CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Animat Navigation Using Landmarks

Navigation of Simulated Animals Inspired by Bees

Master's thesis in Complex Adaptive Systems

Mathias Carlsson

Department of Computer Science and Technology
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018

Animat Navigation Using Landmarks

Navigation of Simulated Animals Inspired by Bees

MATHIAS CARLSSON

Department of Computer Science and Technology
Department of Computer Science and Engineering
Chalmers University of Technology
Gothenburg, Sweden 2018

Animat Navigation Using Landmarks
Navigation of Simulated Animals Inspired by Bees
MATHIAS CARLSSON

Supervisor: Claes Strannegård, Department of Computer Science and Engineering
Examiner: Christos Dimitrakakis, Department of Computer Science and Engineering

Master's Thesis report 2018:
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2018

Animat Navigation Using Landmark
Navigation of Simulated Animals Inspired by Bees
MATHIAS CARLSSON
Department of Computer Science and Technology
Chalmers University of Technology

Abstract

Machine Learning techniques have made large advancements in previously challenging problems. A common problem with these methods is that they are very specialised on their specific field problem and it requires a lot of work to apply the methods on a new problem. A solution to this general issue is Artificial General Intelligence(AGI). This is an AI that is able to identify any problem and find a solution.

Animals often needs to navigate complex environments to survive. This master thesis tries to implement a homing navigation model, used by bees to find food by remembering places relative positions to landmarks, in the general animat model.

The model consists of using sensors that detect landmarks relative position to the animat. The model shows a small improvement when compared against Q-learning. The result suggests that the animat model can produce more sophisticated methods of navigation but further research needs to be conducted to explore its limits.

Keywords: animal, navigation, general intelligence, artificial intelligence, reinforcement learning.

Acknowledgements

I want to thank Fredrik Mäkeläinen and Hampus Torén for letting me use their code as a base for my implementation saving me a lot of time and work. I also want to thank my family for supporting me, especially when things got rough. I also want to thank my supervisor for all the support during the project. I also want to give a final thanks to my examiner.

Mathias Carlsson, Gothenburg, June 2018

Contents

List of Figures **xi**

1 Introduction **1**
1.1 Related work . . . 1
1.2 Research aim and goal . . . 2

2 Theory **3**
2.1 Q-learning . . . 3
2.2 Animat Framework . . . 4
2.2.1 Reward matrix . . . 6
2.3 Snapshot model . . . 6

3 Methods **9**
3.1 Computer implementation . . . 9
3.1.1 Implemented need . . . 9
3.1.2 Direction navigation . . . 9
3.1.3 Reward shaping . . . 10
3.2 Testing . . . 11
3.2.1 First test - 3 landmark test . . . 11
3.2.1.1 Test with basic functionalists . . . 11
3.2.1.2 Test with reward shaping . . . 12
3.2.1.3 Test with stronger smell constant . . . 12
3.2.2 9 Landmark test . . . 14
3.2.3 1 Landmark test . . . 14
3.2.4 Reward shaping test with no landmarks . . . 14
3.2.5 Position sensor test . . . 15

4 Results **17**
4.1 3 Landmark test . . . 17
4.2 Test with reward shaping . . . 21
4.3 Test with high shaping reward . . . 24
4.4 9 Landmark test . . . 25
4.5 1 Landmark test . . . 27
4.6 Reward shaping test with no landmarks . . . 29
4.7 Position sensor test . . . 31

5 Discussion **35**

5.1 Test specific discussion 35
5.1.1 3 landmark test 35
5.1.1.1 Test with reward shaping 35
5.1.1.2 Test with high reward shaping 36
5.1.2 9 landmark test 36
5.1.3 1 landmark test 36
5.1.4 Reward shaping test with no landmarks 37
5.1.5 Position sensor test 37
5.2 General discussion 37
5.2.1 Weakness of the model and result 38
5.3 Future work 38
5.4 Conclusion 39

Bibliography **41**

List of Figures

2.1 The different types of nodes that is used in the network in the artificial animat. 5
2.2 Image displaying the snapshot model from B. A. Cartwright and T. S. Collett. [1]. 7

3.1 Figure showing how the animats different sensors classify landmark directions. Green colours are N, W, E and S. Red Colours are NE, NW, SE and SW. 10
3.2 Figure showing a 10 times 10 test world with three, nine and one identical landmarks(gray circles) positioned around the food(flower). The goal of the animat(bee) is to reach the food and eat at that position. 13

4.1 Result of 3 Landmark test without reward shaping. 18
4.2 Result of 3 Landmark test with Q-learning and without reward shaping. 19
4.3 A single simulation result of the 3 Landmark test with no reward shaping where the animat was forced to eat the first three times it encountered food. 20
4.4 Simulation result of the 3 Landmark test with no reward shaping where the animat learned to navigate without any forced actions. . . 21
4.5 Result of 3 Landmark test with reward shaping. 22
4.6 Result of 3 Landmark test with reward shaping and Q-learning. . . . 23
4.7 Result of 3 Landmark test with high reward shaping. 24
4.8 Result of 3 Landmark test with high reward shaping and Q-learning. 25
4.9 Result of 9 Landmark test. 26
4.10 Result of 9 Landmark test with Q-learning. 27
4.11 Result of 1 Landmark test. 28
4.12 Result of 1 Landmark test with Q-learning. 29
4.13 Result of 0 Landmark test. 30
4.14 Result of 0 Landmark test with Q-learning. 31
4.15 Result of the position test. 32
4.16 Result of the position test with Q-learning. 33

1
Introduction

The ability to navigate is very important for any animal. A good navigation lets the animal find places where it can find food or find a place to take shelter from dangers. This is supported by the fact that all animals can navigate to a high extent.

In recent years Artificial Intelligence (AI) enhanced with Machine Learning techniques have made large advancements in previously challenging problems[2]. A common problem with these methods are that they are very specialised on their specific field problem and it requires a lot of work to apply these methods on a new problems[3]. A solution to this general issue is Artificial General Intelligence(AGI). This is an AI that is able to identify any problem and find a solution with a similar general method that animals and humans solve problems they encounter. AGI also uses this general method to solve many diverse problems.

This master thesis expands on an already developed method by exploring its capabilities of more advanced navigation by implementing an observed navigation behaviour of how bees uses landmarks in its environment to remember an advantageous place as well as finding its way back to it.

1.1 Related work

The Generic Animat research[4] project aims to create an artificial animal that tries to survive in a limited environment by trying to maintain its vital needs. The decisions it makes is learned over its lifespan using a combination of dynamic graphs and reinforcement learning. The decision process is done by either preforming the best action according to its own experience (exploitation) or preforming a random action (exploration) with a weighted probability.

One of the most important abilities that animals need to survive is the ability to navigate its environment. Animals need to navigate to places where they can find food or water, avoid dangerous areas and find safe places to rest. These are examples of why navigation are an important tool for any animal. An experiment of primitive navigation can be done with Braitenberg vehicles[5]. These vehicles are simple robots that move either towards or away from a light source and the vehicles behaviour emerges to either stay in dark areas or in bright areas depending how they are configured.

A method of navigation uses clear structures in the environment, referred to as landmarks, to remember how to find a specific location. The method is called Snapshot Navigation and is described in the paper "Landmark Learning in Bees"[1]. This is one of the strategies that bees uses to remember and find a source of food. The bees do this by remembering the angle and sizes of a number of landmarks viewed from the position of the food. The bees then navigate back to that location by minimising the difference of the viewed image to the remembered image of the landmarks, not unlike how Braitenberg vehicles operate. This simple navigation emerges into a homing navigation.

Recent research have shown evidence that the brain uses place cells that emit signals when it remembers places[6]. These cells are divided over a grid pattern such that at least one cell is always firing. With this lattice of place cells the rodent can effectively navigate in an explored environment and quickly reroute if it encounters new unexpected obstacles. This behaviour has been successfully implemented by a team from Deep mind[7]. This team has also successfully implemented a model of grid cells using a recurrent neural network emulating brain cells. This model was able to perform vector based navigation when trained with deep reinforcement learning. The model was performing on a super human level.

1.2 Research aim and goal

The snapshot navigation model is a simple navigation method that bees use to locate food [1]. The Generic Animat research is aiming to create artificial General intelligence by simulating basic animal behaviour [4].

The aim of this thesis is to implement the homing navigation in the animat framework to improve the animats navigation abilities and show by example that the framework can produce more sophisticated methods of navigation. More specifically the project will extend the current the animat framework so it can learn snapshot navigation while minimising the use of ad hoc solutions.

The algorithm will be compared with a Q-learning algorithm by testing both algorithms on a set of test cases which will be defined later in the report.

The project will answer the research question:
Will the algorithm inspired by the homing navigation preform better than the q-learning algorithm with respect to the agents well-being and effectiveness?

2
Theory

This chapter explains different models, data structures and algorithms that are used in the thesis.

2.1 Q-learning

Q-learning is a form of model-free reinforcement learning that lets agents find an optimal action strategy in controlled Markovian domains [8]. The agent tries actions in different states and evaluate the consequences of the actions, both immediate and long term, to determine what actions are good or bad. The main benefit of Q-learning is that the agent does not need to have a representation of the world and only needs a set of possible states and set of possible actions that can be performed in that state.

Each state-action combination has a Q-value $Q(x, a)$ that updates as the agent selects actions where x is the state and a is an action. The agent gets rewarded for each action depending on how good that action was with the value $\Re_x(a)$. A policy π is the agents strategy that decides what action to do in each state. The policy can be a simple set of rules or something else more complex. The Q-Value with a certain policy π is defined as:

$$Q^\pi(x, a) = \Re_x(a) + \gamma \sum_y P_{xy}[\pi(x)]V^\pi(y)$$

$\pi(x)$ is the chosen action of the policy given state x. V is the discounted expected reward following a policy π where γ $(0 < \gamma < 1)$ is the decay factor and y is the resulting state performing action a in state x:

$$V^\pi \equiv \Re_x(\pi(x)) + \gamma \sum_y P_{xy}[\pi(x)]V^\pi(y)$$

$P_{xy}[\pi(x)]$ is the probability of reaching the state y from state x given the policy π.

The Q-values is updated with the formula:

$$Q_n(x, a) = (1 - \alpha_n)Q_{n-1}(x, a) + \alpha_n(r_n + \gamma V_{n-1}(y)) \texttt{ If } x = x_n \texttt{ and } a = a_n$$

$$Q_n(x, a) = Q_{n-1}(x, a) \texttt{ Otherwise}$$

$$\texttt{Where } V_n(y) = max_b Q_{n-1}(y, b)$$

x_n is the current state in iteration n, a_n is the selected action, y_n is the resulting action of the action a_n, r_n is the immediate payoff. b is the different possible action from state y.

The function $V_n(y)$ gives the discounted reward of the state y, which is the expected reward of selecting the best action in every following states.

Each time step the agent either tries to explore or exploit. The agent tries to maximise its future profit by selecting the action with the highest Q-value in each state when exploiting and when exploring the agent chooses an available action at random. The ratio of exploring is determined by a parameter with a value between $[0, 1]$ and is lowered when the agent has a better distribution of the Q-values.

2.2 Animat Framework

The framework that the animat is based on is built up by sensor nodes and motor nodes. These nodes are connected by a network of edges, either by direct edges from sensor to motor or trough other nodes. These nodes combine sensor node signals and relays new signals in the network. A representation of all the nodes are shown in figure 2.1. This framework is further described in [9].

A sensor is defined as something the animat can perceive in its current environment. Each sensor node represents a sensor the animat has access to. For example if the current position of the animat is blue the animats sensor for blue is active. The sensor is binary, it is either active or inactive at any point, and a set of sensors is defined when the animat is created.

Each animat has a number of actions it can perform at every iteration. Each action has a motor node connected to it. If at any point a motor node receives a signal form the network the animat preforms the action that is associated to it.

The animats choose strategy depending on its most urgent need. Each of the animats needs are defined as a value [0,1] and the most urgent need decides the animats current strategy. The different actions are compared with each other by comparing different reward values and expected reward values that are stored in a reward matrix and is continuously updated during of the simulation.

To avoid getting stuck in a policy that leads to a bad behaviour by greedily only selecting the best action the animat has a small probability ϵ to explore other actions through random exploration. This probability is determined at the start of the simulation.

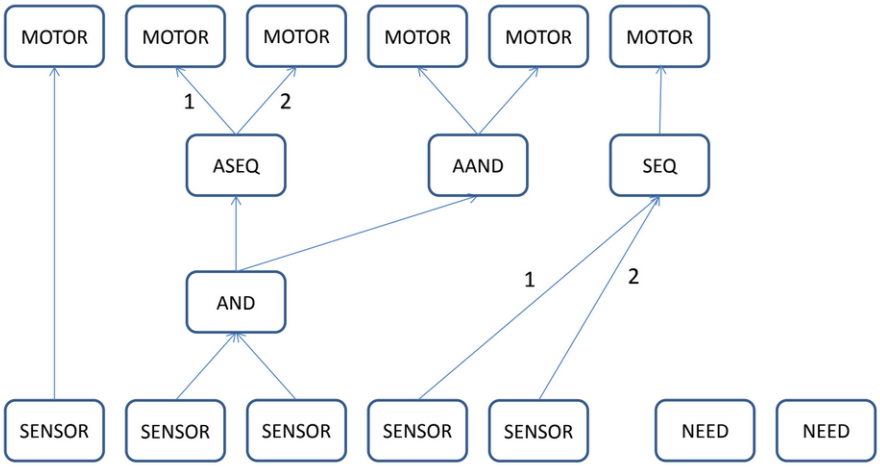

Figure 2.1: The different types of nodes that is used in the network in the artificial animat.

And nodes [AND] takes in two signals and becomes active when both input nodes are active at the same time. The And nodes simplifies the interpretation of sets of active sensors by combining sensor inputs that are common to be active at the same time.

Sequence nodes [SEQ] take in two signals and produce an output if both inputs are active in an ordered sequence in two following time steps. The difference from the AND node is that the signals are registered at different times and enables the animat to take in account changes in its environment.

And-Sequence nodes [ASEQ] is connected to two motor nodes and activates both in an ordered sequence if it receives a signal. Similar to the SEQ node this node activates other action nodes in sequence over two time steps.

Action-And nodes [AAND] is connected to two motor nodes and activates both when it receives a signal.

Both And-nodes and Sequence-nodes are sensor nodes but they can only be created during run time finds a common pattern or it finds it beneficially to use them.

It is a similar case for the And-Sequence and Action-And nodes with the difference that they are action node

When the animat experiences a reward value that differentiates with a sufficiently large amount from what it expected the animat get a so called "surprised reaction". When it gets a surprised reaction it triggers larger modifications to the network such as adding new sensor nodes to differentiate the current outcome from how the expected outcome expected from the reward matrix.

2.2.1 Reward matrix

The reward function is represented as a matrix where every row represent an action and each column is a possible state of the animats active sensors. Each cell is assigned a value and is updated when the action is performed in that state with the average reward given by the simulation from that action-state. After enough iterations the reward matrix will be representative of the reward given by the simulation.

The Q-matrix works in a similar way to the reward matrix with the difference that the value is updated with the Q-value function instead.

2.3 Snapshot model

Bees can use their environment to memorise and pinpoint a location where it have found food before. B. A. Cartwright and T. S. Collett. conducted experiments where a bee was trained to find food, in the form of sugar water, where the position was determined by a set of black cylinders placed in a specific orientation around it[1]. This configuration was placed in an otherwise uniformly white painted room. The bees was trained on a set of landmarks. The food source was then removed and landmark configuration was changed and the bees search pattern was observed.

What was discovered was that the bees uses the landmarks to find the location based on the viewed size of the cylinders and the angle between them and that it relies more on the angles then the sizes. In experiments with a single cylinder they use the relative size of the cylinder viewed from the position of the food to determine how close it spent most time searching for the food. It was assumed that the bee used something in the far distance to determine the orientation around the cylinder. With multiple cylinders the bee has a more robust ability to find the correct position even when one or more of the cylinders were gone.

A computer model was developed to test theories on how the bees navigate with respect to the landmarks. The final model uses a fixed ring of where the angles and sizes of the landmarks are represented as lines. The computer model then matches the angles to the edges of the viewed landmarks to the closest edge of the remembered landmarks and calculates the differences and sum them up to determine its next flying direction. A visual representation can be seen in fig2.2.

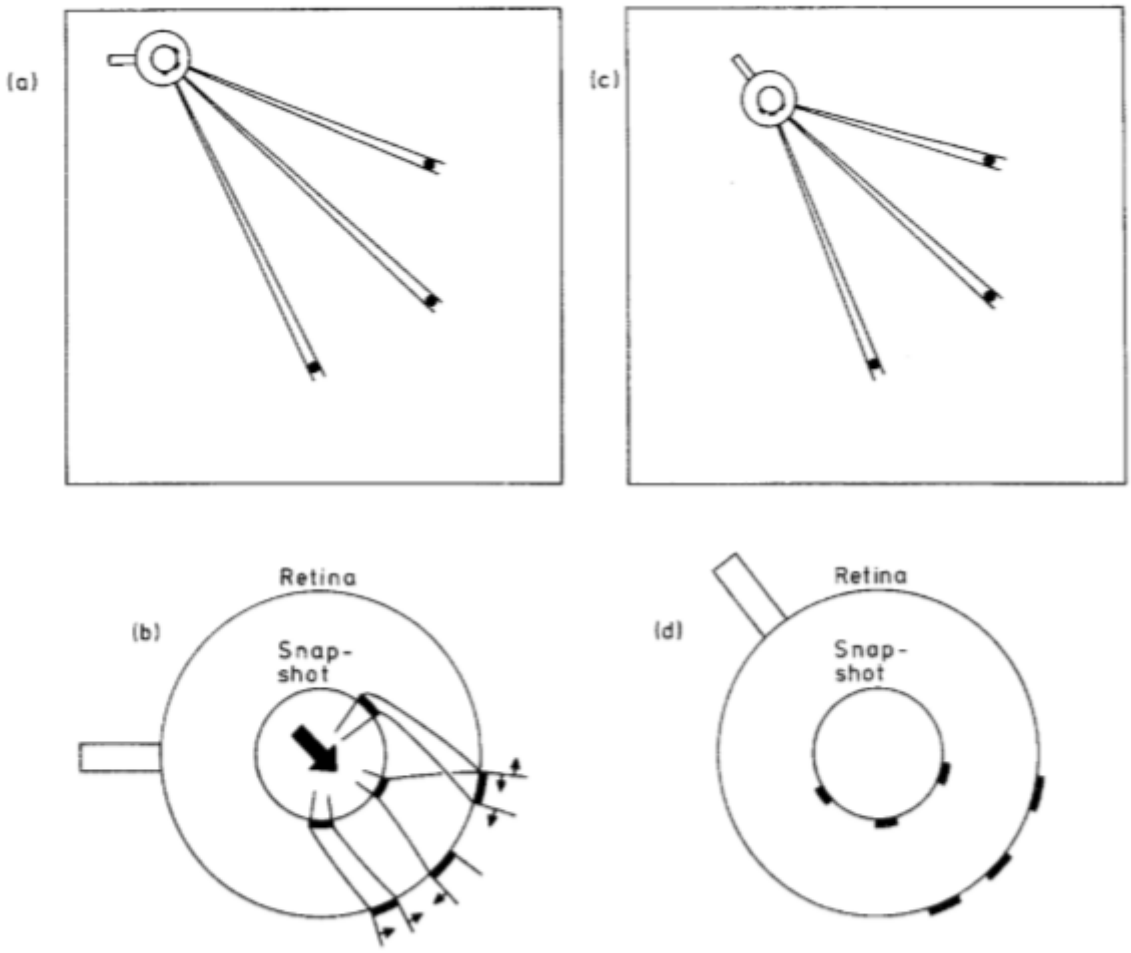

Figure 2.2: Image displaying the snapshot model from B. A. Cartwright and T. S. Collett. [1].

The snapshot model mimics how bees navigate to specific remembered position. The bees does this by remembering clear structures, referred to as landmarks, at the position it will be able to navigate back to. This is remembered as snapshot on its retina where the direction and size of landmarks is represented on a circle around the bee. The bee then compares its visual input to the remembered snapshot and determine its flying direction such that it minimises the discrepancies between the snapshot and its visual representations of the landmarks.

This model, when implemented correctly, mimics the way bees home in on a location of interest. This method is rather robust to changes in the landmark configuration as it can handle slight movements of the landmarks, adding or removing landmarks and other obstacles.

3
Methods

This chapter describes what was done to reach the goal of the project. It describes the different parts of the project work and gives a description of the problems encountered with a description of the solution for each problem. It also explains how the performance of the method is tested and assessed.

3.1 Computer implementation

The implementation is designed with 3 major parts [10].

- Main
- Agent(Animat)
- Environment

Environment owns the spatial positions of all the objects in the world including the animat. It also has a number of sensors and defined actions associated to the agent.

Agent owns and maintains the reward and Q-value matrix and decides on what action the animat is to choose.

Main sets up the environment and agent as well as controls the graphical user interface(GUI). The graphs of the simulation is a part of the GUI.

3.1.1 Implemented need

In all of the implemented simulation environments only one need was used. The need used was energy level which the animat increases by consuming food. This needs lower limit was removed to emphasise the problem of stable navigation by removing the artificial time limit that it introduced. The animats need can then be $[1, -\infty)$ and it strive for it to be as high as possible.

3.1.2 Direction navigation

A set of sensors that gives the animat good navigation using landmarks is to have a sensor that fires if a landmark is in a given direction relative to the animat. A landmark can be of different types that are for the animat considered to be the same object. These types are represented as different colors. It has 8 sensors for 8 different directions (N, NE, E, SE, S, SW, W, NW) for each type of landmark and

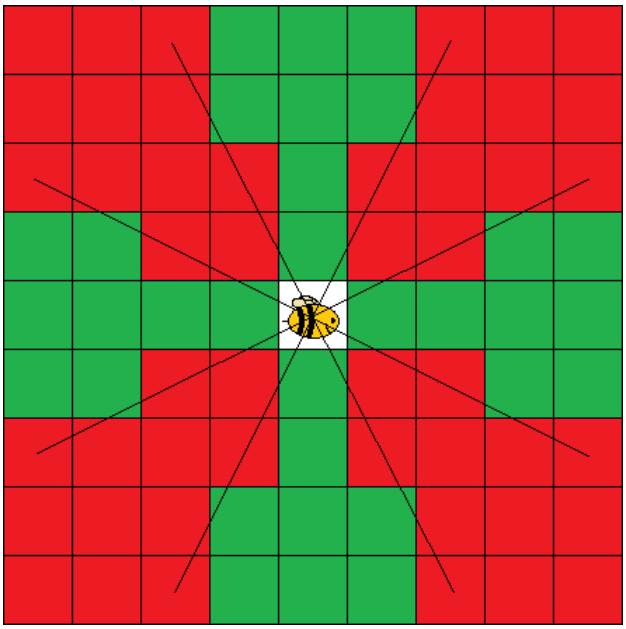

Figure 3.1: Figure showing how the animats different sensors classify landmark directions. Green colours are N, W, E and S. Red Colours are NE, NW, SE and SW.

is calculated in which direction the landmark is located, see fig 3.1. The animat also has a sensor that is active if it is on a position where there is food giving the animat the tools to recognise food when it is on the same position.

3.1.3 Reward shaping

The animat has difficulties to find the food without any initial knowledge of the environment. Several methods was tested to solve this problem. The most successful of those methods was to emulate smell by implementing "reward shaping"[11]. The concept is to give the animat a small reward for making an action that gets it closer to the goal. In this case moving closer to the food.

The reward is based on smell which is perceived as the concentration of particles at the animats position. The food emits particles that spread according with the inverse square law, equation 3.1. The amount of particles at distance d is the intensity I of the source divided by the square distance from the source:

$$\frac{I}{d^2} * C_{constant} \tag{3.1}$$

The animat gets a reward depending on the change of density of particles d from one time step to another multiplied by a constant $C_{constant}$ which is used to control how much the shaping reward affects the animat. The total reward the animat receive each iteration R_i is calculated as the difference in the need value from the last iteration added to the difference of value of the reward shaping.

$$R_i = n_i - n_{i-1} + \left(\frac{I}{d_i^2} - \frac{I}{d_{i-1}^2}\right) C_{constant} \tag{3.2}$$

Here n_i is the energy level the animat has at iteration i. As the distance $d = 0$ produces a division by zero there are special cases to handle these situations. In these situations the distance is set to $d = 1$. The intensity is decided to be $I = 1$ for simplicity.

3.2 Testing

The performance of the animat will be tested with experiments where the animat is released in a world where the animat will learn to navigate in. The tests is designed to make it difficult for the animat to sustain a high level of its need and good navigation skills are required. Some tests are highly inspired by tests from B. A. Cartwright and T. S. Colletts paper[1] (3/9/1 Landmark test) and others are designed to analyse specific test cases (no landmark test and position test). Each test compares the performance animat with network updates(network animat) to the performance of the Q-learning.

The Q-learning is implemented by running the animat network without updating the network or creating new nodes. As the decision in each iteration is chosen by a Q-learning algorithm this is the same as Q-learning when all other network functionalities are removed.

Each test case was tested by running each case for 100 000 iterations. The animat have one need, energy, in each of the tests. Data points were sampled every 1 000 iteration, either as a sample that from iteration(energy level and network size) or as a collective sum of the over the last 1 000 iterations(food consumed).

3.2.1 First test - 3 landmark test

3.2.1.1 Test with basic functionalists

The animat is placed in a 10x10 world with one food source surrounded by three identical landmarks, see first image in fig 3.2. The animats goal is to keep its need as high as possible. The need decreases by 0.001 with each iteration. The need increases with 0.6 when it performs an eat action when in the same position as the food and decrease with 0.2 on any other position. Whenever the animat performs the eat action on the food position the animat is moved to a random position in the world excluding the food position. The animat will need to learn the relative positions of the landmarks and food to be able to thrive in this environment.

This test is also done two times without calculating any average over multiple runs to easily discern if the energy level stabilises or not. Once where the animat is forced to eat the three first times encounter the food and once without any forced

actions. This is to test how much the forced actions affect the learning of the animat.

Parameter	Value
Grid size:	10x10
Number of landmarks:	3
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	0
exploration rate (ϵ):	0.01

Sensor	Condition
FOOD:	Food is on the same tile as the animat
LANDMARK1:	A landmark is on the same tile at the animat
Dir-N LANDMARK1:	A landmark is north relative to the animat
Dir-NE LANDMARK1:	A landmark is north east relative to the animat
Dir-E LANDMARK1:	A landmark is east relative to the animat
Dir-SE LANDMARK1:	A landmark is south east relative to the animat
Dir-S LANDMARK1:	A landmark is south relative to the animat
Dir-SW LANDMARK1:	A landmark is south west relative to the animat
Dir-W LANDMARK1:	A landmark is west relative to the animat
Dir-NW LANDMARK1:	A landmark is north west relative to the animat

3.2.1.2 Test with reward shaping

This test uses the same settings as the previous test with the exception that reward shaping will be used. This test will also be sampled over multiple run to calculate an average.

Parameter	Value
Grid size:	10x10
Number of landmarks:	3
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	0.005
exploration rate (ϵ):	0.01

3.2.1.3 Test with stronger smell constant

This test attempts to minimise the time to reach stabilisation by increasing the shaping constant. The shaping constant is set to 1 which gives the animat a reward with similar significance as the reward for the animat to eat.

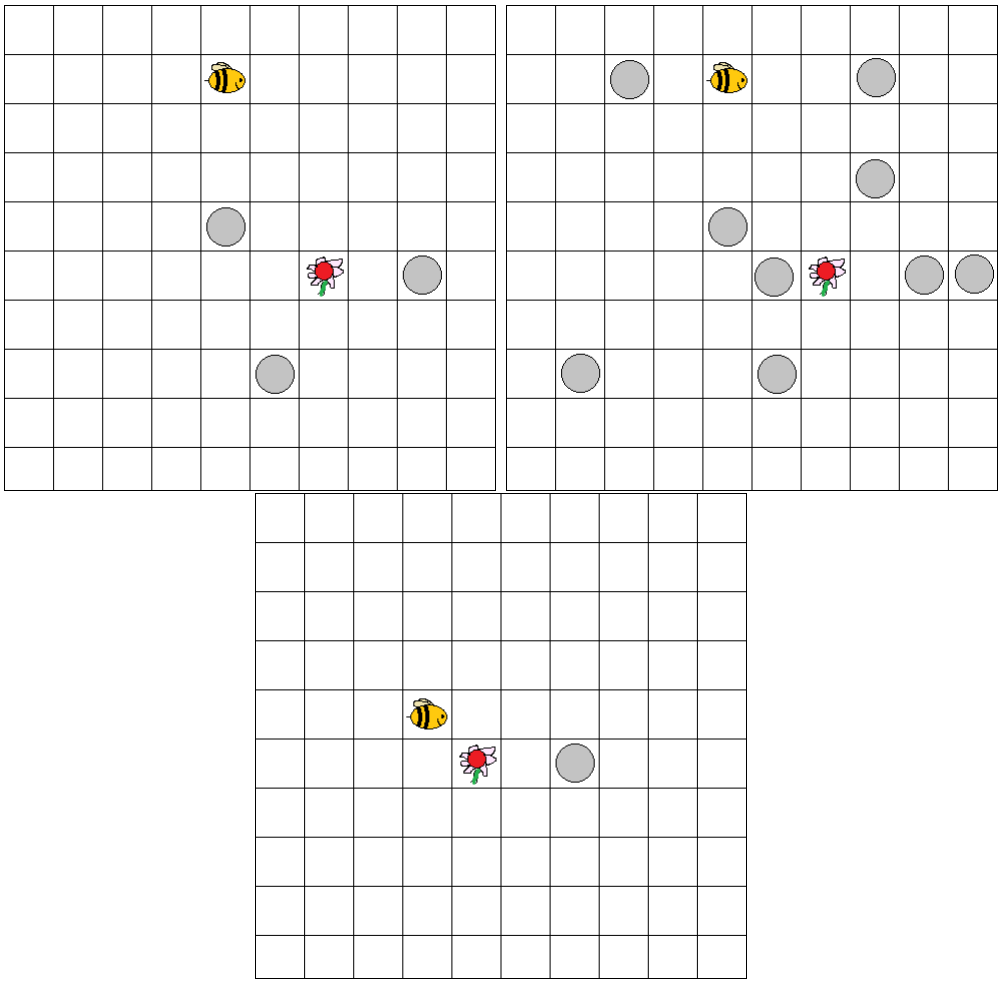

Figure 3.2: Figure showing a 10 times 10 test world with three, nine and one identical landmarks(gray circles) positioned around the food(flower). The goal of the animat(bee) is to reach the food and eat at that position.

Parameter	Value
Grid size:	10x10
Number of landmarks:	3
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	1
exploration rate (ϵ):	0.01

3.2.2 9 Landmark test

A test with the same sensors and parameters as the test with 3 landmarks but with 9 different landmarks positioned as the second image in fig 3.2

Parameter	Value
Grid size:	10x10
Number of landmarks:	9
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	1
exploration rate (ϵ):	0.01

3.2.3 1 Landmark test

A test with the same sensors and parameters as the test with 3 and 9 landmarks but with 1 different landmarks positioned as the third image in fig 3.2.

Parameter	Value
Grid size:	10x10
Number of landmarks:	1
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	1
exploration rate (ϵ):	0.01

3.2.4 Reward shaping test with no landmarks

Unlike the earlier defined tests this test has 3 sources of food, each with their own smell that is summed to the smell sensed for the animat. The goal of this test is to determine if the animat can survive navigating only with the sense of smell. Another change is that instead of the animat relocating after food consumption the food is moved to a random free tile.

Parameter	Value
Grid size:	10x10
Number of landmarks:	0
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	1
exploration rate (ϵ):	0.01

Sensor	Condition
Food sensor:	Food is on the same tile as the animat

3.2.5 Position sensor test

A test with no landmarks an a different set of sensors. The animat has a sensor for each possible row and column, in total 21 sensors. With this set of sensors the animat has information to determine its exact position.

Parameter	Value
Grid size:	10x10
Number of landmarks:	0
Food reward:	0.6
Not food penalty:	0.2
Passive energy decay:	0.001
Shaping constant:	1
exploration rate (ϵ):	0.01

Sensor	Condition
FOOD:	Food is on the same tile as the animat
x-position0:	The animat is on the x-coordinate 0
x-position1:	The animat is on the x-coordinate 1
x-position2:	The animat is on the x-coordinate 2
x-position3:	The animat is on the x-coordinate 3
x-position4:	The animat is on the x-coordinate 4
x-position5:	The animat is on the x-coordinate 5
x-position6:	The animat is on the x-coordinate 6
x-position7:	The animat is on the x-coordinate 7
x-position8:	The animat is on the x-coordinate 8
x-position9:	The animat is on the x-coordinate 9
y-position0:	The animat is on the y-coordinate 0
y-position1:	The animat is on the y-coordinate 1
y-position2:	The animat is on the y-coordinate 2
y-position3:	The animat is on the y-coordinate 3
y-position4:	The animat is on the y-coordinate 4
y-position5:	The animat is on the y-coordinate 5
y-position6:	The animat is on the y-coordinate 6
y-position7:	The animat is on the y-coordinate 7
y-position8:	The animat is on the y-coordinate 8
y-position9:	The animat is on the y-coordinate 9

4

Results

This chapter describes the result of the different tests defined in chapter 3. Figures of the network sizes is not shown in the result because of the networks are static.

Each of the tests have two figures. One for when creation of nodes is enabled that has three sub-graphs (A),(B) and (C). (A) has the sample of the animats need for energy every 1000 iteration. (B) samples the sum of times the animat has consumed food in the last 1000 iterations. (C) has the sample of the number of nodes in the network. The other figure has the result for the Q-learning. A number of sensors in the network does not change for this animat there is no network size graph in these figures. Each figure also have the variance of all the tests displayed.

4.1 3 Landmark test

Figure 4.1 shows the average result of the 3 Landmark test without any reward shaping calculated over 20 runs. Figure 4.1 A and B shows that the animat do not consume any food during the tests. although it does not consume any food the animat does create new nodes to its network.

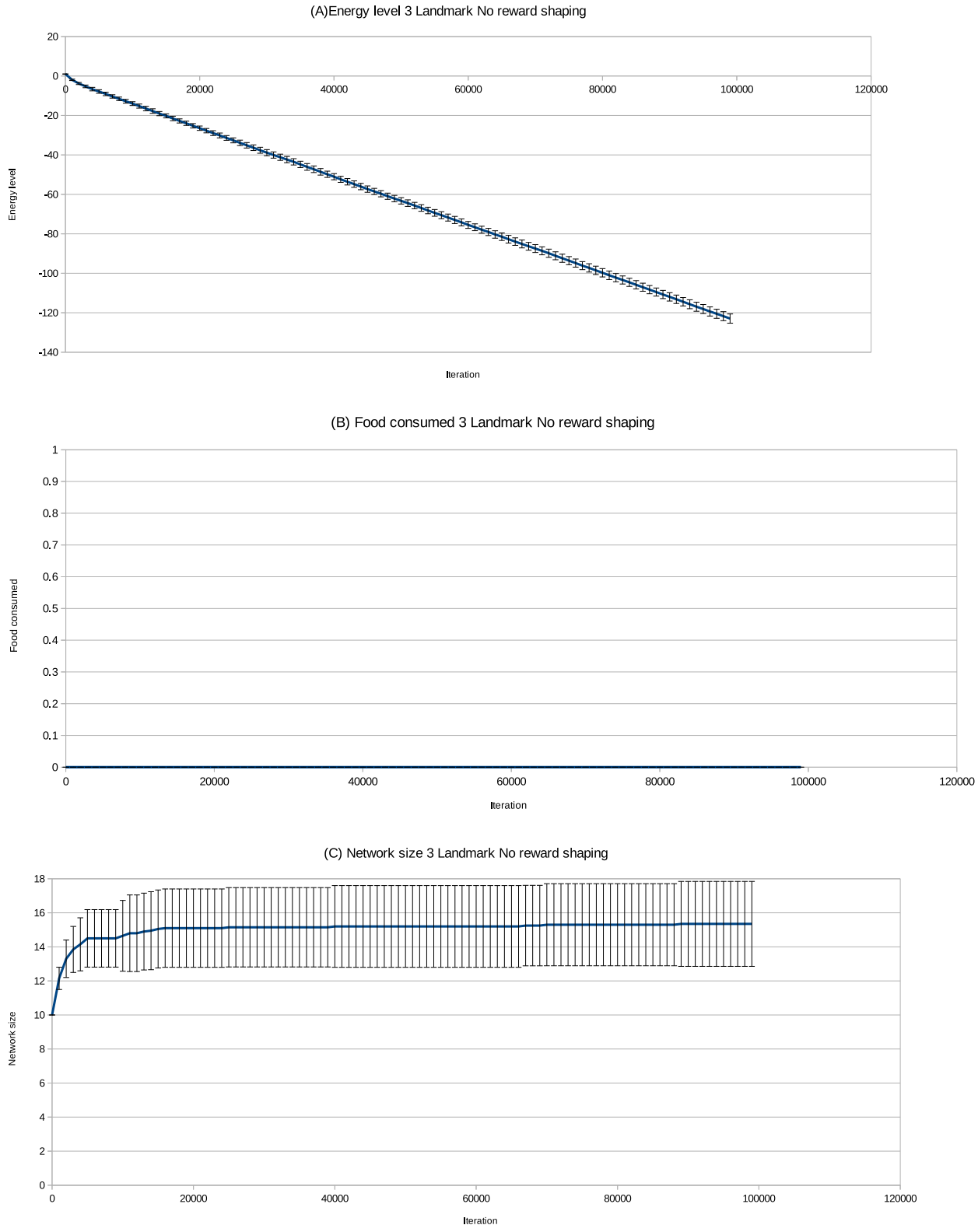

Figure 4.1: Result of 3 Landmark test without reward shaping.

Figure 4.2 shows the 3 Landmark test result with Q-learning with average values calculated over 20 runs. This result is similar to how the network animat performs

in figure 4.1 with the exception that at in least one of the tests runs the animat consumes the food once in an early iteration.

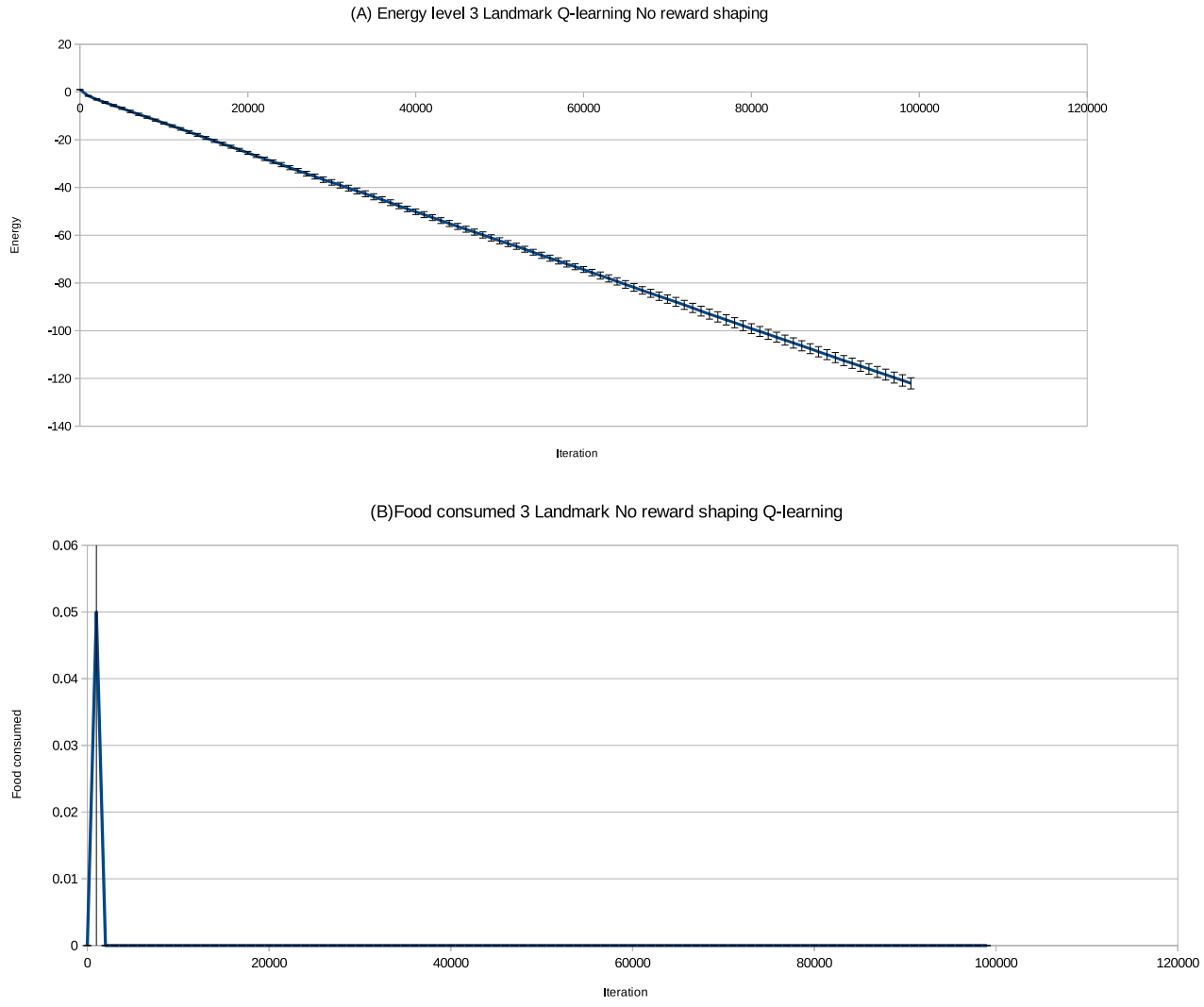

Figure 4.2: Result of 3 Landmark test with Q-learning and without reward shaping.

The results of the single run tests can be seen in figure 4.3 and 4.4. The animats energy level is displayed in the top graph and a local energy level id displayed in the bottom right graph. The bottom left graph displays the distribution of the different actions performed.

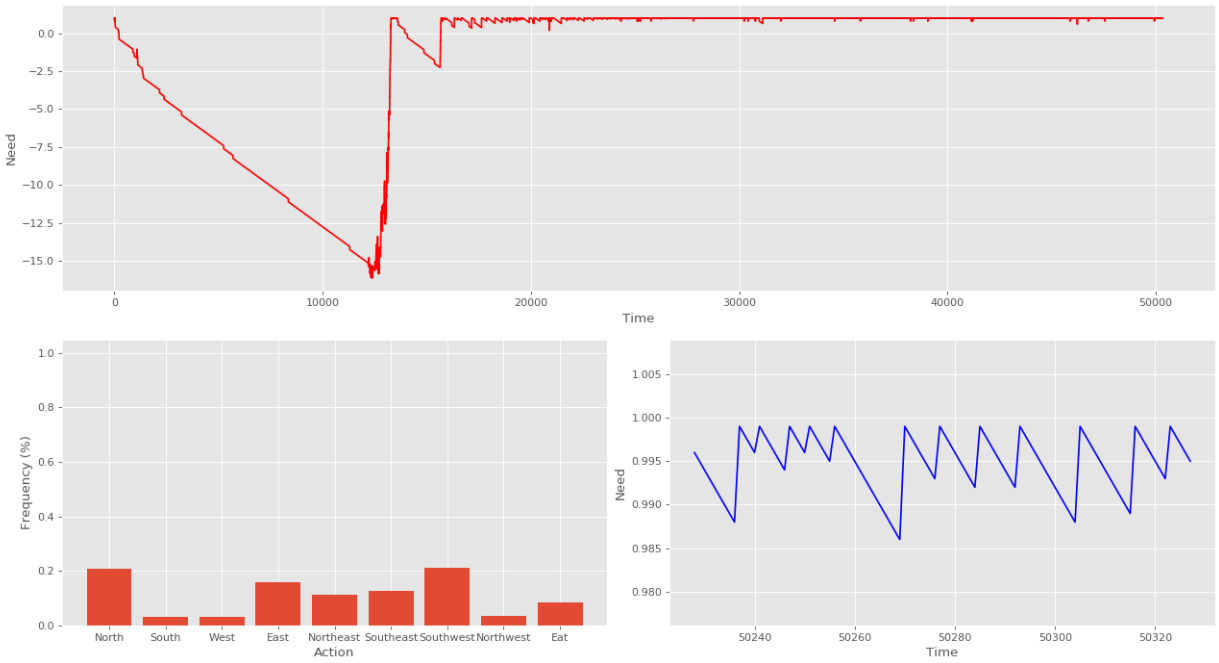

Figure 4.3: A single simulation result of the 3 Landmark test with no reward shaping where the animat was forced to eat the first three times it encountered food.

In both single run tests tests the animat learns to thrive in the test environment after a time of exploration. The animat takes a long time to learn to eat at the food. It usually needs to eat at the food 3-4 times before its need for food stops dropping. A period of high fluctuations follows where the animat learns how to navigate. Then it very quickly learns to find its way to the location of the food from any position in the world and keeps its need for food stable at 1.

In the first supervised test the animats need stabilised after 15 000 time steps. The three forced eat actions occurred at around time 5, 10 and 1800. The animat reached a semi stable state at around 14 000. There it had confused a position similar to that of the food to be a good place to eat. This confusion was solved 2 000 time steps later. After that it showed minor fluctuations for 6 000 time steps. At the end of the simulation the network had created 26 new And nodes. The frequency of the different actions are unevenly spread.

Figure 4.4: Simulation result of the 3 Landmark test with no reward shaping where the animat learned to navigate without any forced actions.

The second unsupervised test ran for 105 000 time steps4.4. The need stabilised at time 95 000 where it continued to be stable. The distribution of the actions are evenly spread except for Eat that is significantly lower. The network has 33 new And nodes at the end of the simulation. Other unsupervised tests looks very similar with the exception that they stabilize at different times, usually earlier.

4.2 Test with reward shaping

In this test, reward shaping was introduced in the algorithm. The result is the calculated average of 10 different runs that run for 100 000 iterations. The network animats average energy level, number of times it eats and number of nodes in its network is shown in figure 4.5 with variance. Each data point is a sample at every 1 000th iteration with the exception of number of times eating is the sum over the last 1 000 iterations.

The results shows that the animat stabilises around 10000 iterations even with some variation. The number of food consumed is stabilising around 135 every 1 000 iterations which gives a net energy of $135 * 0.6 - 0.001 * 1000 = 80$. The network stabilises on 37 nodes and the energy need stabilises with 27 nodes in the network.

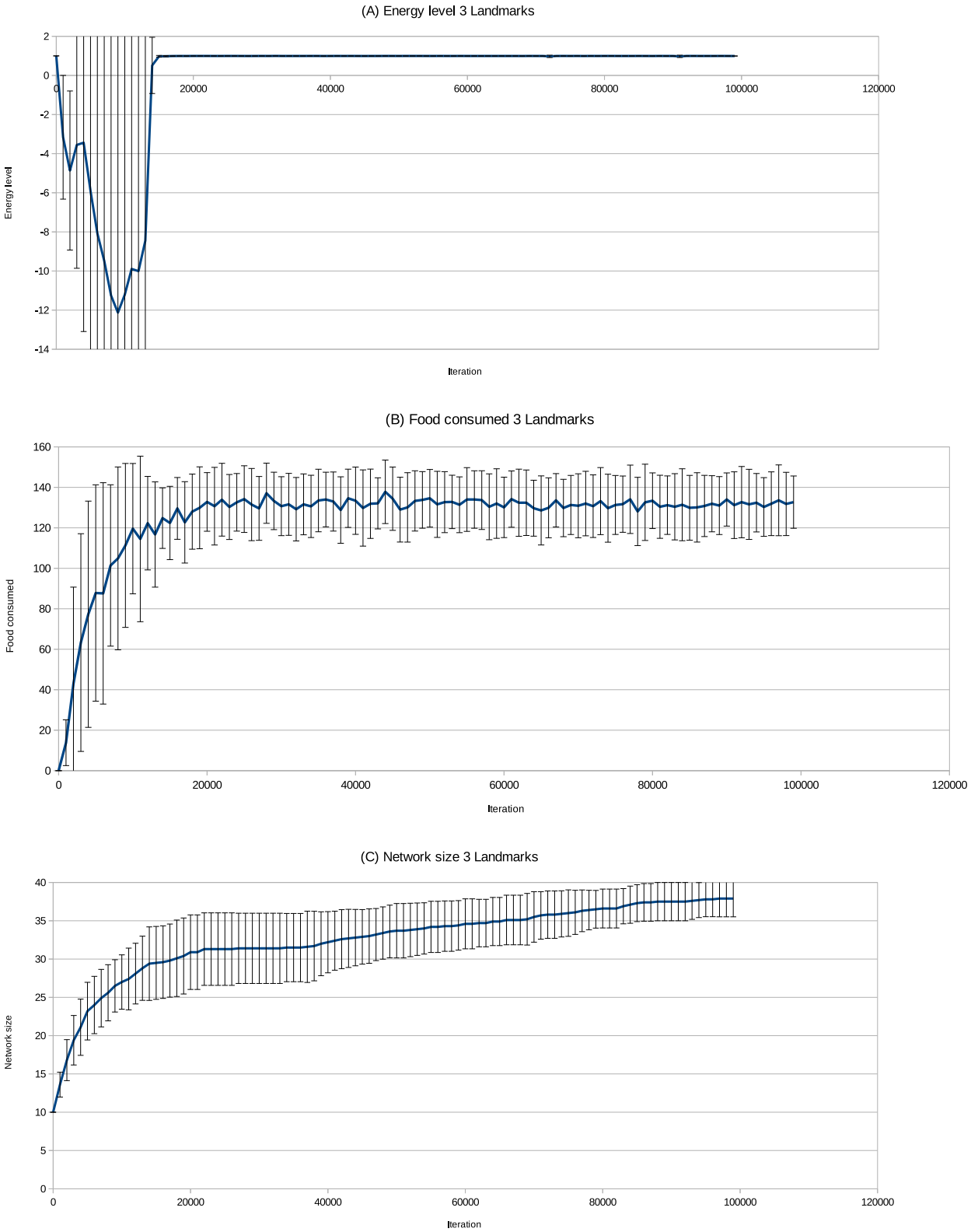

Figure 4.5: Result of 3 Landmark test with reward shaping.

Fig 4.6 shows the results of a similar test where the creation of new nodes was disabled. This is to compare the performance against Q-learning. A figure of the

network size is omitted as it is constant with 10 sensor nodes.

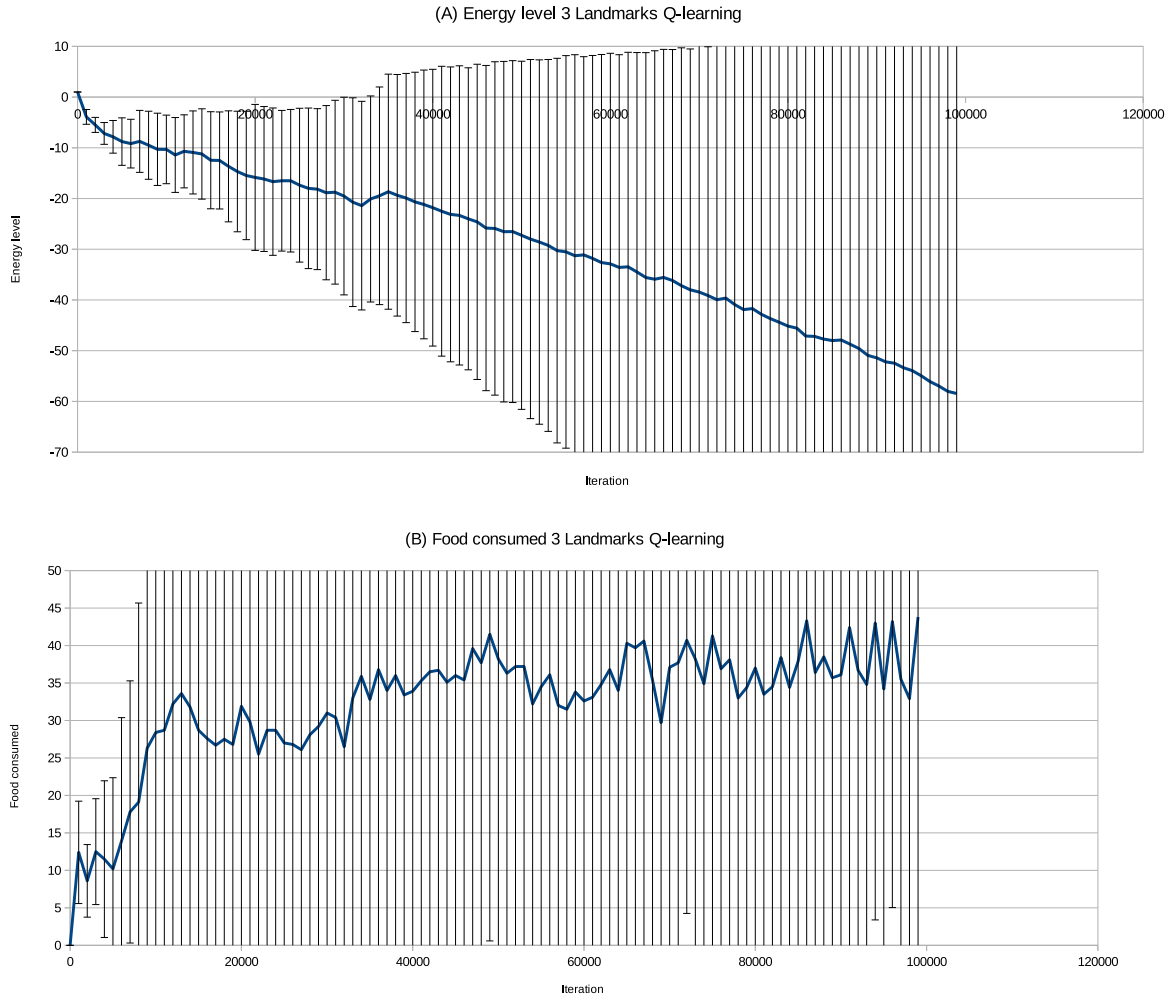

Figure 4.6: Result of 3 Landmark test with reward shaping and Q-learning.

The Q-learning animat test either succeeds to stabilise or continues to drop its energy level. The ratio between these two outcomes are unknown. This is due to the maximum energy level is 1 and has no lower level which lowers the mean result. The mean energy level is significantly lower than the animat with network nodes.

Another indicator pointing to the better performance of the network animat is the number of food consumed. The animat consumes on average 135 over 1000 iterations and Q-learning consumes around 35 on average. This is however including the runs where the Q-learning do not stabilise, lowering its average.

4.3 Test with high shaping reward

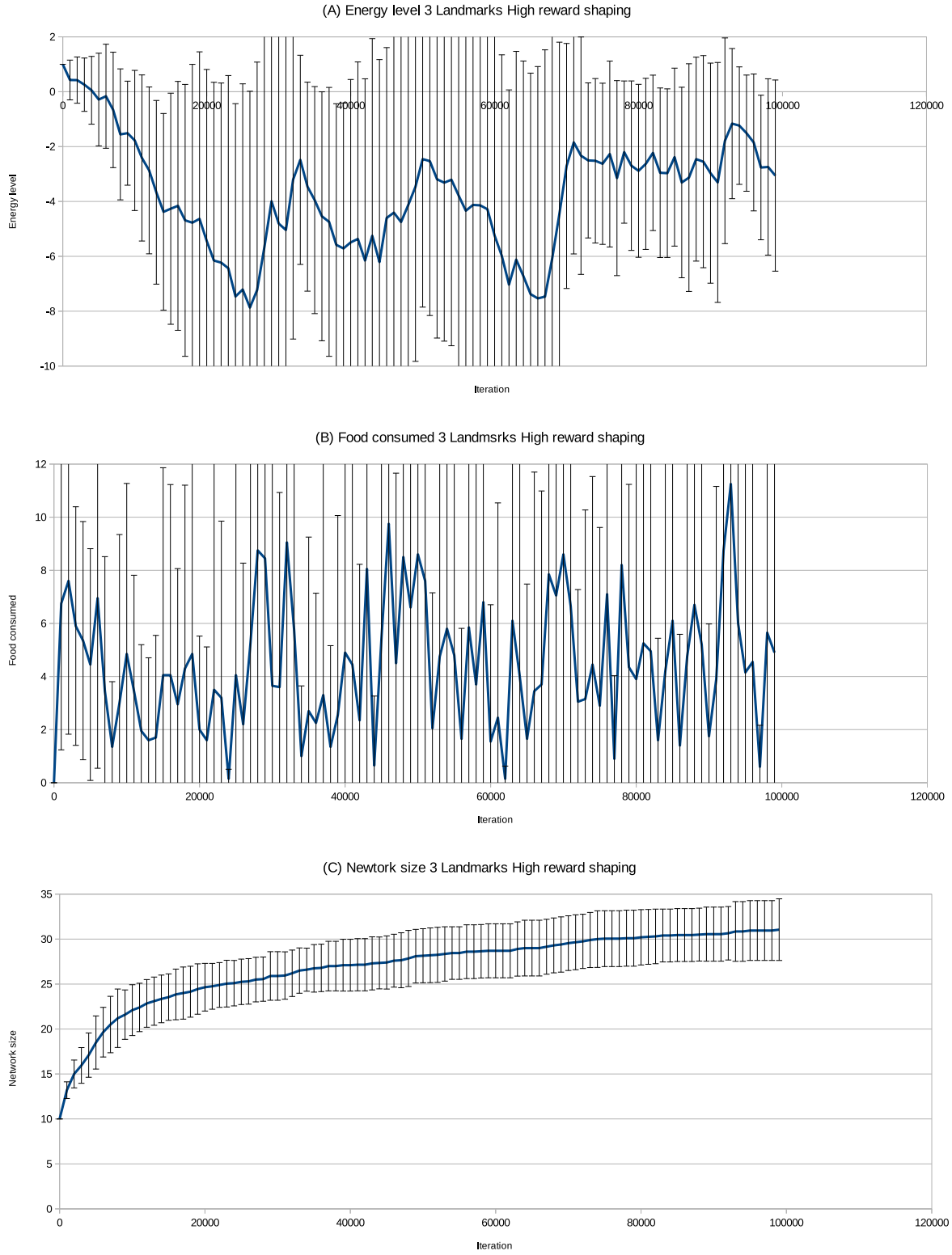

Figure 4.7: Result of 3 Landmark test with high reward shaping.

The network animats energy level is not stable but never drops below an average of -8 energy, see fig 4.7 A. The food consumption does not stabilise and the amount does not reach higher than 12 consumptions per 1000 iterations. The network size in figure 4.7 C is similar to the corresponding test with low shaping reward, fig 4.5 C.

The mean energy level in fig 4.8 A is continuously decreasing with some variation. The figure resembles fig 4.7 A until the 50 000th iteration where the Q-learning animat continue to drop.

The mean food consumption is stabilising around 4 consumptions each 1000 iteration at iteration 5000.

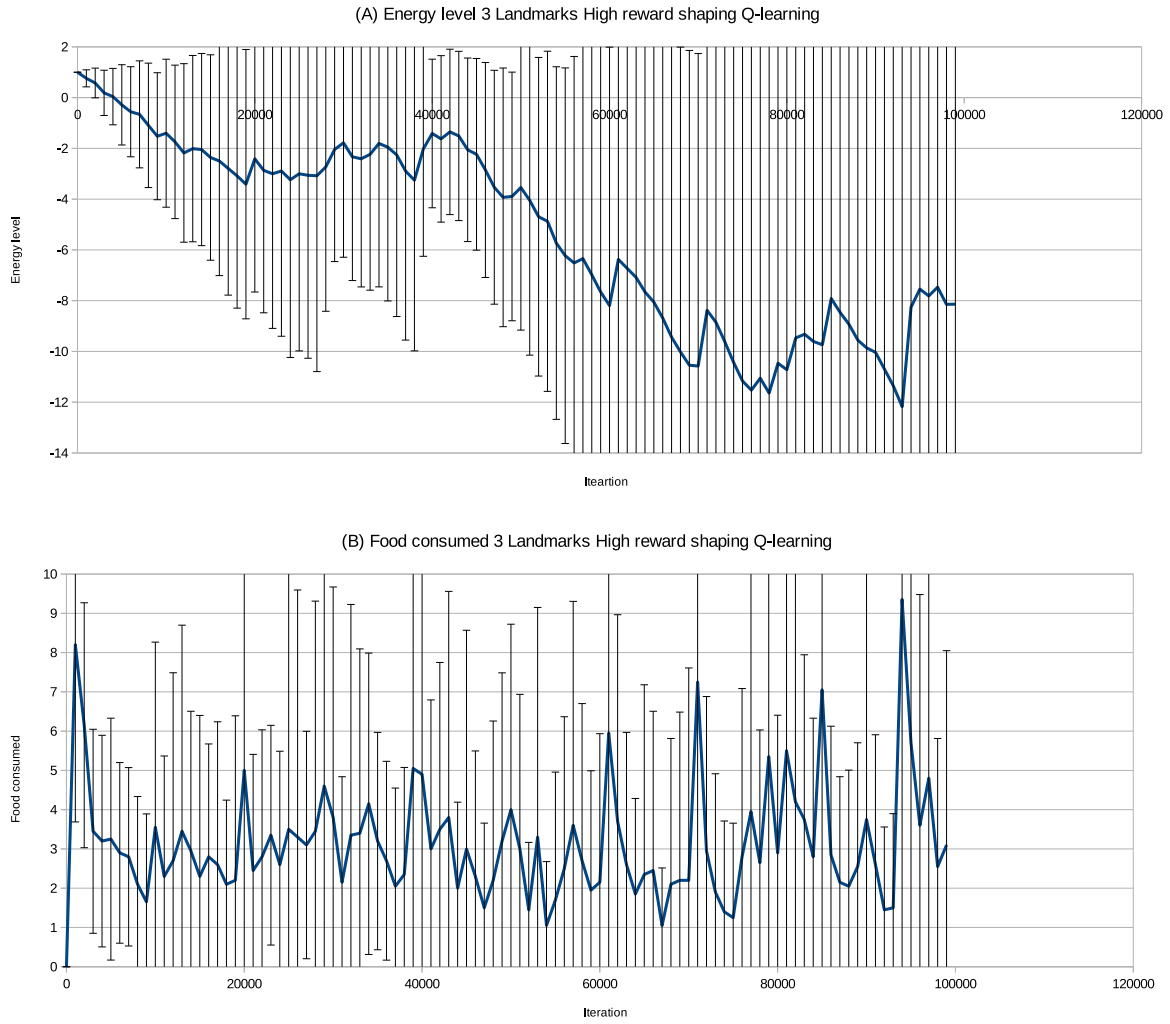

Figure 4.8: Result of 3 Landmark test with high reward shaping and Q-learning.

4.4 9 Landmark test

Figures 4.9 shows the average energy level, the average total number of food consumed between each sample and the average number of sensor nodes in the network

over 20 different runs with each running for 100 000 iterations. Figures 4.10 shows a similar simulation with the exception that the creation of new nodes is disabled.

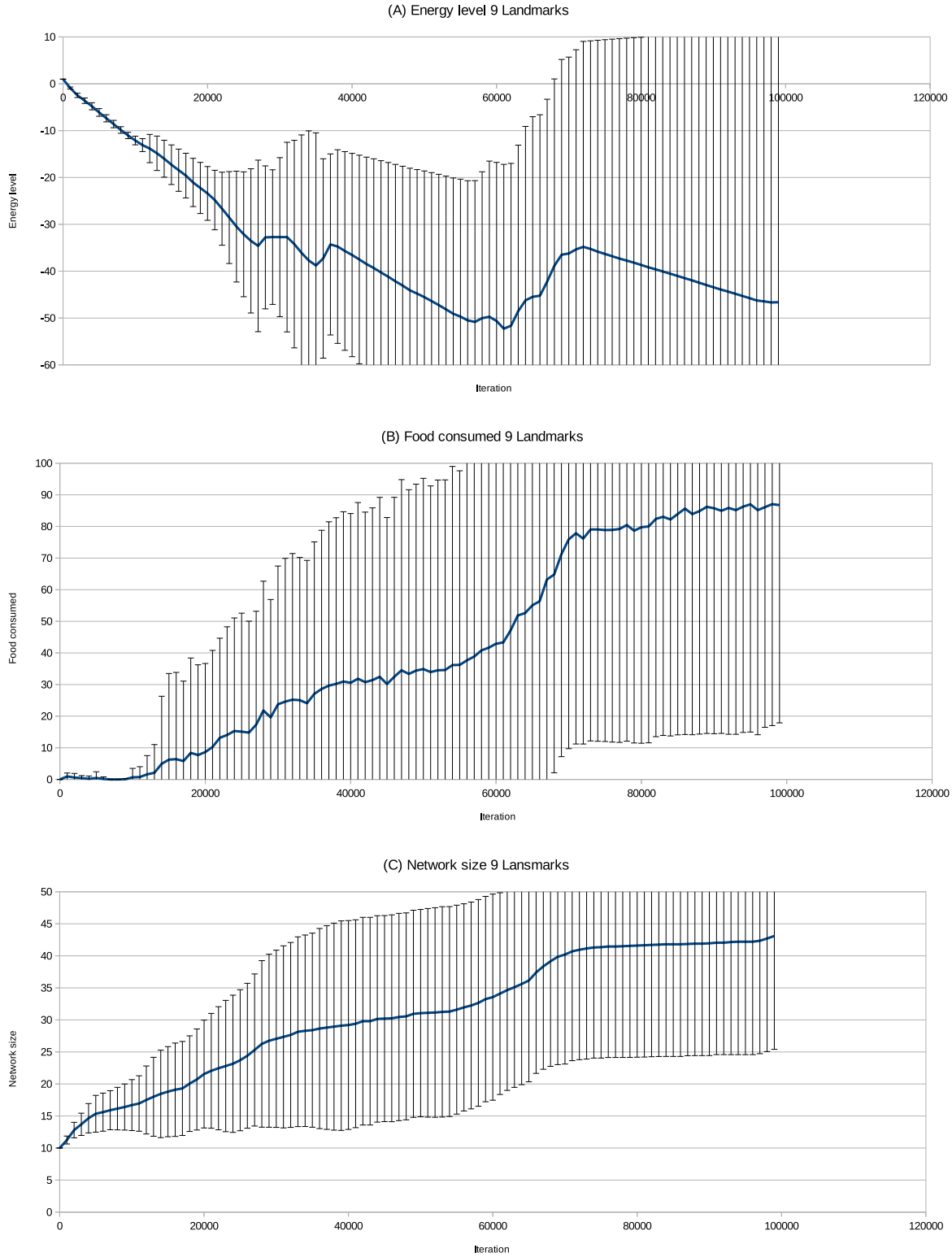

Figure 4.9: Result of 9 Landmark test.

In figure 4.9 A it is clear that the different simulations had significantly different results where some stabilised around 60 000 iterations and some did not stabilise at all lowering the total average and increasing the variance. Figure 4.9 B shows that the average number of nodes increases significantly around iteration 60 000, which is around the same time as some simulations stabilise. In figure 4.9 C the average number of sensors increase form 30 to 40 around the same time as some of the simulations stabilise.

Figure 4.10 A displays a consistently poor performance with low variance. Figure 4.10 B shows that it rarely consumes any food at all. The amount of ordered information is too much for the Q-learning to find a good strategy.

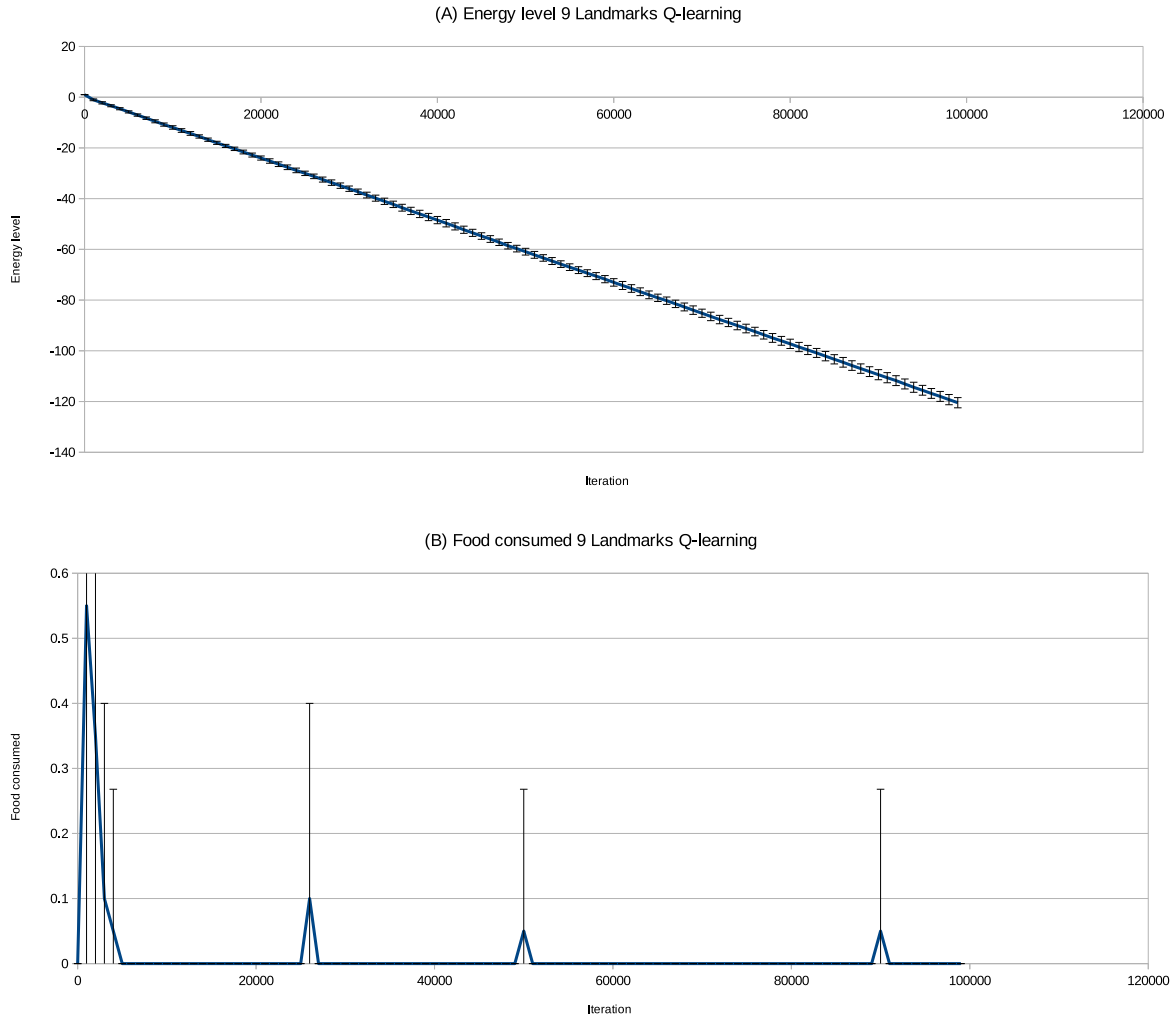

Figure 4.10: Result of 9 Landmark test with Q-learning.

4.5 1 Landmark test

Figure 4.11 shows the average energy level, the average total number of food consumed between each sample and the average number of sensor nodes in the network

over 20 different runs with each running for 100 000 iterations using the network animat. Figures 4.12 shows the same experiment for Q-learning.

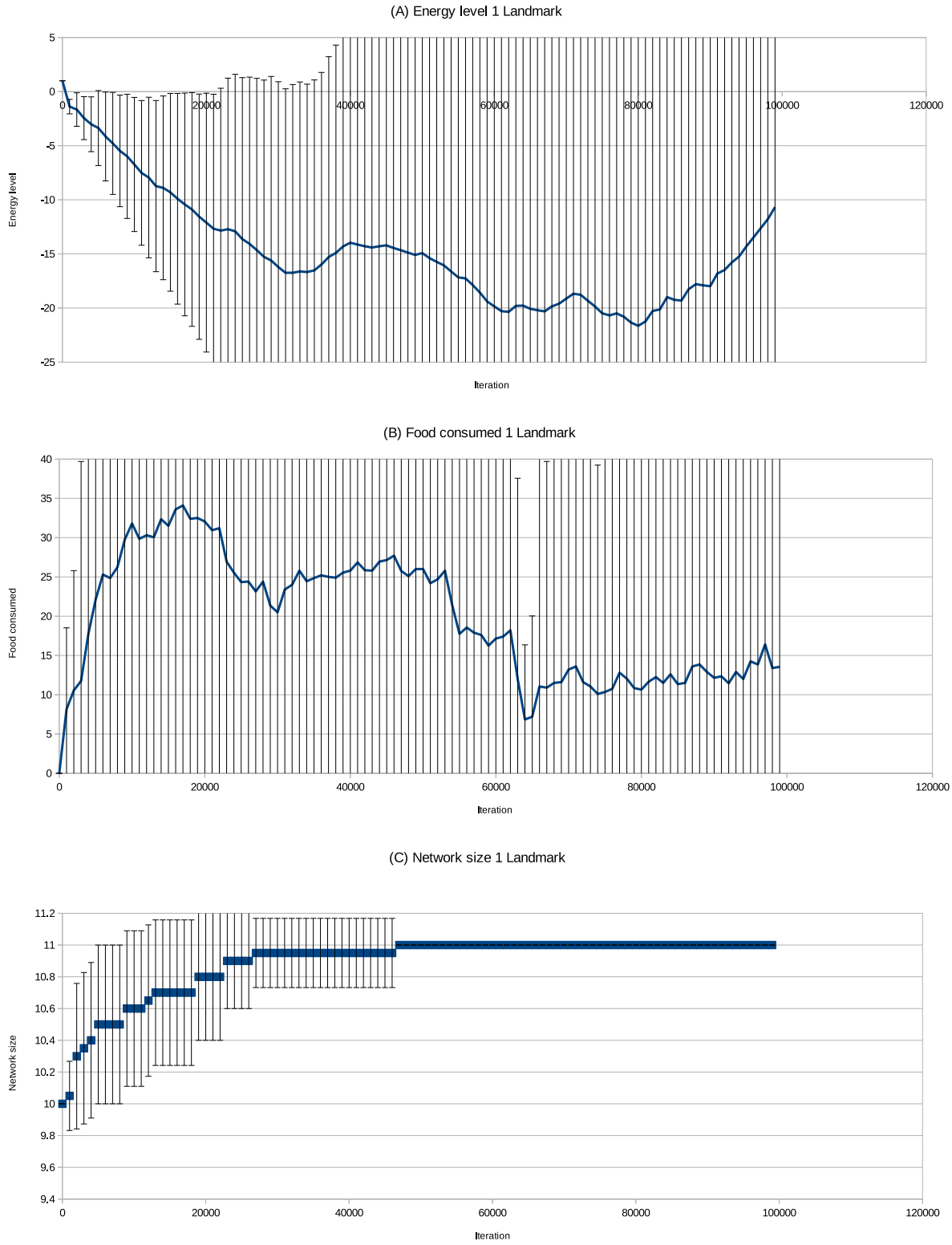

Figure 4.11: Result of 1 Landmark test.

Figure 4.11 A shows that the animat has trouble navigating to the food. It starts to have a continuous increase form iteration 80 000 and forward that suggests that the average will stabilise if it had run for a longer time. The variance is in general very high. The average number of consumed food is very unstable which is visible in figure 4.11 B. Th maximum number of nodes in the network is 11 and is reached in all simulations before the 25 000th iteration.
The simulation with Q-learning, seen in figure 4.12 shows that the animat could not navigate effectively to the food.

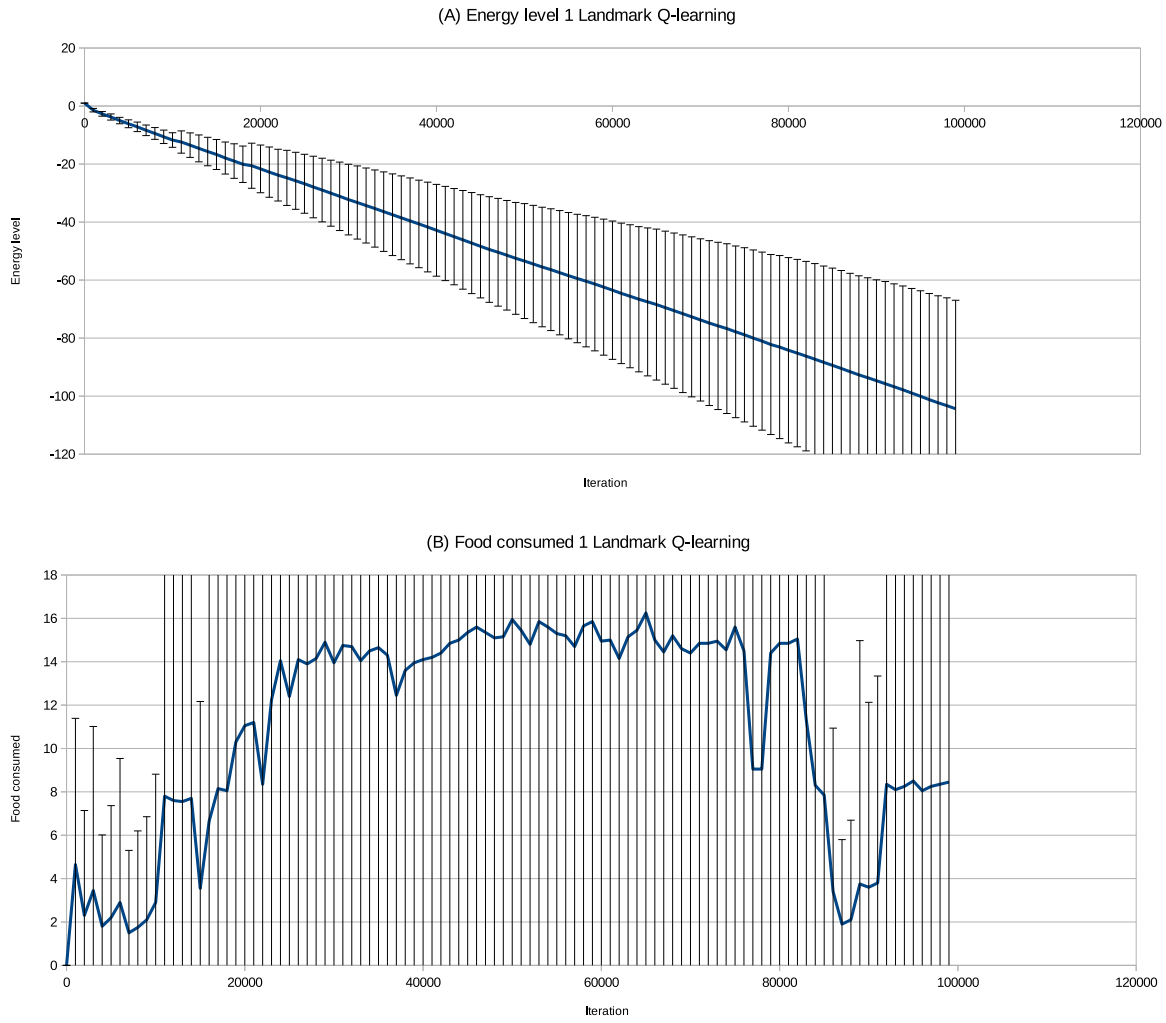

Figure 4.12: Result of 1 Landmark test with Q-learning.

4.6 Reward shaping test with no landmarks

The results of the experiment with only the reward shaping is shown in figures 4.13 and 4.14. There is no difference between the network animat and the Q-learner. This is to be expected as there is no new node that can be created.

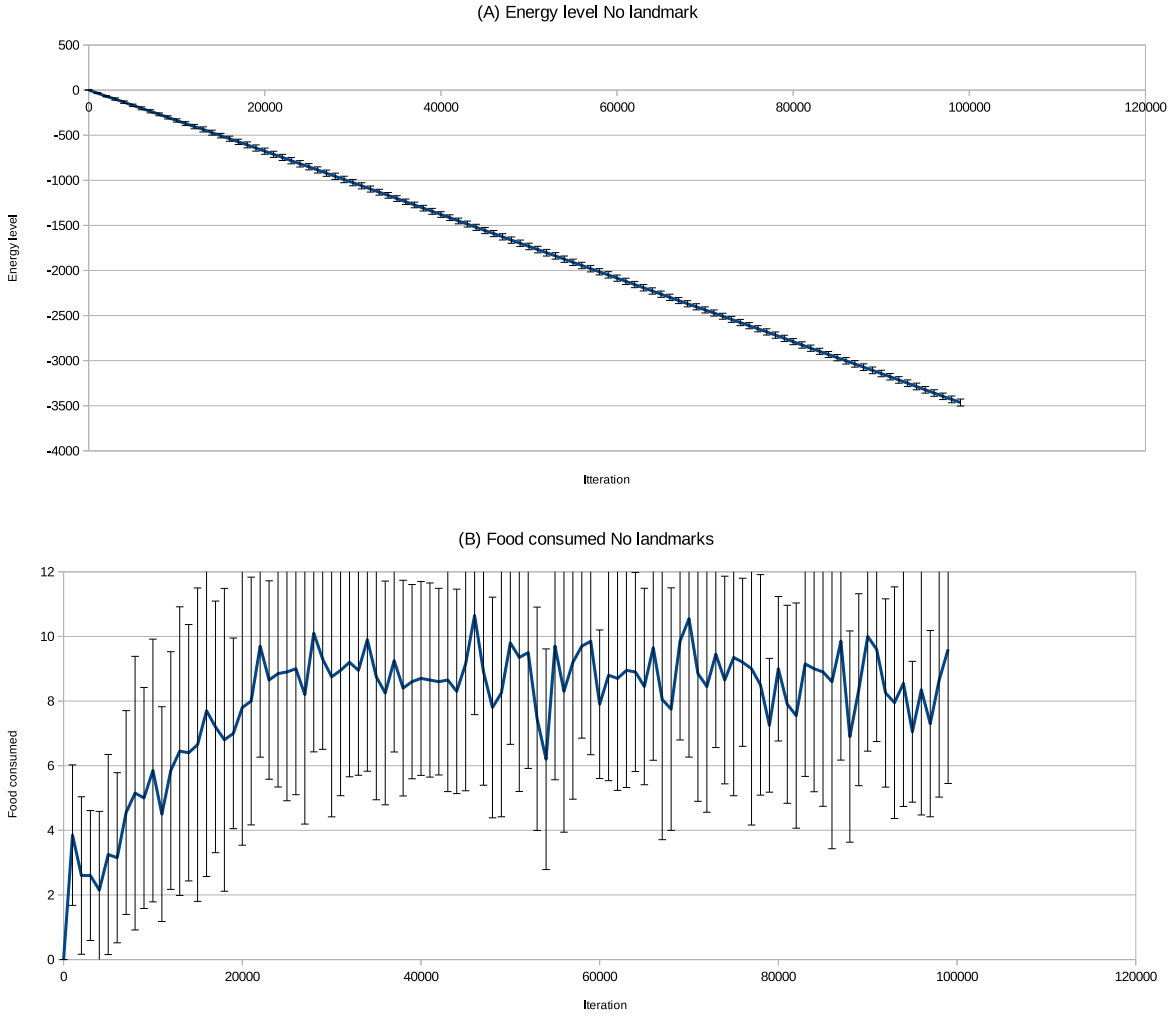

Figure 4.13: Result of 0 Landmark test.

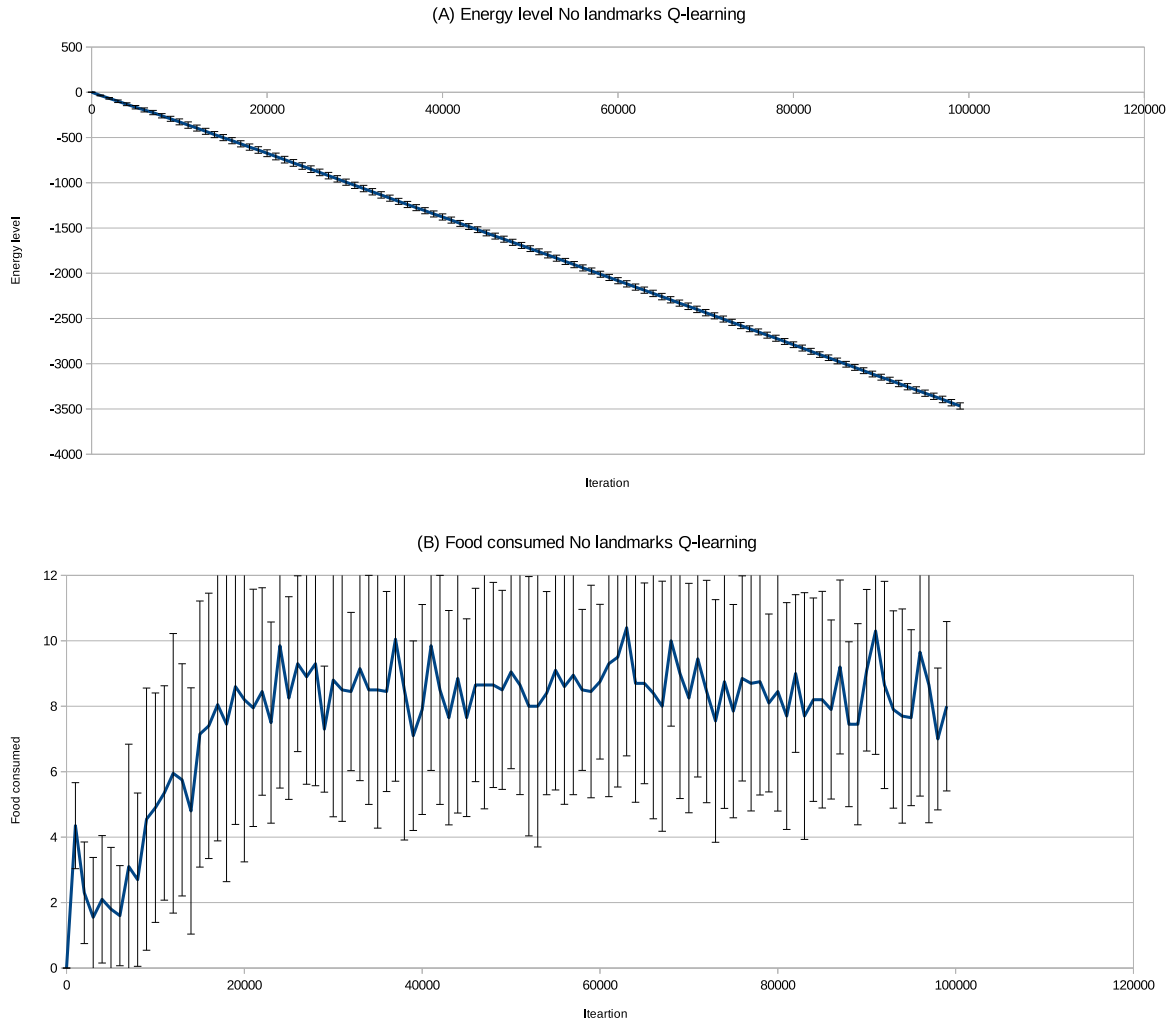

Figure 4.14: Result of 0 Landmark test with Q-learning.

4.7 Position sensor test

Figure 4.15 shows the average energy level, the average total number of food consumed between each sample and the average number of sensor nodes in the network over 20 different runs with each running for 100 000 iterations. Figure 4.16 shows a similar simulation with Q-learning.

The the network animat stabilises fastest of all tests at iteration 7000 with a small variance over all. It consumes about 120 food every 1000 iterations and stabilises around 170 with a small variance. It has about 35 nodes in total when it stabilises and slowly increase after that and ends at 42 t the end o the simulation.

The Q-learning animat stabilises with very low variance after 20 000 iterations while dropping to -18 energy at the lowest, fig 4.16 A. It consumes 140 food every 1000 iteration when is stabilises, fig 4.16 B.

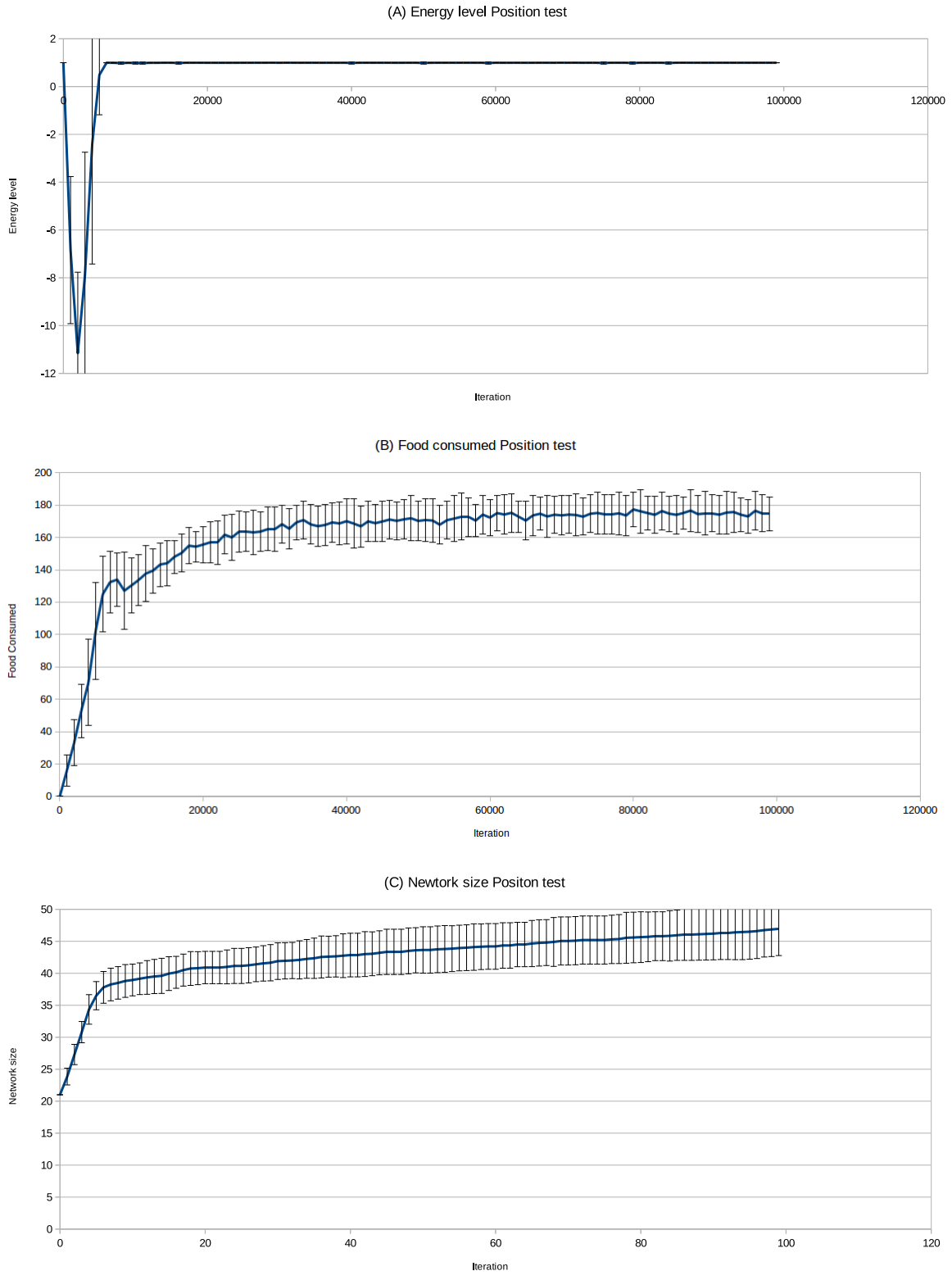

Figure 4.15: Result of the position test.

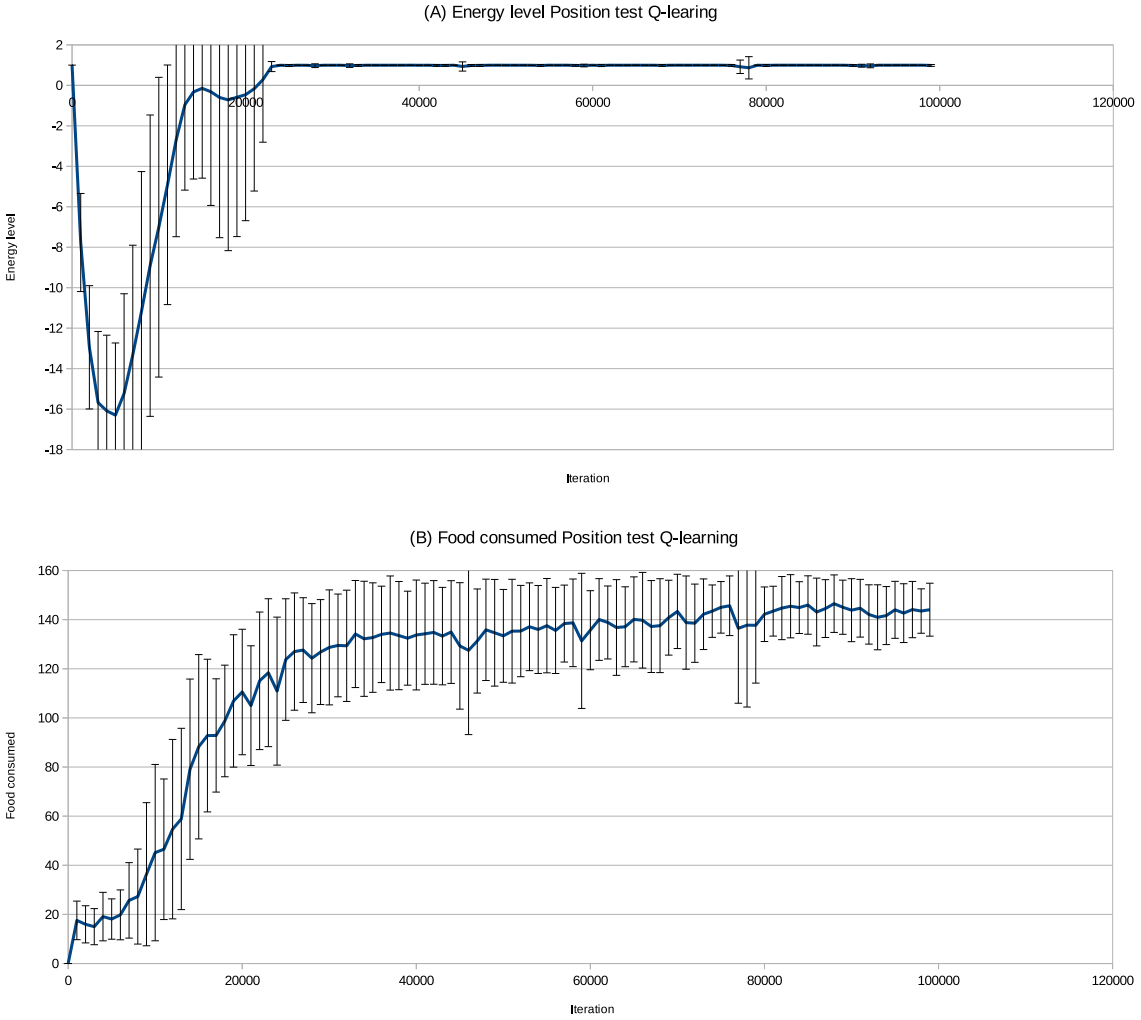

Figure 4.16: Result of the position test with Q-learning.

5

Discussion

5.1 Test specific discussion

The following subsections gives a short discussion on each of the different tests.

5.1.1 3 landmark test

The results of the tests without any supervision or reward shaping(figure 4.1, 4.2 and 4.4) shows that the cannot learn to navigate in the limited time of the tests. The problem is that initially the animat does not have any previous experience and therefor try actions randomly. As most of the positions do not contain any food the animat quickly learns that the Eat action gives a penalty. After this the animat only uses this action with random exploration. For the animat to learn that performing the Eat action at the food be at the correct position and perform the random action Eat. The probability of performing the Eat action is $\epsilon\frac{1}{N_{actions}} = 0.05\frac{1}{9} \approx 0.006$ and the probability of the animat to make this random exploration at the right position $0.05\frac{1}{9}\frac{1}{10x10} \approx 0.00006$. The numbers are not exact as the probability of the animat being in all positions are not equal.

This result shows the the animat needs something to help it find the food in the early part of the simulations. This is the reason why the reward shaping was introduced to the animat.

5.1.1.1 Test with reward shaping

The animat made good use of the ability to combine sensor positions. With this ability it could easily differentiate between positions and is able to have a clear policy for each situation. The Q-learning animat on the other hand did significantly worse as it has more difficulties to differentiate between states as it can only learn to differentiate between states by updating its Q-values. The result is that it sometimes confuse its position and moves in the wrong direction. This leads to the animat in general not getting enough food to sustain its energy need. A subset of the simulations stabilises where the other part does not.

5.1.1.2 Test with high reward shaping

Both the network animat and Q-learning is performing on a similar level with the network animat having a small advantage.

Although the network expands at a similar rate in figure 4.7 C as in the test with lower shaping reward, figure 4.5 C the animat fails to stabilise. The animat has in single runs been observed to move in cycle very close around the food for long periods of time. The reason for this is that the shaping reward given to the animat is similar to what it would receive from eating. When it is caped on energy the reward for eating is even more diminished resulting in the animat.

The cycle the animat enters is three states long. It moves into positions of distance 0, $\sqrt{2}$, 1 and then back to 0. The resulting shaping reward it receives in this cycle is -1, +1, +1 making it gain +1 reward every 3rd iteration. The average reward for each iteration is $\frac{1}{3}$ which in most cases can receive with perfect navigation where the animat receives at most 0.6 every 4th or 5th iteration an average giving 0.15, this not considering the shaping reward penalty it receives when it is relocated.

5.1.2 9 landmark test

With 9 landmarks both the network animat and the Q-learning preformed worse then with 3 landmarks. The animat with network there are some instances where some runs stabilise around 70 000 iterations while the majority still fails to learn to navigate effectively. The animat without network on the other hand consistently fail to learn.

Both of the animats has difficulties to interpret the large amount of information given from the landmarks. The network animat can combine different landmark directions but it will still consider two tiles equal if it has at leas one landmark that activate the same sensors on both tiles.

With the set of sensors and the number of landmarks many of the tiles in this world are identical to each other which leads to inconsistencies in the optimal state action relation.

5.1.3 1 landmark test

The network animat cannot find its way to the food in a reliable way with only one landmark. Although the mean need value don't drop lower then -20 and then starts to increase quickly after the lowest point the time it would take to stabilise is far to long to be called a success. One interesting thing is that the mean number of food consumed is higher in the beginning of the test run compared to later in the run. The maximum number of nodes in the network is 11 and the node is useful to the

animat so it creates it fairly quickly.

The Q-learner on the other hand cannot stabilise without the extra node. The need is also consistently decreasing. The Q-learner will not likely be able to navigate in this world.

5.1.4 Reward shaping test with no landmarks

Both tests are displaying identical result. This is not strange behaviour at it only has one sensor and cannot create new nodes. The animat does not stabilise in either of the tests.

The animat has two states to work with. Either its food sensor is active or it is inactive. As the animat only gets the shaping reward after it performs an action the has to relearn how what action is best in the state of "no food". This explains its poor performance as it never leaves the learning phase.

5.1.5 Position sensor test

Each tile in the world has a unique set of active sensors which makes every tile have a unique footprint. The animat takes great advantage of this as this animat stabilises fastest of all the tests and stays stable throughout the test run. This effect is also apparent with the Q-learner where it is the only Q-learner that reliably stabilises. The time to stabilisation is slower than that of the network animat. This is most likely due to that the network can combine the X- and Y-sensors and more quickly identify specific tiles.

5.2 General discussion

The network animat performs better than Q-learning in all tests. The network animat generates more information for the animat where it can automatically differentiate between states which are uniquely represented by the landmarks. The amount of certainty the animat has close to the food the better it can navigate to it. The pure Q-learning can do the same but instead has to learn the active combination of sensors to identify the same state.

The minimum number of (x) food the animat needs to consume to increase its energy level is:

$$x * 0.6 - 0.001 * 1000 \geq 1$$

$$x \geq \frac{2}{0.6} \approx 3.33$$

Assuming the animat receives a penalty for trying to eat on other places and that it receives the full amount of reward with every consumption the animat needs to

eat every 333 iteration. The animat shows that the amount of food consumption needed for a reliable stabilisation usually needs to be at least 80 per 1000 iterations as all of the tests that stabilises consumes about 80 at there turning point, see fig 4.5 A , 4.5 B, 4.15 A, 4.15 B and 4.16 A 4.16 B.

The introduction of reward shaping improved the animats navigation significantly in the beginning of each simulation. The animats are drawn to the food which increases the probability that random eat action is performed on the food. This leads to the animat stabilising earlier and more consistent in each simulation. It seems like the reward shaping also decreases the occurrences of the animat exploring areas far away from the food for longer times.

Large amount of landmarks confuses the animat. The amount of information available to the animat seems to overwhelm it. The network animat is performing better the Q-learning most likely due to the automatic paring of sensor nodes with the AND-nodes.

5.2.1 Weakness of the model and result

This is not a fair comparison between the Q-learning and network as the network gains more information as more nodes are created and they are otherwise the same. A more fair comparison is to limit the number of new nodes that can be created or to run the Q-learning with a set of and nodes that combines landmark direction. Small tests was also conducted in the three landmark environment where the network was limited to the number of AND-nodes that could be created. These tests suggest that the animat can learn to navigate effectively with down to 5 extra AND-nodes and that significantly better performance was noticed with 2 new nodes. Further investigation and thorough testing is needed to make any conclusive statement of the importance of how much the extra nodes contribute to the navigation performance increase.

5.3 Future work

There are many ways this work could expand. Simple directions as doing more similar tests. Test how many new nodes are required for fast stabilisation. Introducing a new resource that the animat needs to learn to balance alongside the energy.

Implementing other navigation methods used by animals. For example Place cells seems like a promising model to implement as it mimics how neurons in animals brains interact when navigating an environment.

5.4 Conclusion

The results shows a clear example where the homing navigation model can, to some extent, successfully be introduced in the animat framework. The animat perform better with the network framework than with Q-learning in all of the tests conducted. This is due to the framework automatically gives more information about its position as it explores the environment. This successful adaptation of a navigation method strongly suggest other navigation methods also can be implemented in the network structure.

The results does not disprove the viability of the animat frameworks potential to produce an AGI. If further research can show results of the animat framework producing place cell functionality the viability of the framework increases immensely. On the other hand an AGI must be able to solve many other problems than navigation and the animat need large developments in most of these fields.

Bibliography

[1] B. A. Cartwright and T. S. Collett. Landmark learning in bees. *Journal of comparative physiology*, 151(4), December 1983.

[2] Geoffrey Hinton Yann LeCun, Yoshua Bengio. Deep learning. *Nature*, 2015.

[3] Somesh Jha Matt Fredrikson Z. Berkay Celik Ananthram Swami Nicolas Papernot, Patrick McDaniel. The limitations of deep learning in adversarial settings. In *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016.

[4] C Strannegård et al. The animat path to artificial general intelligence, 2017.

[5] Braitenberg V. *Vehicles: Experiments in Synthetic Psychology*. Cambridge, MA: MIT Press/Bradford Books, 1984.

[6] Edvard I. Moser, Emilio Kropff, and May-Britt Moser. Place cells, grid cells, and the brain's spatial representation system. *Annual Review of Neuroscience*, 31(1):69–89, 2008. PMID: 18284371.

[7] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J. Chadwick, Thomas Degris, Joseph Modayil, Greg Wayne, Hubert Soyer, Fabio Viola, Brian Zhang, Ross Goroshin, Neil Rabinowitz, Razvan Pascanu, Charlie Beattie, Stig Petersen, Amir Sadik, Stephen Gaffney, Helen King, Koray Kavukcuoglu, Demis Hassabis, Raia Hadsell, and Dharshan Kumaran. Vector-based navigation using grid-like representations in artificial agents. *Nature*, pages 1–5, 5 2018.

[8] Christopher J.C.H. Watkins. Q-learning. *Machine Learning*, 8, May 1992.

[9] C Strannegård and N Engsner. Artificial animals and general intelligence. Unpublished paper draft, February 2018.

[10] Hampus Torén and Fredrik Mäkeläinen. Optimizing survivability for artificial animals in dynamic environments. Unpublished master thesis, June 2018.

[11] Stuart Russell Andrew Y. Ng, Daishi Harada. Policy invariance under reward transformations: Theory and application to reward shaping. *ICML*, 99, June 1999.