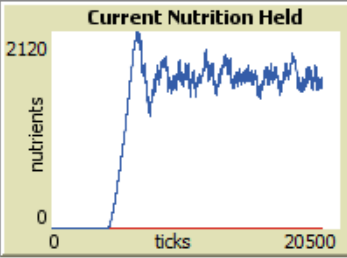
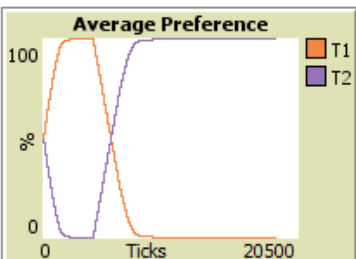
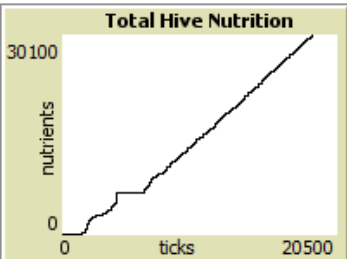
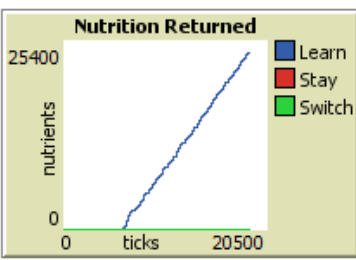
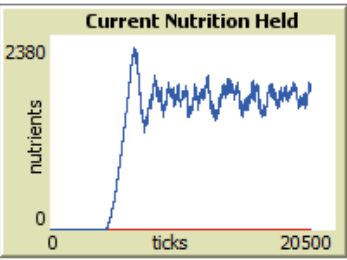
Output		Ticks to harvest 30,000 nutrients
 	 	21091
 	 	19302
 	 	19875

Table 6.9.4: Even split strategy, 3 runs

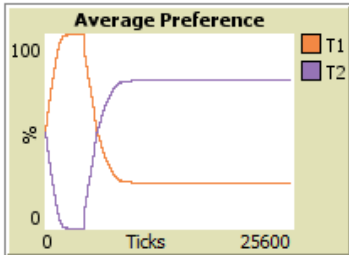
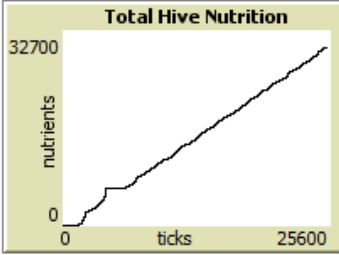
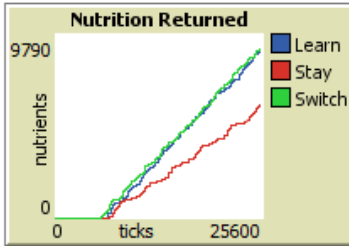
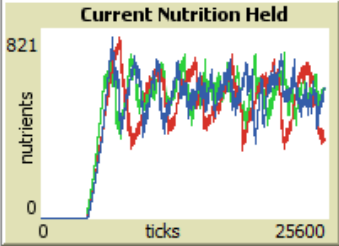
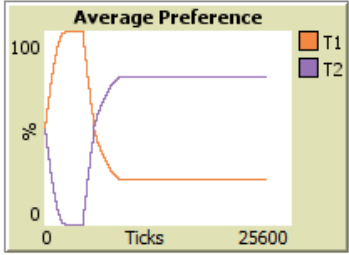
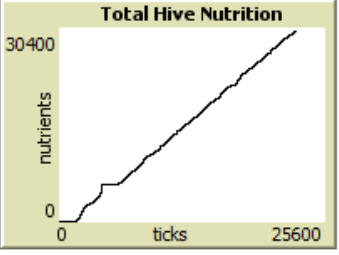
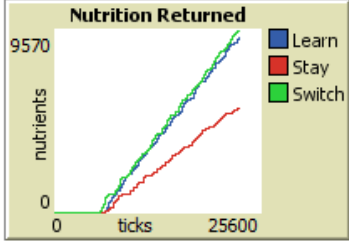
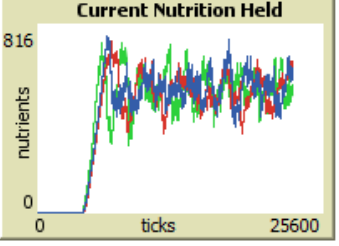
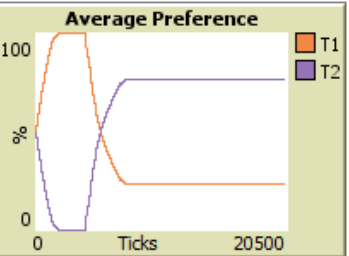
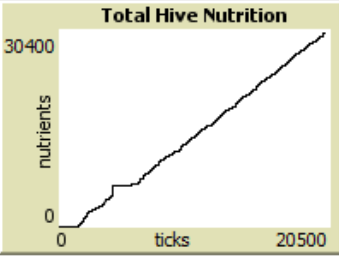
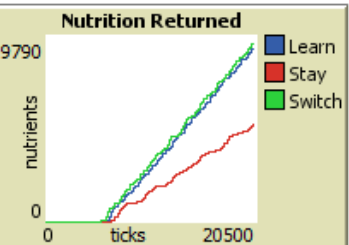
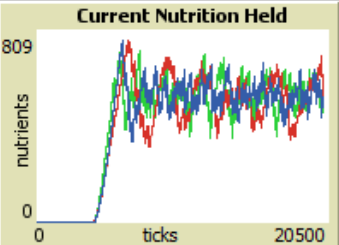
Output		Ticks to harvest 30,000 nutrients
 	 	25240
 	 	22919
 	 	20167

Table 6.9.5: Average from 3 runs from previous 3 tables	
Scenario	Average Time to Harvest 30,000 nutrients
All Stay	28863
All Switch	20268
All Learn	20089
Even split	22775

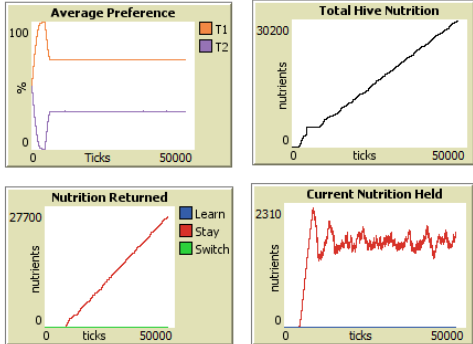
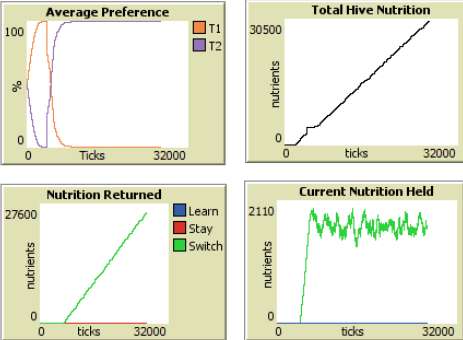
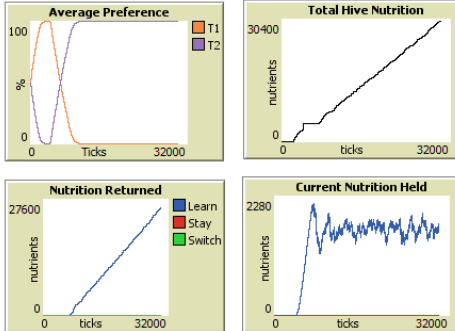
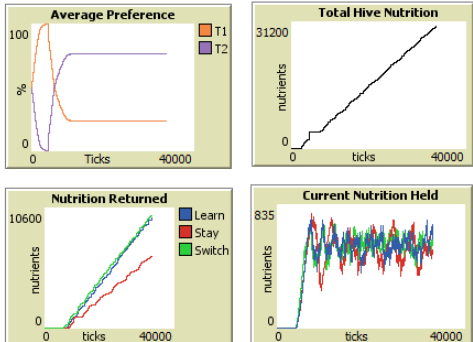
Table 6.9.6: 20% plant density for all strategies	
Output	Ticks to harvest 30,000 nutrients
	<p>Stay strategy: 47355</p> <p>Change from 30% plant density average: $47355 - 28863 = 64.1\%$ slower</p>
	<p>Switch strategy: 28548</p> <p>Change from 30% plant density average: $28548 - 20268 = 40.9\%$ slower</p>
	<p>Learn strategy: 30079</p> <p>Change from 30% plant density average: $30079 - 20089 = 49.7\%$ slower</p>
	<p>Even split: 33225</p> <p>Change from 30% plant density average: $33225 - 22775 = 45.9\%$ slower</p>

Table 6.9.7: 40% plant density for all strategies

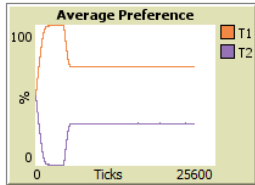
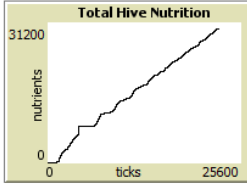
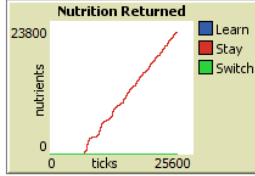
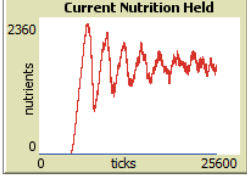
Output		Ticks to harvest 30,000 nutrients
   		<p>Stay strategy: 22671</p> <p>Change from 30% plant density average: $22671 - 28863 = 21.5\%$ faster</p>
		Switch strategy: 14671
		Change from 30% plant density average: $14671 - 20268 = 27.6\%$ faster
		Learn strategy: 15125
		Change from 30% plant density average: $15125 - 20089 = 24.7\%$ faster
		Even split: 16139
		Change from 30% plant density average: $16139 - 22775 = 29.1\%$ faster

Table 6.9.8: 80% of the plants are T1, 20% are T2

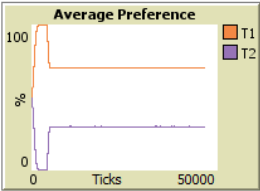
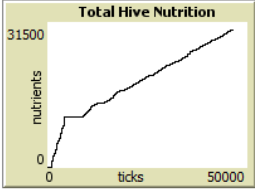
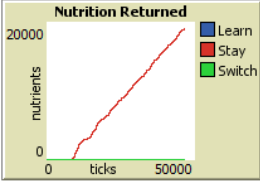
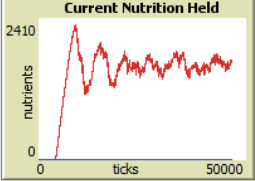
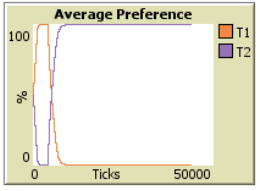
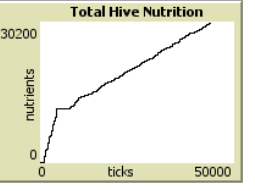
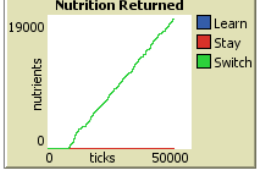
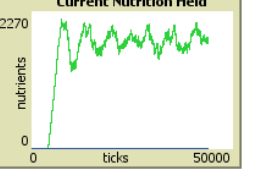
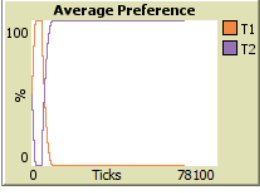
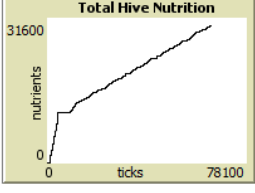
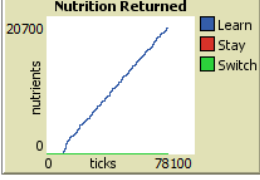
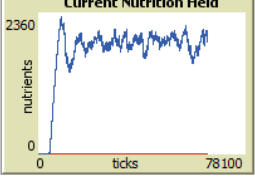
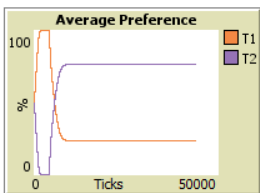
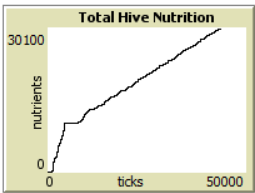
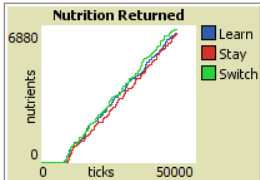
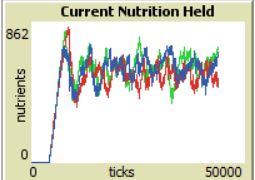
Output		Ticks to harvest 30,000 nutrients
 	 	All Stay: 46607
		Change from 50% spread: 46607 – 28863 = 61.5% slower
 	 	Switch strategy: 43947
		Change from 50% spread: 43947 – 20268 = 116.8% slower
 	 	Learn Strategy: 64039
		Change from 50% spread: 64039 – 20089 = 218.8% slower
 	 	Even split: 43817
		Change from 50% spread: 43817 – 22775 = 92.4% slower

Table 6.9.9 20% of the plants are T1, 80% are T2

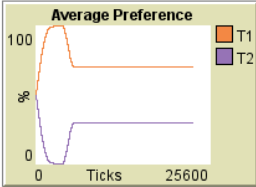
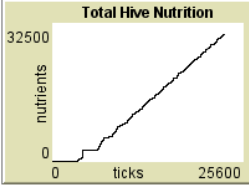
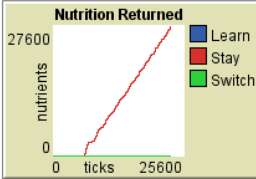
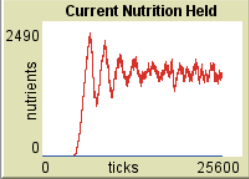
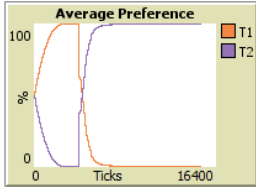
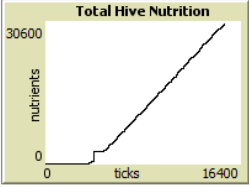
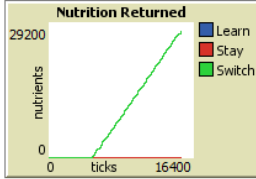
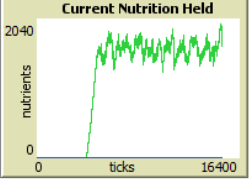
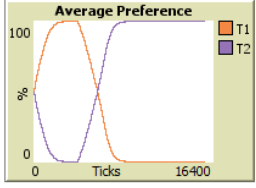
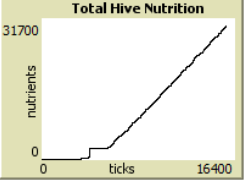
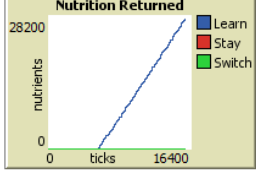
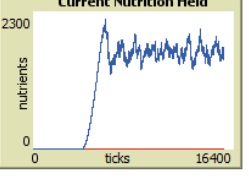
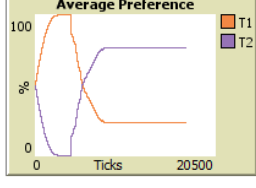
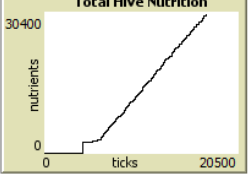
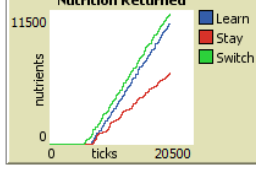
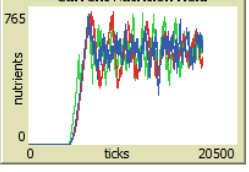
Output				Ticks to harvest 30,000 nutrients
				Stay strategy: 23019
				Change from 50% spread: 23019 – 28863 = 20.25% faster
				Switch strategy: 15028
				Change from 50% spread: 15028 – 20268 = 25.85% faster
				Learn strategy: 15645
				Change from 50% spread: 15645 – 20089 = 22.12% faster
				Even split: 17117
				Change from 50% spread: 17117 – 22775 = 24.84% faster

Table 6.9.10 90 bees in the hive

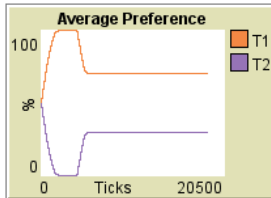
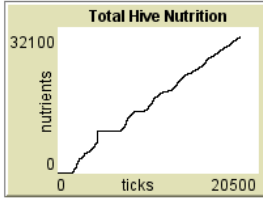
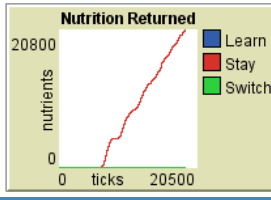
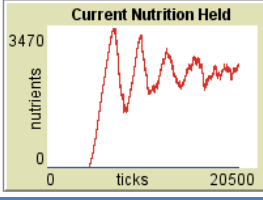
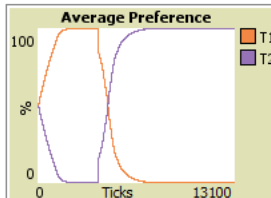
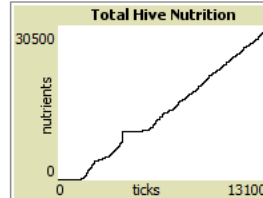
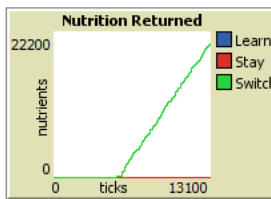
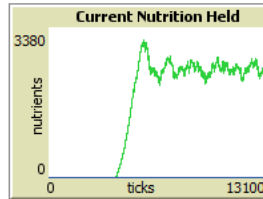
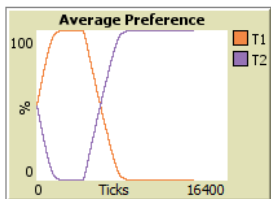
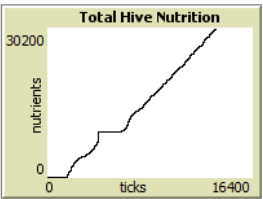
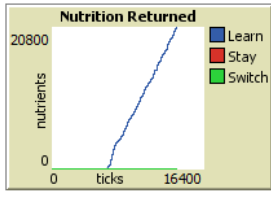
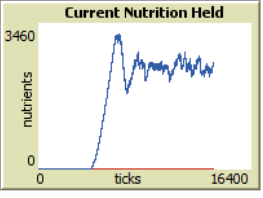
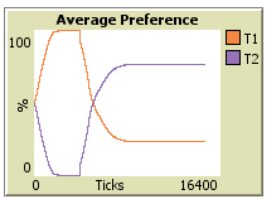
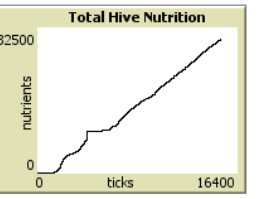
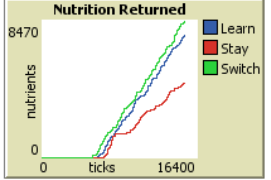
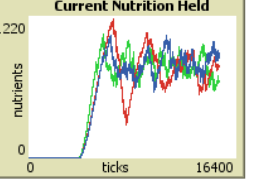
Output		Ticks to harvest 30,000 nutrients
 	 	<p>Stay strategy: 18663</p> <p>Change from 60 bees in a hive: $18663 - 28863 = 35.33\%$ faster</p>
 	 	<p>Switch strategy: 13094</p> <p>Change from 60 bees in a hive: $13094 - 20268 = 35.40\%$ faster</p>
 	 	<p>Learn strategy: 13513</p> <p>Change from 60 bees in a hive: $13513 - 20089 = 32.73\%$ faster</p>
 	 	<p>Even split: 15063</p> <p>Change from 60 bees in a hive: $15063 - 22775 = 33.86\%$ faster</p>

Table 6.9.11: 30 bees in a hive

Output		Ticks to harvest 30,000 nutrients
	<p>Stay strategy: 59384</p> <p>Change from 60 bees in a hive: $59384 - 28863 = 105.74\%$ slower</p>	
	<p>Switch strategy: 37124</p> <p>Change from 60 bees in a hive: $37123 - 20268 = 83.16\%$ slower</p>	
	<p>Learn strategy: 38652</p> <p>Change from 60 bees in a hive: $38652 - 20089 = 92.40\%$ slower</p>	
	<p>Even split: 42615</p> <p>Change from 60 bees in a hive: $42615 - 22775 = 87.11\%$ slower</p>	

Table 6.9.12: Initially 51% T1 reward chance, 49% T2 reward chance

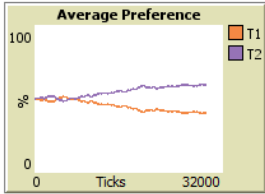
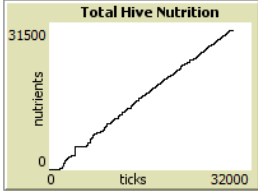
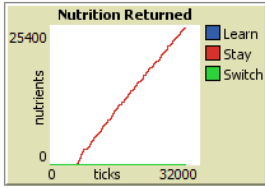
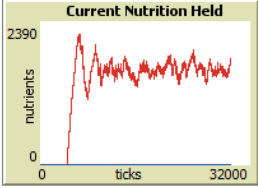
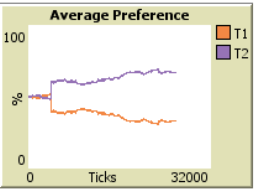
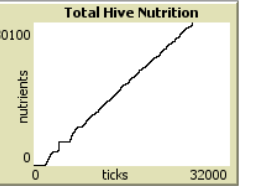
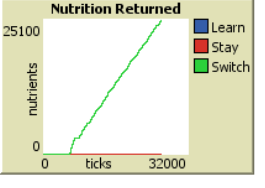
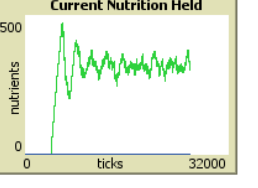
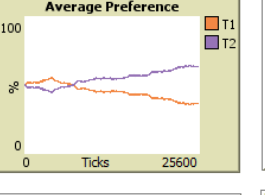
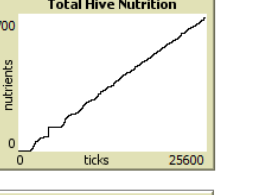
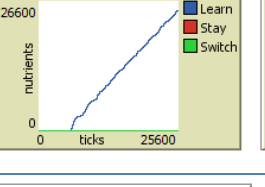
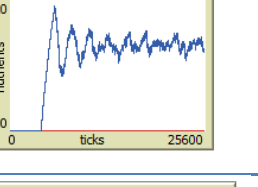
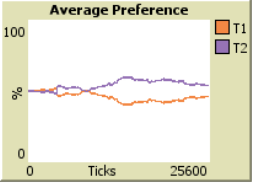
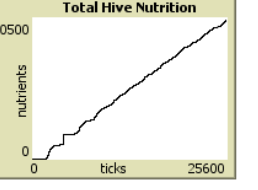
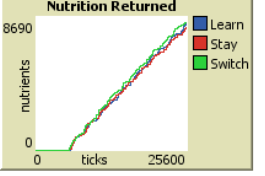
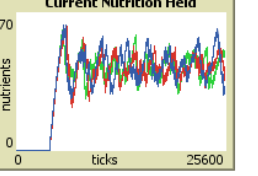
Output		Ticks to harvest 30,000 nutrients
   		<p>Stay strategy: 29095</p> <p>Change from 90%/10%: 29095 – 28863 = 8.04% slower</p>
   		<p>Switch strategy: 25923</p> <p>Change from 90%/10%: 25923 – 20268 = 21.81% slower</p>
   		<p>Learn strategy: 25495</p> <p>Change from 90%/10%: 25495 – 20089 = 26.91% slower</p>
   		<p>Even split: 25482</p> <p>Change from 60 bees in a hive: 25482 – 22775 = 11.89% slower</p>

Table 6.9.13 : Early switch time for Stay strategy at 100 ticks instead of 4000

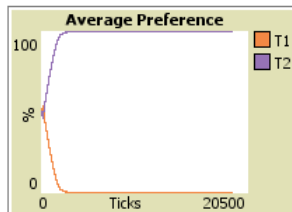
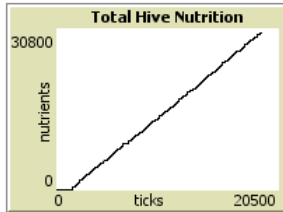
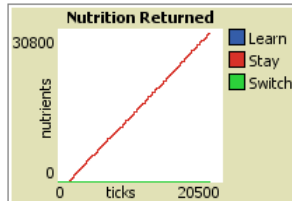
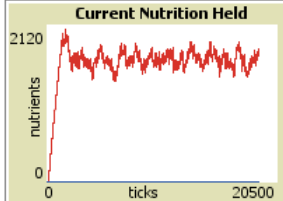
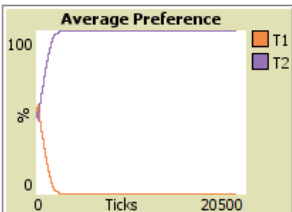
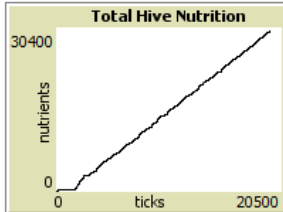
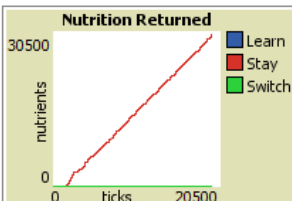
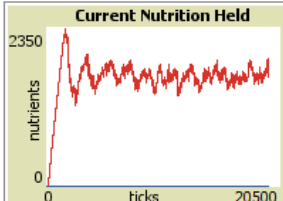
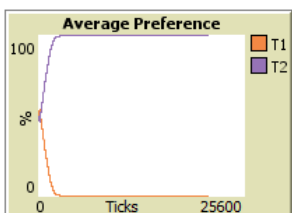
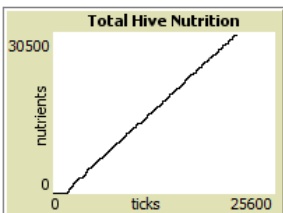
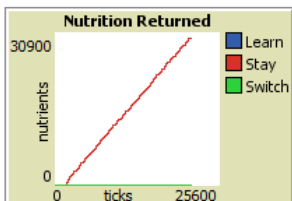
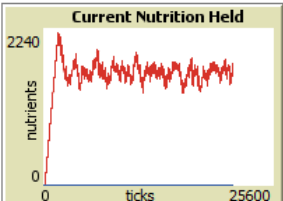
Output		Ticks to harvest 30,000 nutrients
		18914
		
		19875
		
		21148
		
Average for early switch: 19979 Early switch-time Stay vs Long switch-time Stay: 19979 - 28863: 30.77% faster		

Table 6.9.14: Low switch time at 100 ticks for other strategies

Output		Ticks to harvest 30,000 nutrients									
<div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32100</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Switch strategy: 19359</div> <tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>30500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 19359 – 20268 = 4.48% faster</div></td></tr> <tr><td colspan="2"><div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>30200</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Learn strategy:18697</div><tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>31900</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2370</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 25495 – 20089 = 6.92% faster</div></td></tr><tr><td colspan="2"><div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Even split: 17693</div><tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div></td></tr></td></tr></td></tr>		<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>30500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 19359 – 20268 = 4.48% faster</div>		<div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>30200</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Learn strategy:18697</div> <tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>31900</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2370</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 25495 – 20089 = 6.92% faster</div></td></tr> <tr><td colspan="2"><div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Even split: 17693</div><tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div></td></tr></td></tr>		<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>31900</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2370</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 25495 – 20089 = 6.92% faster</div>		<div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Even split: 17693</div> <tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div></td></tr>		<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div>	
<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>30500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 19359 – 20268 = 4.48% faster</div>											
<div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>30200</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Learn strategy:18697</div> <tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>31900</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2370</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 25495 – 20089 = 6.92% faster</div></td></tr> <tr><td colspan="2"><div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Even split: 17693</div><tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div></td></tr></td></tr>		<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>31900</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2370</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 25495 – 20089 = 6.92% faster</div>		<div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Even split: 17693</div> <tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div></td></tr>		<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div>					
<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>31900</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>2370</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 25495 – 20089 = 6.92% faster</div>											
<div><div><div><div>Average Preference</div><div><div><div><div></div><div>T1</div></div><div><div></div><div>T2</div></div></div><div><div><div>%</div><div>100</div><div>0</div></div><div><div></div><div>Ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Total Hive Nutrition</div><div><div><div>nutrients</div><div>32400</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Even split: 17693</div> <tr><td colspan="2"><div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div><div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div></td></tr>		<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div>									
<div><div><div><div>Nutrition Returned</div><div><div><div><div></div><div>Learn</div></div><div><div></div><div>Stay</div></div><div><div></div><div>Switch</div></div></div><div><div><div>nutrients</div><div>10500</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div></div> <div><div><div>Current Nutrition Held</div><div><div><div>nutrients</div><div>871</div><div>0</div></div><div><div></div><div>ticks</div><div>20500</div></div></div></div><div><div><div></div><div>0</div></div><div><div></div><div>20500</div></div></div></div> <div>Change from normal 4000 ticks: 17693 - 22775 = 22.31% faster</div>											

6.10. Interpretation

The above analysis and results explore a large variety of scenarios to see how each parameter affects each strategy.

The effect of number of bees:

The number of bees has proven to not be a vital parameter in affecting the outcome. It certainly has an effect as seen from Tables 6.9.10 and 6.9.11 but is in no way surprising. The more bees in a hive, the faster the hive will harvest nutrients. For 30 bees, there is a drop in performance of around 80-100% which is consistent across all strategies. For 90 bees, the improvements are much closer with 33-35% improvements across all strategies. These are all consistent and shows that despite there

being a limited number of nutrients on the field, the fact that a bee can only land on a flower if it is not full, is negligible and the number of bees does not drastically impact the bumblebee foraging.

The effects of the number of plants:

The outcome of varying plant density has yielded predictable results. The higher the plant density, the faster the hive can harvest nutrients simply because there are more flowers in the field the bees can harvest from. In table 6.9.7, an increase in plant density of 40% shows a consistent improvement of around 21% to 27% increase across all strategies which is in line with our assumptions. Of particular interest perhaps is the 64.1% slower performance for the Stay Strategy in Table 6.9.6 compared to 40%-50% range for the other strategies. This is because of the nature of the Stay strategy; once the switch occurs, all these bees are locked onto T1 flowers. With there being fewer flowers on the field, the stay bee has to now not only find a flower, but it also has to be T1. The other strategies, Switch and Learn can adjust their preferences to the new switch on any of the flowers they encounter (even if they find a T2 flower, their T1 preference can be adjusted unlike the Stay bees). The plant density in conclusion has a marginal effect on the model, as apart from the obvious outcome of fewer flowers resulting in fewer nutrients, it is really only the Stay strategy that is affected when there are fewer flowers.

The effects of flower reward chance:

The outcome of a small difference in reward chance between the flowers doesn't seem to affect the Stay strategy that much affecting it only by 8.04% slower but affects the switch strategy and learn strategy by 21.81% and 26.91% respectively as shown in table 6.9.12. This in turns affects the even split strategy by reducing its performance by 11.89%. The reason it doesn't affect the stay strategy as much is because the stay bees by the time the switch occurs have already developed a strong sense of preference for a particular flower type that is 49% rewarding. For any bee, success is dictated as: flower preference * flower reward chance. The reason why a stay bee is not affected as much is because $70\% * 49\%$ is still greater than $30\% * 51\%$.

The effects of a low switch time:

The effects of a low switch time reveal some interesting results. Table 6.9.14 shows with a small switch time of 100 ticks a 4.48% improvement for the All Switch strategy, a 6.92% improvement for All Learn but oddly enough a 22.32% improvement for the even split. The cause of this is due to the effects of the low switch time on the Stay strategy bees as outlined in Table 6.9.13 which receives on average a 30.77% boost in performance. A low switch time only really helps the Stay Strategy bees because it prevents the bees from accruing a >70% for a flower type that eventually becomes non-rewarding and allows them to learn and develop their preference for T2.

The effects of T1 and T2 distribution:

The distribution of plants is an interesting factor in our tests. Table 6.9.8 shows a scenario where 80% of the plants are T1 and 20% of the plants are T2 and there are wildly different results. The stay strategy has suffered a 61.5% hit to performance, Stay has a 116.8% loss in performance and Switch has a gargantuan 218.8% slower performance. The Stay and Switch strategies affect the even split causing it to perform 92.4% slower. Attempting to analyse this one has been tricky as it goes against any of the preconceived assumptions, but a deeper analysis of it results in some interesting answers.

The reason for the Stay Strategy not failing as bad as the others when 80% of the flowers are T1 is because of how the Stay bees work in the 2nd phase – they have developed >70% preference for T1 flowers and thus whenever they fail on T1 or succeed on T2 their preference for T1 will never fall. With T1 being 80% of the plants, the Stay bees are more likely to land on these plants. Their 10% reward chance is not a factor at all because failing on T1 does not change their T1 preference and neither does succeeding on T2. While the T2 plants have a much higher percentage in rewarding the bees, with there being less T2 flowers on the field, it's likely they will fly over more T1 flowers than T2.

This explains why both Switch and Learn strategies are affected so much more from the T1 distribution – it's not that they're affected so much more, it's that the Stay strategy is affected so much less. In this scenario, the learning of either the Switch bees or Learn bees are not affected – they still do their own thing, but the lack of rewarding T2 flowers hinders their performance. Given the abundance of non-rewarding T1 flowers, it would be very common to see Switch and Learn bees develop 0% preference for T1 flowers.

On the opposite end of the spectrum where there is 80% T2 shows that each strategy is performing much better lining up with our assumptions. What is of interest is the sharp decline for T1 and sharp increase for T2 preferences when the switch occurs. The rate at which both these occurs is much steeper than that of before the switch and the reason for that is simple. With there being so many rewarding T2 flowers, the preferences for T2 rises up much faster causing at the same time T1 to fall much faster.

All of this data and analysis begs the final question – **what is the ideal scenario that will harvest 30,000 nutrients the fastest?**

From the control scenarios in table 6.9.5, All Switch, All Learn and Even split are the 3 viable options. All Stay is only viable in very odd situations we determined above such as a low switch time. But even then, we concluded that the stay strategy at best could only be as good as the learn strategy. As such, looking at the 3 remaining strategies:

- The ideal scenario for All Switch is when the bees have accrued 100% for T1 and then immediately switch allowing the bees to immediately have a 100% preference for T2. As seen from table a bigger distribution of T2 flowers helps the switch bees.
- By working off the idea that the stay bees are learn bees at their best performing, we can then see that the Even Split can essentially become 2/3s Learn and 1/3 Switch. If we look at what scenario the learn bees are best in, we can tune the Even Split to be the best performing scenario. As discussed earlier in section 6.2, learn bees are most effective when the difference in reward chance between T1 and T2 is large.

But even after all of this analysis and final talk of which is the best, we cannot conclude on what strategy is the best because there are best strategies for different scenarios. Changing the switch time or the reward chances might make one strategy the outright winner but that's only for a particular scenario – why stop at changing the reward chances to 90% 10 when you could change it to 99% and 1%? Why stop at changing 1 parameter when we could change 5 of them at the same

time? Coming up with the best strategy for a specific, particular scenario is futile – there are infinite combinations and it wouldn't be fair to say one strategy is definitively "better" than the other.

6.11. Conclusion

Bee foraging techniques and indeed those in the biological world are fascinating because they can never be 100% simulated. What this models shows is that while there are an infinite amount of combination of parameters, none of them can truly predict a real life scenario of bees harvesting. However, what it does show us is that bees and their hive have the capability to change the bee behaviour to best serve the hive for any given number of scenarios. Their surprising learning capabilities while rudimentary, are a fascinating look at how not only bumblebees but the entire biological world are perhaps much smarter than we think.

References

A.G Dyer, A. Dorin and K.B Korb (2012) *Simulation Design*

M.H Nguyen (2012) *Agent-based model of bumblebee foraging test report*

A. Netlogo Simulation Source Code

```
;; This software is the original work of Michael Hieu Nguyen. ID: 22042962
;; This software is submitted in partial fulfillment of the
;; requirements for the degree of Bachelor of Science,
;; Monash University

globals ;; global variables
[
  hive-nutrition ;; amount of nutrition the hive currently has
  ticks-to-harvest ;; number of ticks it takes for a bee to stay on a flower to harvest nutrients
  hive-saturation ;; max saturation of nutrients for hive
]

extensions ;; library extensions
[
  bitmap ;; import bitmap extension to place hive image
]

turtles-own ;; variables for bees
[
  t1pref ;; preference for t1 plants (0 <= t1pref <= 100)
  t2pref ;; preference for t2 plants (0 <= t2pref <= 100)
  current-nutrition ;; current amount of nutrition bee is holding
  harvesting? ;; boolean to state whether the bee is currently foraging
  start-harvest-time ;; variable to store time (tick) when they start foraging
  returning-hive? ;; if bee is returning home
  past-flowers ;; past flowers
  strategy ;; string for strategy
  total-collected ;; total nutrients bee has returned to hive
  past-flowers-success-type ;; list of past successful flower types
  past-flowers-failure-type ;; list of past failed flower types
]

patches-own ;; variables for plants
[
  hive? ;; Boolean to represent if patch is the hive (only true on px/ycor 0 0)
  plant? ;; Boolean to represent if the patch is a plant
  flower-type ;; Type of the plant and its flowers (T1/T2)
  number-of-flowers ;; Number of flowers on this plant
  full? ;; Max number of bees on this flower (might delete)
  reward-chance ;; Chance this plant and its flowers can reward bee
]

to setup ;; initial method run when 'setup model' button is clicked
  clear-all ;; Clear everything for a new start
  if debug-mode [ print "Starting new model..." ]
  set hive-saturation 200000

  set-default-shape turtles "butterfly" ;; make the bees use the "butterfly" graphic default in netlogo
  setup-bees ;; setup the bees
  setup-field ;; setup the area
  setup-plants ;; setup the patches
  reset-ticks ;; reset all ticks
  bitmap:copy-to-drawing bitmap:import "hive.gif" 300 300 ;; make the centre patch have a graphic (300px,300px). does not work if model
  is scaled.

  if debug-mode
  [
    print "Setup complete." type "Final parameters: "
    type "Plant density: " print plant-density
    type "T1's initial rewarding chance: " print initial-t1-pref
    type "T2's initial rewarding chance: " print initial-t2-pref
    type "Time to switch rewards/strategy: " print switch-time
  ]
end

to setup-bees ;; setup all bees
  create-turtles number-of-bees ;; create number of bees based on parameter slider
  [
    setxy 0 0 ;; make them all start in the centre (hive)
```

```

set color yellow ;; the bees are yellw
set t1pref initial-t1-pref ;; set their flower preferences
set t2pref initial-t2-pref
set current-nutrition 0
set ticks-to-harvest 5 ;; time in ticks it takes for bee to harvest nutrient from flower
set harvesting? false ;; boolean to check if the bee is currently in the start of harvesting a flower
set start-harvest-time 0 ;; time in ticks when the bee starts harvesting
set returning-hive? false ;; boolean: if bee is in progress of returning to hive
set past-flowers [] ;; list to keep track of all bees past flowers it's landed on
set strategy "normal" ;; strategy bee starts off with
set total-collected 0 ;; obviously starts off with 0 nutrients
set past-flowers-success-type [] ;; separate list to keep track of its success flower types
set past-flowers-failure-type [] ;; same but for failure
]
end

to setup-field ;; setup the world grid
ask patches ;; ask all patches
[
  set pcolor green ;; make them green (represent the grass - nonfactor in model)
  set plant? false ;; set it by default as not a flower
  setup-hive ;; setup the hive
]
end

to setup-plants ;; method to setup all plants on field
ask patches ;; ask every patch (1600 of them)
[
  if random 100 < plant-density ;; generate a number between 0 and 100 and see if it's under density value %, ie: higher plant density = more plants
  [
    let random-x random-pxcor ;; select a random patch to become a plant
    let random-y random-pycor ;; select a random patch to become a plant

    while [random-x = 0] [ set random-x random-pxcor ] ;; remomve centre patch if it is chosen as a flower (reserved for hive)
    while [random-y = 0] [ set random-y random-pycor ]

    ask patch random-x random-y ;; ask a random patch to become a plant, that patch could be set a flower twice, overriding itself
    [
      set plant? true ;; set boolean as being flower
      set full? false ;; plant is not full of bees
      set number-of-flowers random 3 + 3 ;; random number of flowers between 3 - 5

      ifelse random 100 < T1-vs-T2-distribution ;; set flower type based on parameter
      [
        set flower-type "T1"
        set pcolor orange + 1
        set reward-chance t1-reward-chance ;; set T1's reward chance based on parameter
      ]
      [
        set flower-type "T2"
        set pcolor violet + 1
        set reward-chance t2-reward-chance
      ]
    ]

    if debug-mode
    [
      type "Plant (pxcor, pycor): "
      type "( " type pxcor type " , " type pycor type " )"
      type " is type: " type flower-type
      type " and has: " type number-of-flowers type " flowers"
      type " with reward chance of: " type reward-chance print "%."
    ]

    set label number-of-flowers ;; label for number of flowers on each plant
  ]
]
end

to setup-hive ;; setup hive in centre

```

```

set hive? (distancexy 0 0) < 1 ;; set the middle patch to be the hive
set hive-nutrition 0 ;; set starting nutrition value collected of hive to be 0
end

```

to half-day-strategy ;; method that is called from a turtle context in go method. used for changing all bee strategies at half day.
while [current-nutrition > 0] ;; while loop to teleport bees back to hive
;; so they return all their nutrients. was too complicated making them 'fly' to hive. to make distinction between strategy nutrients from earlier phase.

```

[
  return-hive
  return-nutrients-hive
]

```

set total-collected 0 ;; reset this (total-collected not reset in return-hive) so that the plotting information is accurate for each strategy, remembering all previous total-collected was done under "normal"
set returning-hive? false

```

;; assign strategies based off what the selected scenario is
if after-half-day-scenario = "All Learn"
[
  set strategy "learn"
  if debug-mode [ type "**** Bee: " type who type " has now adopted strategy: " print strategy ]
]
if after-half-day-scenario = "All Switch"
[
  set strategy "switch"
  if debug-mode [ type "**** Bee: " type who type " has now adopted strategy: " print strategy ]
]
if after-half-day-scenario = "All Stay"
[
  set strategy "stay"
  if debug-mode [ type "**** Bee: " type who type " has now adopted strategy: " print strategy ]
]

```

```

if after-half-day-scenario = "Even split bet hedging"
[
  if debug-mode [ type "**** Even split bet hedging chosen. Now evenly assigning different strategies to all bees **** " ]
]

```

```

let bee-id-split (who mod 3) ;; since increment of bees is in 3, can do an even split of strategy
if bee-id-split = 0
[
  set strategy "stay"
  if debug-mode [ type "**** Bee: " type who type " has now adopted strategy: " print strategy ]
]
if bee-id-split = 1
[
  set strategy "switch"
  if debug-mode [ type "**** Bee: " type who type " has now adopted strategy: " print strategy ]
]
if bee-id-split = 2
[
  set strategy "learn"
  if debug-mode [ type "**** Bee: " type who type " has now adopted strategy: " print strategy ]
]
]
end

```

to go ;; Method that runs constantly
refresh-flowers
ask turtles
[

move-bees ;; call move turtles command constantly

```

;; condition: if the bees are currently holding as much as they can hold (or more due to some bug/skipping, return to hive
if (current-nutrition >= max-nutrition-for-bees)
[
  return-hive
  return-nutrients-hive
]

```

```

]

;; condition: once the half day has elapsed, change all bees to their respective strategies. "normal" strategy is their starting off one.
if ((ticks > switch-time) and (strategy = "normal"))
[
    if debug-mode
    [
    ]
    half-day-strategy
]

;; condition: if a bee has strategy "switch" and it has at least 4 successes and 4 failures, see what the history is and maybe learn off that
if ((strategy = "switch") and (length past-flowers-success-type >= 4) and (length past-flowers-failure-type >= 4))
[
    switch-pref-strategy
]
]

if ticks = switch-time ;; switch the reward chance of the flowers when half the day elapses
[
    if debug-mode
    [
        print "" type "**** HALF DAY HAS PASSED. TICKS: " type switch-time print " IS UP. **** "
    ]

    switch-flower-reward-chance

    if debug-mode
    [
        type "**** SELECTED HALF DAY SCENARIO FOR BEES: " type after-half-day-scenario print ". **** "
        print "**** RETURNING ALL BEES TO HIVE AND SWITCHING STRATEGY FOR ALL BEES NOW **** "
    ]
]

if (hive-nutrition >= hive-saturation)
[
    stop
]
tick
end

to switch-pref-strategy ;; method for bees under "switch" strategy to switch flower if they find 4 success and 4 failures
let num-success-t1 length filter [? = "T1"] past-flowers-success-type
let num-success-t2 length filter [? = "T2"] past-flowers-success-type
let num-failure-t1 length filter [? = "T1"] past-flowers-failure-type
let num-failure-t2 length filter [? = "T2"] past-flowers-failure-type

if num-success-t1 > 3 and num-failure-t2 > 3
[
    if t1pref < t2pref
    [
        let temp t1pref
        set t1pref t2pref
        set t2pref temp
        if debug-mode
        [
            type "^ Bee: " type who print " learnt 4 successful T1s, 4 failed T2: Switching preference now..."
            type "^ New t1pref: " type t1pref type " New t2pref: " print t2pref
        ]
    ]
]

if num-success-t2 > 3 and num-failure-t1 > 3
[
    if t2pref < t1pref
    [
        let temp t1pref
        set t1pref t2pref
        set t2pref temp
    ]
]

```



```

    if debug-mode
    [
        type "^ Bee: " type who print " learnt 4 successful T2s, 4 failed T1: Switching preference now..."
        type "^ New t1pref: " type t1pref type " New t2pref: " print t2pref
    ]
    ]
]

set past-flowers-success-type []
set past-flowers-failure-type []
end

to switch-flower-reward-chance ;; method to switch the reward chance of t1 and t2 flowers
let temp t1-reward-chance ;; temp variable to store, refers to global variables
set t1-reward-chance t2-reward-chance
set t2-reward-chance temp

if debug-mode
[
    print ""
    print "**** Switching flower reward chance ****"
    type "**** T1 reward-chance was: " type temp type "% T2 reward-chance was: " type t1-reward-chance print "% ****" ;; Those
variables are intentional
    type "**** T1 reward-chance now: " type t1-reward-chance type "% T2 reward-chance now: " type t2-reward-chance print "% ****"
]

ask patches
[
    ifelse flower-type = "T1"
    [
        set reward-chance t1-reward-chance
    ]
    [
        set reward-chance t2-reward-chance
    ]
]
end

to refresh-flowers ;; method to update whether a flower is full so not allow any other bees to harvest from it
ask patches
[
    ifelse count turtles-here <= number-of-flowers
    [
        set full? false
    ]
    [
        set full? true
    ]
]
end

to return-nutrients-hive ;; method for transferring bee nutrient to hive
if hive? ;; add the bee's current nutrition to the hive and reset it
[
    set hive-nutrition hive-nutrition + current-nutrition
    set total-collected total-collected + current-nutrition

    if debug-mode
    [
        print "" type "++ Bee: " type who type " transferred " type current-nutrition type " nutrients to hive. Current hive total now: "
        print hive-nutrition
        type "-- Clearing Bee: " type who print " past flowers. Setting off to new bout..."
    ]

    set harvesting? false
    set start-harvest-time 0
    set returning-hive? false
    set current-nutrition 0
    set past-flowers []
]
end

```

```

to return-hive ;; method for bees to return back to hive, should be continually run from go
if not returning-hive? ;; only prints once
[
  if debug-mode
  [
    print ""
    type "+ Bee: " type who type " returning to hive with " type current-nutrition print " nutrients."
  ]
]

set returning-hive? true ;; set boolean so this method only prints once
right random 360 ;; random degree from 0 to 360
forward 1 ;; head 1 unit in that direction, to simulate a more 'real' path and not straight to hive
facexy 0 0 ;; face the hive and go
forward 2 ;; 2 Movement speed is the extra speed they get when returning with full nutrition

end

to move-bees ;; Method for bees to fly
let already-been? false ;; already-been is temp variable to determine if bee has already visited this flower before

set label-color red ;; label for their nutrition
set label current-nutrition ;; red label next to them to show current nutrition

let flower-xy (list pxcor pycor) ;; make temporary value of the flower bee is on to check if it's already been

foreach past-flowers
[
  if flower-xy = ?
  [
    set already-been? true
  ]
]

if (plant? and not full? and not already-been?) ;; if you're on a flower and it's not completely full and you haven't already been here
[
  let chance random 100 ;; generate a temp number between 0 and 100 to see if you can land on the flower

  ifelse flower-type = "T1"
  [
    ifelse chance < t1pref ;; if the number generated falls within the bees preference, harvest it
    [
      harvest-flower
    ]
    [ ;; otherwise fly past it
      if debug-mode
      [
        print "" type "Bee: " type who type " flew over but decided not to land on plant ( " type pxcor type ", " type pycor type " ) of type: "
      ]
    ]
  ]
  print flower-type
  fd 1
]
]

[
  ifelse chance < t2pref
  [
    ;;type "Bee " type who type " decided to land on plant ( " type pxcor type ", " type pycor type " ) of type: " print flower-type
    harvest-flower
  ]
  [
    if debug-mode
    [
      print "" type "Bee: " type who type " flew over but decided not to land on plant ( " type pxcor type ", " type pycor type " ) of type: "
    ]
  ]
  print flower-type
  fd 1
]
]
]

```

```

if(plant? and (start-harvest-time + ticks-to-harvest) < ticks and harvesting?) ;; time for the plant to stay on the plant for a certain number
of ticks before moving off
[
  set harvesting? false ;; finished harvesting and can move off
  set start-harvest-time 0 ;; reset the time in which he started harvesting
  if debug-mode
  [
    print "" type "+ Bee: " type who type " finished harvesting plant ( " type pxcor type " , " type pycor print " ). Moving off plant."
  ]
  fd 1
]

if(not harvesting?) ;; otherwise, if the bee has finished harvesting (or hasn't harvested) keep flying around
[
  right random 360 ;; randomly rotate between 0deg and 360deg
  forward 1 ;; move forward 1 unit in that direction, simulates a more realistic movement
  ;; *** CONSIDER ADDING VISION?
]
end

to harvest-flower ;; method called in the context of turtles, from move-bees, for harvesting flower
ifelse count turtles-here > number-of-flowers ;; if the number of bees on that plant is more than the number of flowers it has, do not
harvest
[
  set full? true
]
[
  harvesting-process;; otherwise, there is a free flower so run the procedure to harvest flowers
]
end

to harvesting-process ;; ***** CONSIDER RENAMING ***** Method for bee to attempt and extract the nutrient from the flower for all
strategies
set full? false
if not harvesting?
[
  set start-harvest-time ticks ;; set the time when the bee starts to harvest in order to make him stay there for some time

  if debug-mode
  [
    print ""
    type "Bee: " type who type " strategy: " type strategy type " landed on plant ( " type pxcor type " , " type pycor
    type " ) of type: " type flower-type type " with reward chance of: " type reward-chance
    type "% at tick time: " type start-harvest-time print "."
    type "-- Starting attempt of harvest.."
  ]

  set past-flowers fput (list pxcor pycor) past-flowers ;; add the current flower position it is harvesting onto its list of previous flowers
  (append to start)

  if flower-type = "T1" ;; if the plant the bee landed on is of type T1
  [
    let chance random-float 100 ;; generate a number to see if it can harvest it or not

    ifelse chance < (reward-chance) ;; create reward chance based off plant's reward %
    [
      set current-nutrition current-nutrition + 1 ;; success condition for t1.
      set past-flowers-success-type fput (flower-type) past-flowers-success-type ;; save the type of flower onto its past success types
      (mainly for switch strategy)

      if t1pref < 100 and t2pref > 0 ;; condition to prevent >100% preference for t1 plant
      [
        ifelse strategy = "stay" and t2pref = 70 ;; preventing change if bee is on stay strategy and hit the threshold
        [
          if debug-mode
          [
            print ""
            type "-- Bee: " type who type " failed! No change in current nutrition, still: " print current-nutrition
            type "-- T1 preference: " type t1pref type "% T2 preference: " type t2pref print "%."
            type "-- Bee: " type who print " is on strategy: Stay. Hard wired not to drop below 70%. Not changing preference..."
          ]
        ]
      ]
    ]
  ]
]

```

```

    ]
  ]
  [
    ;; otherwise, if its on a different strategy or not met the stay threshold continue
    set t1pref t1pref + 1 ;; increase T1 plant preference
    set t2pref t2pref - 1 ;; decrease T2 plant preference

    if debug-mode
    [
      print ""
      type "-- Bee: " type who type " strategy: " type strategy type " succeeded! Increase nutrition by 1, current nutrition: " type current-nutrition print "."
      type "-- Increase preference for T1 by 1%, now t1pref: " type (t1pref)
      type "%, decrease preference for T2 by 1%, now t2pref: " type (t2pref) print "%."
      type "-- Adding to bee's past flowers: ( " type pxcor type ", " type pycor print " )."
      type "-- Bee " type who type " past flowers: " print past-flowers
      type "-- Last 4 success types: "

      ;; print out the past 4 success types with appropriate length checking if there is less than 4
      ifelse length past-flowers-success-type < 4
      [
        type past-flowers-success-type
      ]
      [
        type sublist past-flowers-success-type 0 4
      ]
      type " Last 4 failure types: "

      ifelse length past-flowers-failure-type < 4
      [
        print past-flowers-failure-type
      ]
      [
        print sublist past-flowers-failure-type 0 4
      ]
    ]
  ]
]
[
  ;; failure condition for t1
  set past-flowers-failure-type fput (flower-type) past-flowers-failure-type ;; append the failed flower type to the list of failures

  if t1pref > 0 and t2pref < 100 ;; condition to prevent negative preference
  [
    ifelse strategy = "stay" and t1pref = 70 ;; threshold measure for bees on strategy stay
    [
      if debug-mode
      [
        print ""
        type "-- Bee: " type who type " failed! No change in current nutrition, still: " print current-nutrition
        type "-- T1 preference: " type t1pref type "% T2 preference: " type t2pref print "%."
        type "-- Bee: " type who type " is on strategy: Stay. Hard wired not to drop below 70%. Not changing preference..."
      ]
    ]
    [
      set t1pref t1pref - 1 ;; decrease preference for t1
      set t2pref t2pref + 1 ;; increase preference for t2

      if debug-mode
      [
        print ""
        type "-- Bee: " type who type " failed! No change in current nutrition, still: " print current-nutrition
        type "-- Increase preference for T2 by 1%, now t2pref: " type (t2pref)
        type "%, decrease preference for T1 by 1%, now t1pref: " type (t1pref) print "%."
        type "-- Adding to bee's past flowers: ( " type pxcor type ", " type pycor print " )."
        type "-- Bee " type who type " past flowers: " print past-flowers
        type "-- Last 4 success types: "
        ifelse length past-flowers-success-type < 4
        [
          type past-flowers-success-type
        ]
      ]
    ]
  ]
]

```



```

]
]
[
;; failure condition for t2
set past-flowers-failure-type fput (flower-type) past-flowers-failure-type

if t2pref > 0 and t1pref < 100
[
  ifelse strategy = "stay" and t2pref = 70
  [
    if debug-mode
    [
      print ""
      type "-- Bee: " type who type " failed! No change in current nutrition, still: " print current-nutrition
      type "-- T1 preference: " type t1pref type "% T2 preference: " type t2pref print "%."
      type "-- Bee: " type who print " is on strategy: Stay. Hard wired not to drop below 70%. Not changing preference..."
    ]
  ]
]
[
  set t2pref t2pref - 1
  set t1pref t1pref + 1

  if debug-mode
  [
    print ""
    type "-- Bee: " type who type " failed! No change in current nutrition, still: " print current-nutrition
    type "-- Increase preference for T1 by 1%, now t1pref: " type (t1pref)
    type "%, decrease preference for T2 by 1%, now t2pref: " type (t2pref) print "%."
    type "-- Adding to bee's past flowers: ( " type pxcor type ", " type pycor print " )."
    type "-- Bee " type who type " past flowers: " print past-flowers
    type "-- Last 4 success types: "
    ifelse length past-flowers-success-type < 4
    [
      type past-flowers-success-type
    ]
    [
      type sublist past-flowers-success-type 0 4
    ]
    type " Last 4 failure types: "
    ifelse length past-flowers-failure-type < 4
    [
      print past-flowers-failure-type
    ]
    [
      print sublist past-flowers-failure-type 0 4
    ]
  ]
]
]
]
set harvesting? true
]
end

```

