

Homeostatic Adaptation for Sensor Disruption and as a Mechanism for Action Selection

Candidate Number: 73495
Adaptive Systems Coursework
MSc Evolutionary and Adaptive Systems
COGS-GRC

April 23, 2003

1 Introduction

A homeostatic adaptation mechanism based on Ashby's (1960) Homeostat and his idea of 'ultrastability' previously explored by Di Paolo (2000, 2003) is implemented on a Braitenberg Vehicle architecture. This robot is able to adapt to radical sensor disruptions (such as sensor inversion) and in the search for making more autonomous robots, homeostatic adaptation is proposed as an action selection mechanism.

Di Paolo suggest that it is time to move to the next level in the field of artificial intelligence—that is, to design robots based on organismically-inspired ideas beside the biological-inspired ones being used in modern artificial intelligence. Instead of designing robots to *behave* like an animal, we have to design them to *be* like an animal, if our main task is to make them more autonomous.

In this project one of Di Paolo's experiments on homeostatic adaptation to sensor inversion is reproduced (with some slight changes) and complemented. Homeostatic adaptation is going to be proposed as a mechanism for action selection, without pretending to solve all the problems and model all the possible phenomena that this involves. It is going to be shown how a homeostatic adaptation mechanism can be used to make a robot pick between two independent behaviours by reconfiguring his organisation to maintain some 'essential' variables within a range.

The idea that a robot should adopt a particular behaviour because he needs it and not because it was design for it is going to be explored.

One difference between animals and robots is that they generate their own purposes, and that these puposes have a meaning to the animal. In the case of a robot, the purpose has a meaning for the designer and not for the robot itself, further on, if a robot performs badly on a task it doesn't even notice, while for an animal it can be a matter of life or death.

Animals adapt when their internal stability is threaten, when they are aware that something is not going as well as it should be going, and that it is necessary to adapt because they need this stability in order to be alive. One of the properties of living organisms is that they constantly rearrange their structure to deal with a changing environment.

Robot control systems are design so that a robot fulfils a function, they are based on the robot's performance on a specific task and at no moment they concerned about the internal structure or the stability of the robot. There is no autonomy on the robot. The tasks it is completing have no meaning for it but for a designer. If we really want a robot to be like an animal the robot has to generate his own tasks, he has to behave in a certain manner in order to survive and to feel stable (or safe), and in some way he has to 'feel' threaten when its performance is not good enough.

It is a known fact that a sensory-motor coordination mechanism is needed in order to adapt to certain types of sensor disruptions. (Held, 1965) Whenever an organism is in a situation where one of its senses have been distorted, in order to adapt the individual has to 'move' in the environment, adjust its configuration, and coordinate its movements with information about the 'new' effect these movements have on the environment.

During an animal's lifetime, adaptation to their constantly changing body (e.g. limp growth and eye separation) has to take place within the internal organisation of the individual. This adaptation process can't be genetically encoded since the information required would be immense. We know that not every behaviour is hardwired by evolution, and that there is a developmental phase where the internal configuration of the individual moulds according to its sensorimotor experiences.

The ability to adapt to radical changes such as the inversion of the visual system cannot easily be explained in terms of evolutionary pressures, however research have shown that animals are able to adapt to such disruptions. (Kohler, 1964)

Control systems are usually not adaptable to radical changes (such as sensor inversion or sensor damage), some are based on linear approximations of a known unchangeable function—describing the robot-environment interactions—which is not allowed to change if the nature of the interactions change.

2 Background Concepts

By behaviour we mean; the observed result of an interaction that a specific organisation (or internal configuration) of the robot has with the environment. It is a consequence of an agent-environment-observer interaction and is determined by the internal-external configuration of the agent, as well as by a surrounding environment. This means that in order to survive, the organisation of the agent has to couple with the outside environment. The organisation of the system includes the parameters describing the sensory-motor links, as well as a description of the sensors and the motors, and any rule that influences the behaviour of the system.

Taylor (1962) attempted to use Ashby's (1960) framework of the 'ultrastable' nature of organisms to explain adaptation to visual disruptions. By using the same Ashbyan ideas Di Paolo (2000, 2003) showed that this framework can be used in a robot to make it adaptable to such disruptions.

Living organisms have essential variables that they (themselves) regulate within certain limits. Examples of these variables are temperature, ionic concentrations, hormones, some internal liquid pressure, humidity, etc. To regulate all of these variables, living organisms have to change their internal compositions and organisation, and therefore their behaviour.

In one of Di Paolo's robots the 'essential' variable is the energy level of an internal battery which can be recharged with energy coming from light sources in the environment. In order to keep this essential variable within the limits the robot has to perform phototaxis. Using Ashby's (1960) idea of ultrastability whenever the variable goes out of bounds, 'breaks'¹ will take place in the internal configuration of the robot, until this essential variable returns within its natural bounds.

Homeostasis means (for the purpose of this project) to maintain some essential variables (energy level) within some strict limits. Homeostasis is a characteristic of living organisms—maybe not with the strict limits of Ashby's framework— but organisms do tend to have physical variables stable within a certain range which they regulate by reconfiguring their organism.

3 Sensor Disruption

Experiments where an individual wears goggles which invert or distorts the visual field, can cause an individual to fall down, crash into furniture or try to reach objects where they're not, however after a while they adapt and begin to see the world almost normally again.

Experiments on the limits of adaptation in humans and animals have shown that sensory-motor coordination process is more important for an individual to adapt in the case of a sensor disruption than simply an error-correction mechanism. (Held, 1965)

3.1 Model

The architecture used for the robot is inspired on a Braitenberg Vehicle (Braitenberg(1984)). For simplicity the robot is circular (radius $R=4$), with 2 light sensors (left and right) on the front (separated on average $2\pi/3$ radians) and one motor on each side which can move backward or forward. Random (5 % uniformly distributed) noise is added to the position of the sensors, so that on average, each sensor is located $\pi/3$ radians from the front of the robot to the respective side. Both of the sensors are connected to both motors with direct sensory-motor links (which are described in section 3.1.2)

¹Breaks in Ashbyan terms, are discontinuous changes in the parameters that describe the system.

3.1.1 Dynamics

The dynamics are very simple since the robot is massless and there is no friction with the surface. The robot has a position, an orientation and can move in an infinite 2D arena. The activation of each of the motors is the tangential velocity at that point, so that the linear velocity of the robot at each time step ($dt = 0.2$) is

$$v = 0.5(M_l + M_r) \tag{1}$$

and the angular velocity is

$$\Omega = (M_l - M_r)/R \tag{2}$$

where M_l and M_r are the activation of the left and right motors respectively. The new coordinates and orientation of the robot are obtain using first order Euler integration method.

3.1.2 Sensory-Motor Connection

Both sensors are connected to both motors with a sensory-motor link that is specified with the following mapping function

$$M = A \sin(\omega S + \phi) \tag{3}$$

where M is the contribution from one of the sensors to the motor activation, S is the activation of the respective sensor and A , ω and ϕ are the parameters that describe that specific connection (see figure 1).

The contributions from the 2 sensors are averaged to give the activation of the respective motor.

These 12 parameters (3 per connection) are chosen randomly at the beginning of the simulation, so that A is between the range $[8,12]$, ω between $(0,1]$ and ϕ between $[-\pi, \pi]$.

Parameters are allowed to change when the energy level of the robot is below or above certain thresholds (see section 3.1.5). There was no restriction on the values that these parameters could take, with the exception that amplitude and frequency should be positive. By not having such restrictions the mapping function within the sensor activation range $[0, 10]$ can take forms similar to the ones in Di Paolo(2003)².

²The mapping function is centered on the sensor axis, for a more complete function an offset value should be added, a further discussion is given in section 3.3.

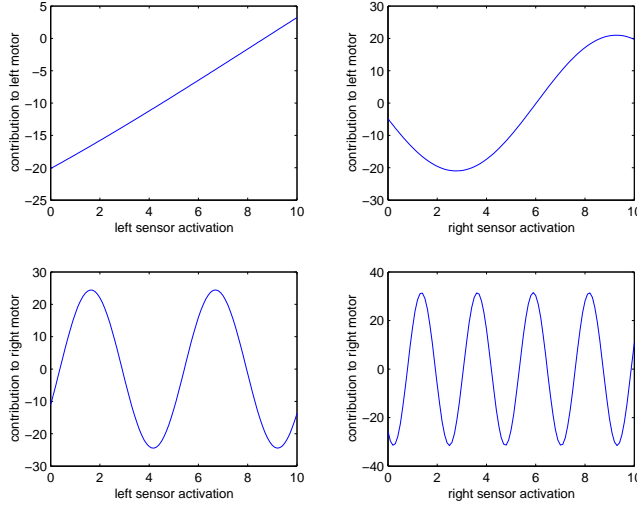


Figure 1: Mapping functions between the sensors and the motors for one particular successful individual that is able to perform long term phototaxis. The total activation for each motor is the average contribution from both sensors.

3.1.3 Energy levels

The robot has an internal battery that can be recharged with energy coming from light sources (or lamps) and decays exponentially with time. A term related to the robot wasting energy (by moving its motors) is introduced as well. The differential equation governing the dynamics of the robot's energy level is

$$\tau_E \frac{dE}{dt} = 10 \left(\frac{S_l + S_r}{2} \right) - \gamma |M_l + M_r| - E \quad (4)$$

where τ_E is a time constant for the decay of the energy level (E), S_l , S_r , M_l and M_r are the activation for left and right sensors and motors respectively.

3.1.4 Lamps

One lamp at a time is placed at a random distance (between [50,100] units) from the robot and at a random angle (between $[0, 2\pi]$ radians). Light sources have an intensity that varies on the range [500,1500] and a life-time between $[0.75 T, 1.25 T]$, where T is a parameter that can take the values 100 or 400, after that time the lamp is turned off and a new lamp appear.

The activation of the sensors is the intensity of the light divided by the squared distance from the sensor to the source, activation values are constrain to a maximum of 10. Light is also occluded by the robots body ³.

³Using that $\cos(\theta) = (a \cdot b) / (|a||b|)$ you can find the angle between the normal vector at

3.1.5 Plastic ‘Breaks’

The energy level of the battery has to be within 2 boundaries (E_{min} and E_{max}). Soft and hard plastic changes are applied to all of the parameters that define the mappings between sensors and motors whenever the energy level goes out of these boundaries. Different values for the lower threshold have successfully been tried ($E_{min} = 2, 3, 4, 5$) as well as for the upper boundary ($E_{max} = 8, 10, 12, 15, \infty$).

Plastic changes consist on adding a random number (that can be negative) to each one of the parameters. It is determined as follows: a random number between $[-0.5, 0.5]$, scaled according to the respective parameter and multiplied by a ‘rigidity’ factor. The scaling factors for the amplitude, frequency and phase are 10, 1, and 1 respectively. The ‘rigidity’ factor is constant ($p=0.1$) for hard changes, or varies according to a ‘rigidity’ function (shown in figure 2) for soft changes. The ‘rigidity’ factor is determined by 2 parameters ($\alpha = 1.5$ and $\beta = 1$), the low boundary E_{min} , the upper boundary E_{max} and 2 energy values $E1$ and $E2$ which determine the ‘softness’ of each boundary. The equations governing this factor are

$$p = \exp \frac{(-x + E1)}{\alpha(E_{min} - E1)} \quad (5)$$

for the low boundary and

$$p = \exp \frac{(x - E2)}{\beta(E2 - E_{max})} \quad (6)$$

for the upper boundary.

3.2 Results

At the beginning of each run the robot starts with random connections and the minimum permitted energy (E_{min}) within the safe zone. The energy level drops quickly and the parameters describing the sensory-motor connections start to change. Finally the robot reaches a configuration where its strategy is good enough and allows it to gain the necessary energy to be within the stable zone where no further internal change takes place. On this first stage the robot has to adapt to the ‘new’ world, it has to reconfigure its internal hardware in order to survive—that is, to gain more energy than what it consumes, but also to keep it within the bounds his organisms can tolerate.

Several adaptation experiments were carried out to see how well the robot adapts to severe sensor disruptions. The term goggles is going to be used to refer to a sensor disruption situation, since most of the research in this area has been done with visual distorting goggles (see Kohler, 1965).

the sensor and the vector from the sensor to the light source. Vectors whose angles are more than $\theta > |\pi/2|$ are occluded

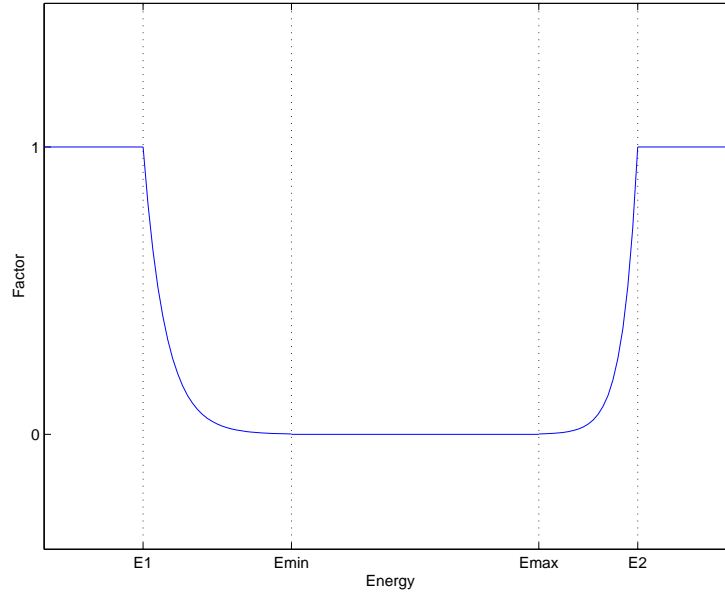


Figure 2: The ‘rigidity’ factor for soft boundaries plotted against the battery level.

Because of the nature of the sensory-motor links the robot can approach light sources in several different ways (see figure 5). Some are better than others in the sense that the average energy level—after the adaptation transient—is higher. The robot can sometimes approach the light source by moving backwards using the advantage that light is occluded, or it can turn around the light source in circles of different radius. The strategies used by the robots to approach light are sometimes too efficient and can raise the energy level above the upper limit. Given this case two main things usually happened: 1. The internal state of the robot change to a configuration where the energy obtained is stable within the boundaries, or 2. The energy raises to high above the boundary and the robot with its new configuration is not able to maintain its energy level within the stable zone, when the energy level is below the lower limit plastic changes take place again.. On figure 3 you can see examples of these 2 cases.

Robots are able to adapt to sensor inversion (figure 3). After the sensors are inverted the energy level drops because the robot is not able to perform phototaxis any more. Some time after the sensors are inverted, the robot is able to perform phototaxis again and raise its energy level.

Sometimes an adapting robot with low energy (below threshold) finds a good configuration to perform phototaxis, but because it keeps changing as long as it is out of the safe zone, loses his phototactic ability before reaching the boundary. Almost always he finally manages to reach the stable region. (see figure 3 just after the sensor inversion) .

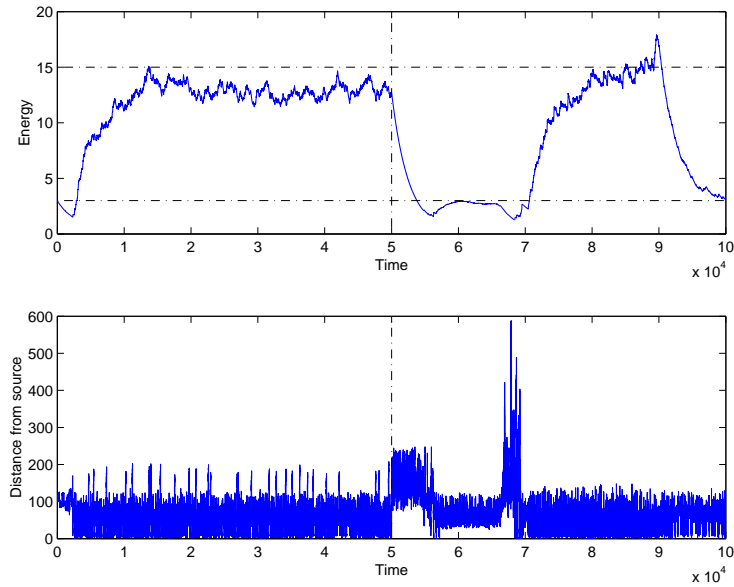


Figure 3: a) The energy of the robot is plotted against time. Energy boundaries are shown with horizontal lines and the vertical line show the moment where the sensors are inverted. After sensor inversion follows an adaptation phase where the robot performs very badly. The energy decays again at the end because of the energy rising to fast over the upper boundary.

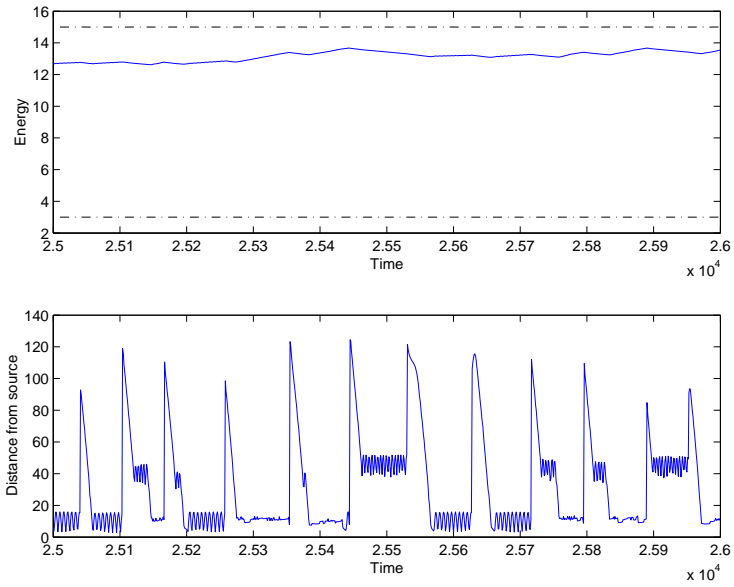


Figure 4: Zoom from figure 3. A) The energy of the robot is plotted against time and looks very stable. B) The distance to the source remains (almost) constant for small periods of time (when the light is present and the robot is already in the vicinity of the lamp). Peaks show the time period where the robot is approaching the light source. Not all the ‘constant’ distances are equal since the robot can approach light sources using different strategies (see figure 5), also lamps have random positions and intensities.

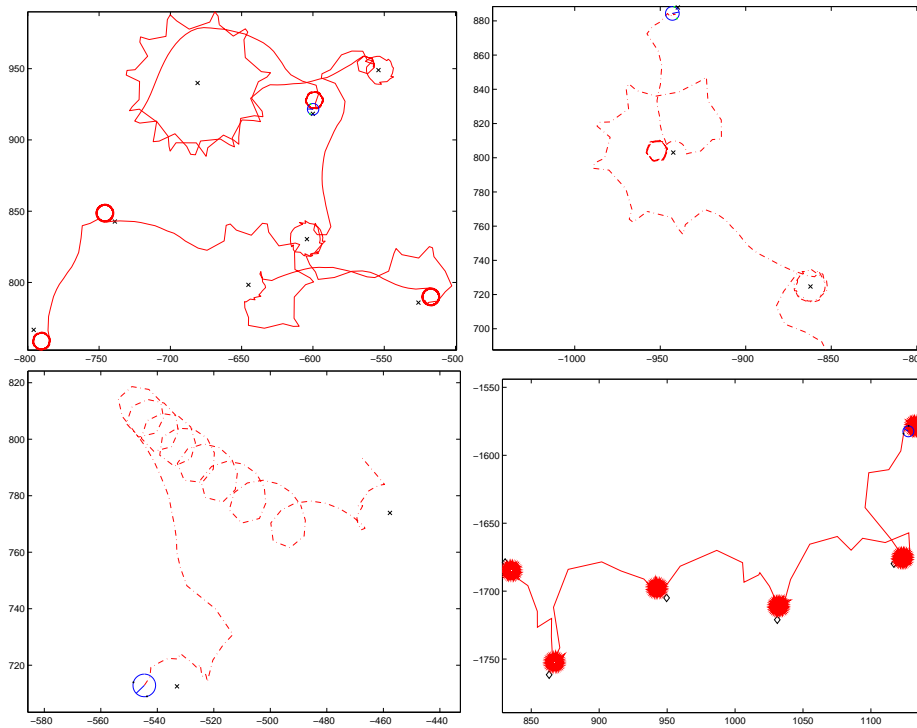


Figure 5: Examples of different strategies used to approach light. Light sources are marked as crosses or diamonds. The line show the trajectory of the robot, and a circle is the robots final position.

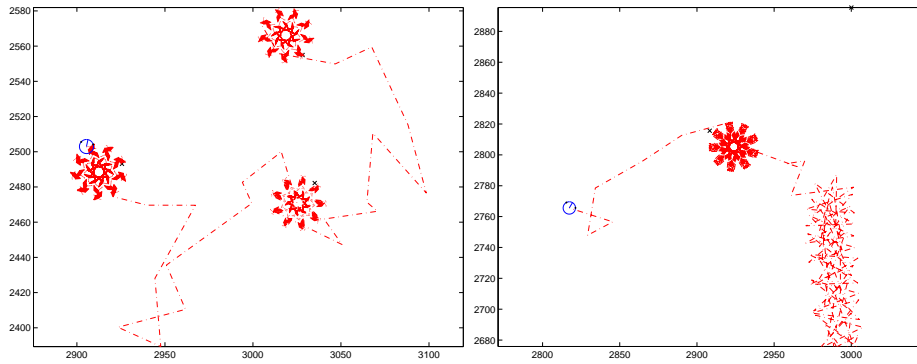
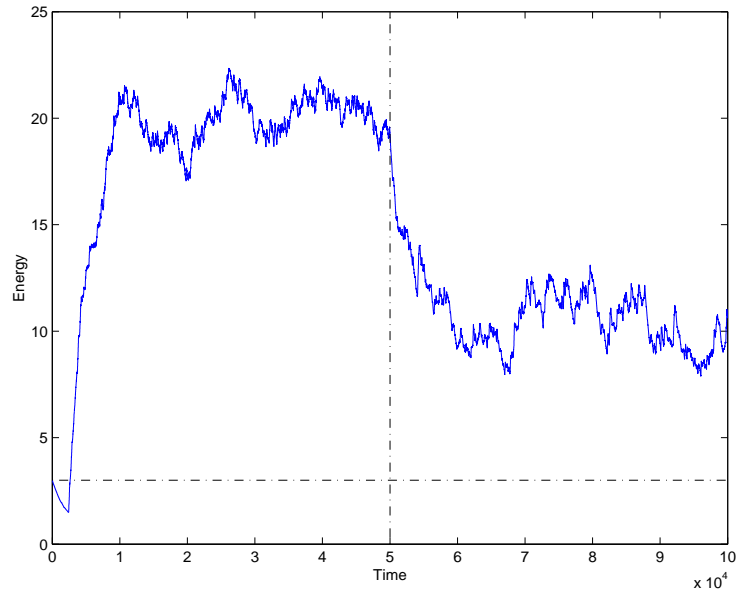


Figure 6: After the left sensor is damaged (25% performance) the robots performance also decreases, and since the energy didn't leave the stable zone no changes were made to the robots internal structure. Strategy used to approach light before (lower-left) and after sensor is damaged (lower-right) differ slightly. It can be seen that the strategy used before the sensor was damaged is more efficient.

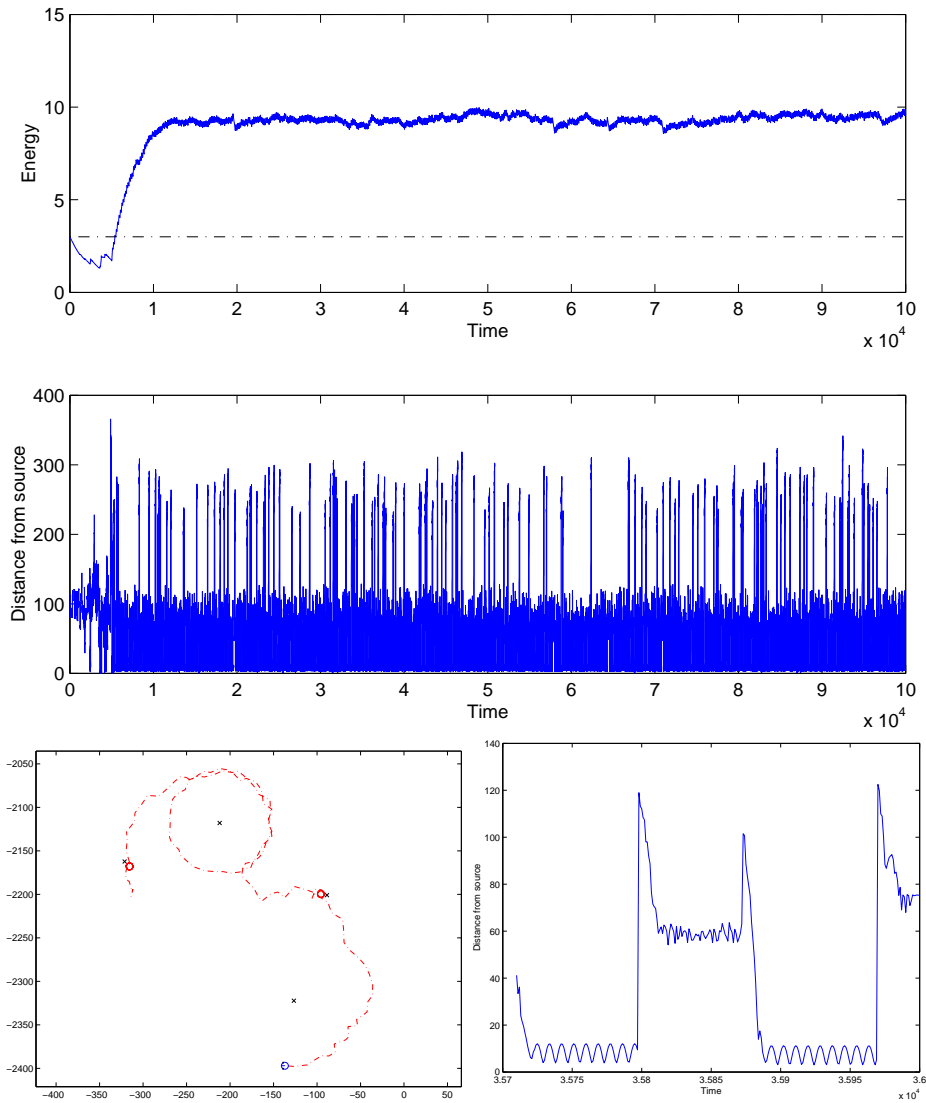


Figure 7: When the robot puts on and takes off the inverting goggles every light presentation it is able to adapt to both conditions(top), but still performs better in one of the 2 cases. The robot comes closer to the light without the goggles than with the goggles on (or the other way around).(lower left and right)

Cases where the robot lost part of his visual system and was still able to keep the energy level within bounds were observed, see figure 6.

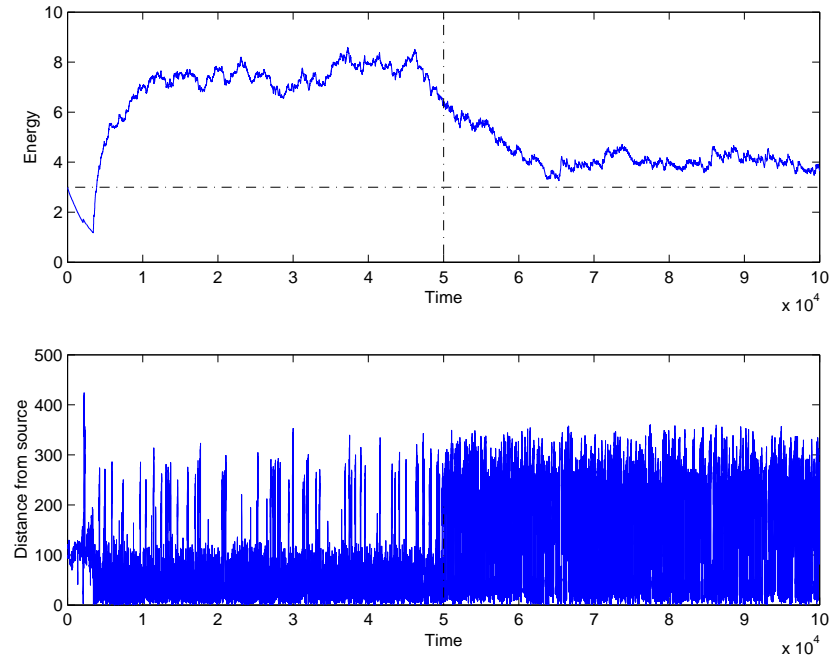


Figure 8: Halfway (vertical line) the robot puts on and take off the inverting goggles every new light presentation, since the robot’s strategy is good enough to ignore half of the lamps, it doesn’t need to adapt again.

When the sensors in the robot are inverted at every light presentation, the robot is able to adapt to both conditions, however it uses different light approaching strategies with and without the inverting goggles. (see figure 7) Figure 8 is another example of a sensor disruption that reduces the robots performance but is not enough to cause plastic changes in its internal structure.

Figures 10 and 11 illustrate cases where after the robot has adapted to the disruption, the robot actually performs better and is able to achieve a higher average energy level.

Soft and hard plastic changes had a very similar effect.

3.3 Discussion

Although the mapping function is different from that used by Di Paolo (2003), it is complex enough to make a Braitenberg vehicle perform phototaxis. For more complex behaviours an offset parameter can be included in the mapping function. The decision to use sine functions was made because they are continuous and differentiable, and by varying few parameters it can take many different

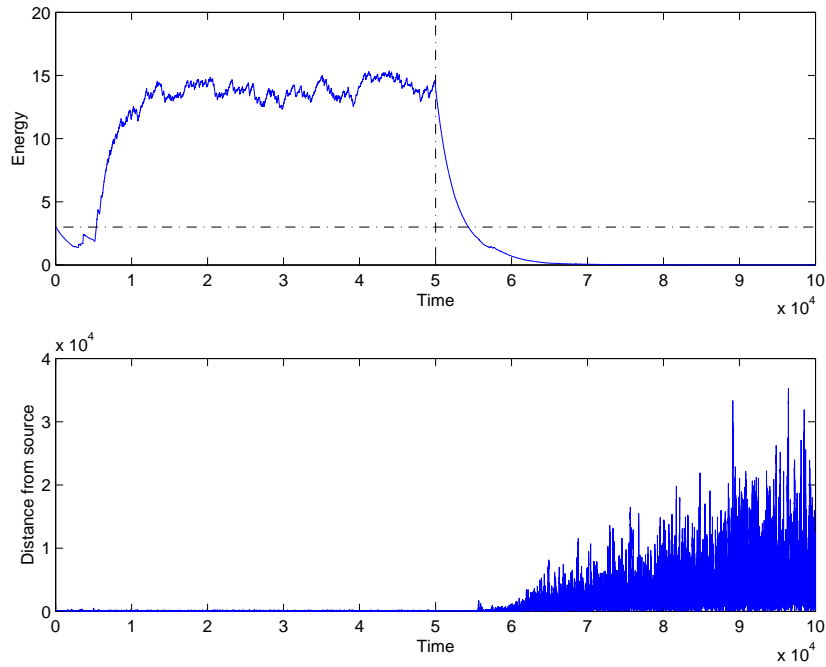


Figure 9: The effect of turning off one of the sensors is that the robot is not able to adapt. There might be a possible strategy to approach light with just one sensor using the information coming from light occlusion, but still it would take a lot of time for a robot to adapt to a severe change.

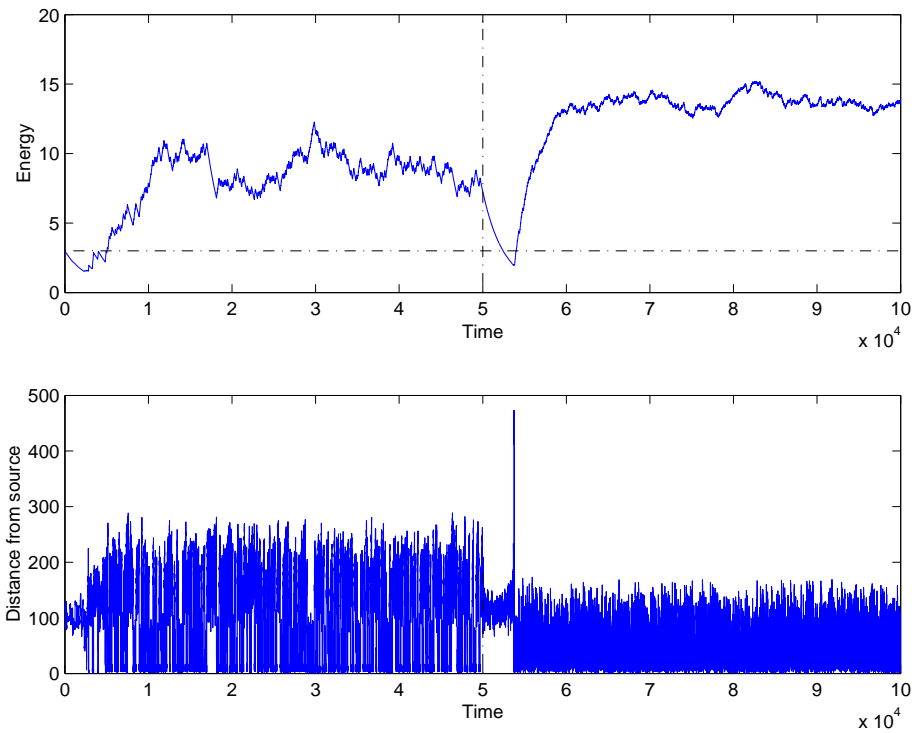


Figure 10: At the moment the robot puts on some goggles which reduce the light intensity (vertical line), its energy level drops very fast, however after a short while it adapts and even improves its performance.

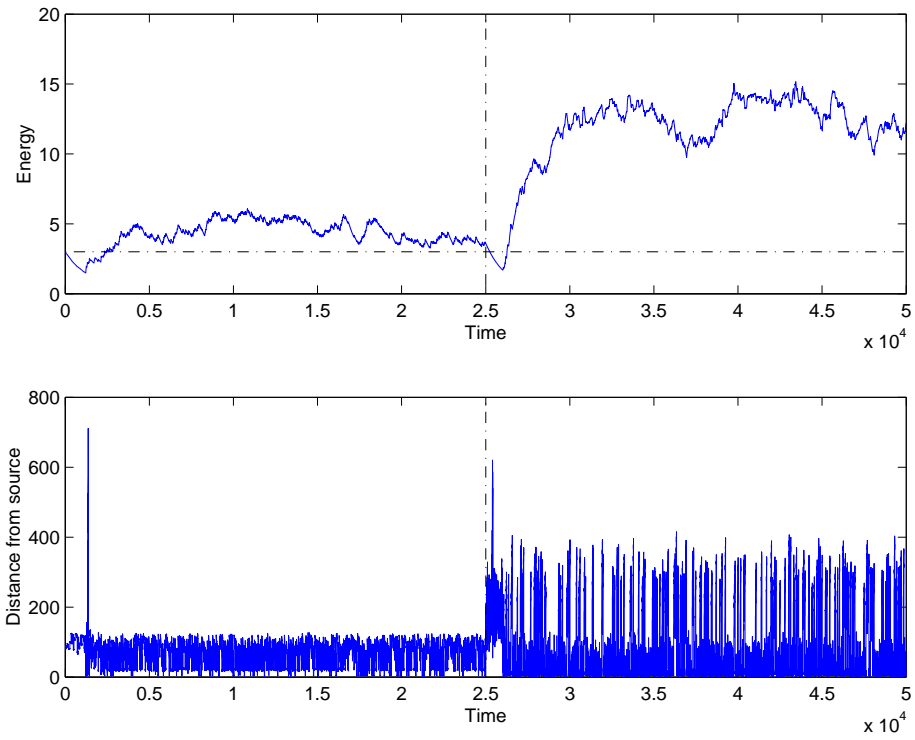


Figure 11: Adding a constant value to the left sensor ($S_l + 1$) (vertical line) makes the robot decrease performance but after a small while adapts and even performs better.

forms within the sensor range.

Most (not all) of the individuals reach a stable state where the robot is able to keep his energy level within the margins.

It is interesting how an specific individual can be defined here. It can't be defined by its internal parameters—that describe the sensory-motor mappings—because they are constantly changing, neither can be described by the properties of it's external structure (such as sensors or motors), because we could also change their properties and the robot would still have the same identity. It can't be identified by an observed behaviour since it changes every time the environment or the internal structure changes. The only way an individual can be identified is by specifying the (physical) rules that govern that individual's dynamics during his lifetime.

One of the interesting phenomena which arise is that although the rigidity function (figure 2) is design to make the plastic changes smaller as you get closer to the boundary, sometimes that small change reconfigures the internal structure to a state where the robot completely stops doing phototaxis until its stability feels threaten again. Cases like this one make this robot a bit more autonomous than other phototactic robots, in the sense that it is following the light source for himself, and if he already is charged-up, he can stop doing it until he feels the need to do it again (i.e. feeling hungry).

A term was included in equation 4 standing for the robot wasting some of its energy by moving, this term didn't show to have any interesting effect in this particular case and was removed. But it might be useful in other cases and can be a replacement of the exponential decay of the battery. This idea can be further explored since organisms do consume energy, not only by moving but also by reorganizing themself..

One of the problems about Ashby's model is that adaptation occurs only when the essential variable go out of the bounds, which doesn't leave a space to include the concept of maladaptation. Figures 6 and 8 are cases where the individual after the distortion is not as good as it was before, and still does nothing about it, the robot doesn't even notices that its left eye is working at 25% of the normal eyesight, or that he is now wearing inverting goggles every other light source presentation. This framework can be useful for the purpose of illustrating that internal stability can make a robot do something, and makes the robot a bit more autonomous.

Ashby's model might not be very biologically plausible, but it is useful to illustrate the idea of homeostatic adaptation, and that a robot can perform a task just by trying to keep his internal structure stable and his essential variables within some limits. The robot following a long term phototactic behaviour is just a consequence of coherence between the structure and organisation of an individual with the environment, that evolve the system towards an stable region.

4 Homeostatic Adaptation as a Mechanism for Action Selection

The problem of action selection —meaning how does an animal or a robot comes to be doing one thing rather than another—is of interest in the field of adaptive behaviour.

Seth (1998) has shown that with a set of continuous active sensory-motor links an agent can show a full range of action selection phenomena. He uses artificial evolution to determine the form of the connections between the sensor and motors to find an internal configuration for his robot which enables it to approach food, water and avoid traps. Instead of using artificial evolution to shape the sensorimotor connections, homeostatic adaptation is going to be used. Di Paolo(2003) and in section 3 have shown that a homeostatic adaptation mechanism can make a similar robot to the one used by Seth to perform phototaxis, and Seth showed that an action selection mechanism can be the result of a coherence between continuous and concurrent sensorimotor links. By combining both ideas a similar robot to the one treated before(3) has to maintain 2 energy levels which are feed with different types of fuel within certain margins. Previous approaches to this problem used modules and winner-take-all hierarchies (Tyrell (1993)) and characterized behaviour as an action that the system could take, instead of an agent-environment-observer interaction. Seth's approach proves to be more biologically plausible method for action selection. Continuing with the idea that behaviours observed in autonomous robots must have meaning and intentionality, and that robots in order to be like animals should also have a need (or intention) to pick between a number of behaviours. Homeostatic adaptation is explored as an action selection mechanism.

The purpose of this model is to see how plastic changes can control the behaviour of a robot so that it maintains 2 essential variables within some bounds. We have already seen that the robot can adapt to maintain one variable within some bounds, even when dealing with severe sensor disruptions. Now we are going to see that the robot is able to maintain 2 essential variables by changing its internal configuration whenever one of the variables go out of bounds. The idea of this type of control is that the robot 'knows' when one of the essential variables is going out of bounds and has to do something about it (reconfigure).

4.1 Model

Using a very similar model to the one described in section 3.1, the problem of action selection is going to be studied. The robot now has 2 batteries —instead of one— whose energy level has to be maintained within bounds. In order to do this the robot has to decide to approach one from two different types of energy sources.

In the model used here a robot in order to survive has to pick up fruit and drink water by first approaching them and then passing over them. This

new robot has 2 sensors on each side—in exactly the same positions as in the previous model(section 3—each connected to both motors also in the same way. The sensors are of 2 different types and each type can detect 1 different source of energy (which can be analogous to fruit and water). With these sensors the robot is able to smell the presence of food (or water) in the same way as the previous model perceived light intensity with the sensors, but instead of charging its 2 batteries with the intensity received by each sensor type, the robot has to go to the energy source and pick it up (or touch it) – process which will increase the respective robot’s energy by one unit. Energy still decreases exponentially (now with $\tau_E = 15000$) and it doesn’t decrease with the robot’s activity. At the arena there is always one energy source of each kind, they are of radius 3 and as soon as it is eaten it disappears and another one appears in the vicinity of the robot. Energy sources have the same qualities (intensity and duration) as the lamps in the previous model and they appear in the same way as they did before.

The new boundaries are $E_{min} = 10$, $E_{max} = 40$. Soft plastic changes are used in the same way as in the previous model, with parameters $E_1 = 1$ and $E_2 = 50$. When both energy levels reach zero the robot dies.

4.2 Result

On most of the runs the robot is able to maintain both of the battery levels within the allowed margins. Figure 12 is an example of a typical run. At the beginning the robot’s energy levels oscillate more often between the 2 boundaries. After a while the state of the robot moves to a region in parameter space where the behaviour of the robot is able to maintain both energy levels within the boundaries for longer periods of time. Probably if you leave it long enough it will find an internal configuration where the system will be stable under normal conditions—meaning that the strategy adopted by the robot with this internal configuration will get him enough food and enough drink, so that none of his essential variables would run out of the boundaries.

Figure 14 show how the robot vary its feeding strategy through time. The robot sometimes focuses in getting one of the 2 food sources, sometimes the other, or sometimes it gets both. It can be seen how the agent is forced to stop picking up one kind of food and start picking up the other because one of his energies is running either too low or too high. Figure 13 illustrate this situation.

Many of the possible internal configurations will allow the robot to pick up both type of fuels and raise the energy levels, however not at the same speed. In a small time scale the robot approach both types of energy sources successfully and maintain its energy levels within the safe boundaries. But on a bigger time scale an adopted strategy can be not very good at obtaining one of the 2 sources, or can be too good at obtaining the other—raising the energy too low or too high—which can be harmful for the robot.

Although both behaviours can be seen as independent, the robot is using the same motors to achieve both. A change in the connections from one type of sensors with the motors is going to affect the performance in the other behaviour.

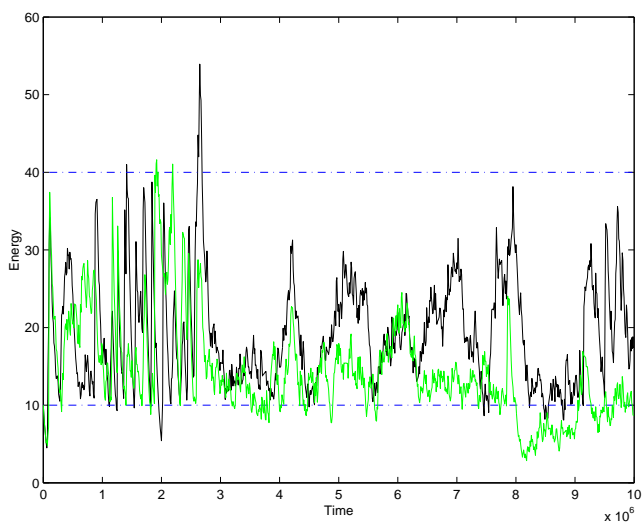


Figure 12: During a long period of time the robot maintains its essential variables ‘mostly’ within bounds. Adaptive changes are more often seen at the beginning and less often as time goes by. Towards the end of the run the state of the robot gets into a zone in parameter space where stability can be maintained for longer periods—less changes are required.

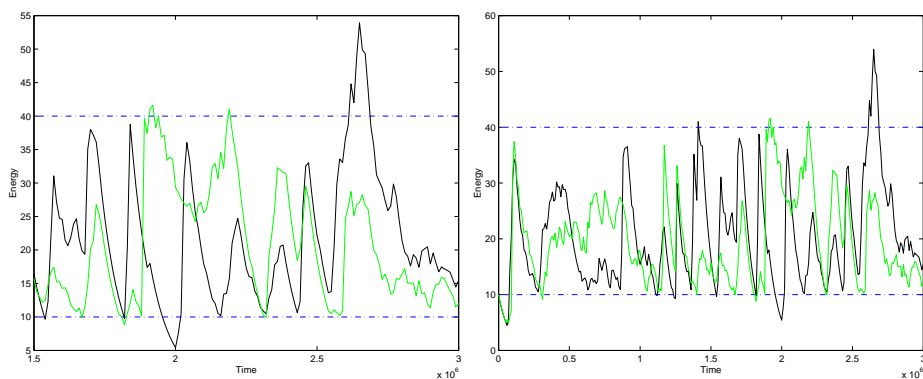


Figure 13: Both energy levels are kept mostly within bounds. When one of the energy levels goes out of bounds a change in the internal configuration of the system is forced. As a result the robot has to sacrifice what he is doing making the other energy level drop.

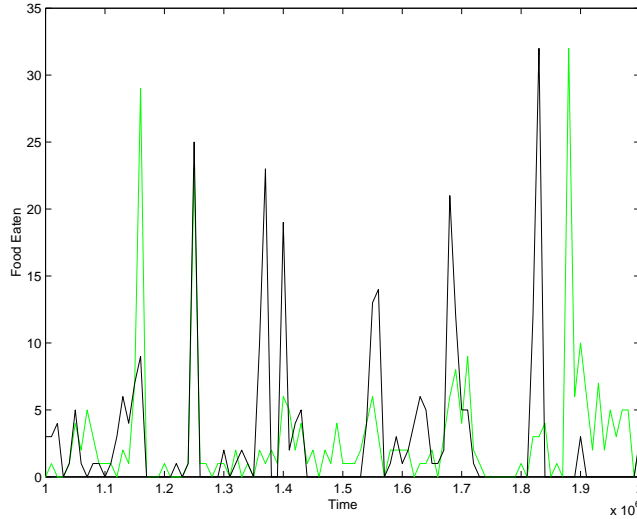


Figure 14: This figure show that the robot sometimes focuses on obtaining just one source of energy instead on having a constant strategy where it picks the same amount of each fuel. The environment changes very rapidly and randomly.

With no (or a very high) upper energy boundary, the robot can adopt an internal state which allows him to get a hold of the 2 sources maintaining both energy levels within the bounds, however it takes some time. Having an upper boundary for the energy is interesting, it gives the system a negative feedback mechanism besides the exponential decay of the battery, and it can be used as a model of satisfaction.

Comparing this model with Seth's (1998), this robot looks more alive (autonomous) since it acts because it needs to and not because the behaviour is evolutionary hardwired into it.

4.3 Discussion

The internal structure of the robot used in Seth (1998) was obtained using a GA to find the individual with the most fitness (average battery level). In this model the internal structure is constantly changing so as to keep the battery level within some limits. This model was not proposed to make an efficient robot, able to maintain very high energy levels, but the opposite, this robot is going to do something because it needs to, and if he already has enough he can lay down or wander around, until shortage is perceived in one of his energy storage.

The difference between both models is that Seth proposes that the behaviour adopted by a robot is the result of an interaction between a genetically encoded hardware with the environment, while in the homeostatic adaptation model, the

behaviour is the result of an interaction of a constantly changing structure—that needs to remain stable—with an environment. Seth’s model can be a good model of certain action selection phenomena, such as opportunism, while a homeostatic adaptation model can be better to model another (i.e. prioritising). The two models are based on different time scales, homeostatic adaptation is a good model of long term adaptation (days) while the mechanism proposed in Seth can be of the order of minutes or seconds, making it difficult to compare them thoroughly. To be able to compare both models, it could be useful to make an exact copy of Seth’s animat, but instead of running evolution to determine the parameters use homeostatic adaptation to determine the parameters of the sensorimotor connections in his animat. You can still use evolution to optimize some of the parameters, or the changing rules. Maybe approaching food or water are not good examples for this kind of adaptation rules. They are behaviours that we see as every day behaviours, and we can switch from one to the other without any difficulty. Homeostatic adaptation might prove more useful in modeling a long term action selection such as hibernation, or a hunting season. More investigation on homeostatic adaptation as an action selection mechanism is still needed, it has been shown that it is possible to have a robot performing different behaviours while keeping more than one essential variables within limits.

5 Conclusion

In an exploration on homeostatic adaptation to sensor disruption and action selection it has been shown that by forcing the essential variables of a system to remain within a region, the system will rearrange its configuration until it permits the robot to remain stable. There is still much to be done exploiting this new idea of organismically-inspired robotics. A new different framework from Ashby’s is needed, although it was useful to illustrate the missing ingredient needed to make robots more autonomous.

The last model used in this paper needs further exploration (with an extensive parameter search) and biological examples to compare to. It has been discussed that Homeostatic adaptation is a model of long term adaptation, so if we want to model a mechanism of action selection with this model, it should also have a long time scale. The energy decay time constant as well as the energy boundaries and their respective softness were picked using a trial and error technique, some further scanning is needed to get as much as it is possible out of this model. Some interesting results can be observed, a model has been proposed and a motivation of exploring ‘organismically-inspired’ ideas has been exposed.

A APPENDIX: PROGRAM CODE

```

/*****
LIBRARIES
*****/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>

/*****
STRUCTURES
*****/

typedef struct{
    //This structure has the position of the sensor
    //and the 2 different activations for food and water.
    double x;
    double y;
    double value[2]; //activation for food-water
}type_sensor;

/*****
typedef struct{
// Parameters describing sensorimotor link (sine wave)
    double amplitud;
    double w;
    double phi;
}sen_mot;

/*****
typedef struct{
    //This structure is to store information about the robot
    double x; //position
    double y;
    double angle; //angle
    double E[2]; //energy on its batteries
    type_sensor sensor[2]; //sensors information (left-right)
    double motor[2]; //activation motors
    sen_mot connection[2][4]; //sensorimotor links
    //first index stands for (0)left (1) right, and second index (0,2) left motor sensor1,sensor2
    // (1,3) right motor sensor1, sensor2
}type_robot;

/*****
typedef struct{
    //Information about the lamps are store in this structure
    double x; //position

```

```

    double y;
    double I; //intensity
    double T; //duration
}type_lamp;

/*****/

/*****
GLOBAL VARIABLES
*****/

type_robot animat;
type_lamp lamp[2];
//int LEFT=0;
//int RIGHT=1;
int googles=0; //'1' when the goggles are on

/*****
CONSTANTS
*****/

#define dt 0.2
#define R 4 //radius

#define NUM_LAMPS 2 //Number of batteries
#define LEFT 0
#define RIGHT 1

#define PI 3.1416
#define pi_1_3 1.0472
#define pi_1_6 0.5236

#define taoE 15000 //Time constant for energy
#define taoPmin 0.15 //constant determining softness
#define taoPmax 0.1 //constant determining softness

#define Emin 10 //energy boundaries
#define Emax 30
#define E1 1
#define E2 50

#define STOMACH 1 //how much energy does a piece of food mean
#define SIZE_FOOD 5 //distance between food and robot in order to be eaten
#define WINDOW 5 //how often to save in file

/* parameters for plastic changes */
#define delta_all 0.1
#define deltaW 1
#define deltaPhi 1//3.1416
#define deltaA 10

```

```

#define MAXT 100 //constant determining average lamp duration

#define tmax 1000000 // time of simulation
#define TINV 250000 //time where goggles are put on

#define SOFT 1 //flag, specifies type of plasticity

/*****
DECLARATION OF FUNCTIONS
*****/

void initialize();
void simulation_loop();
void calculate_new_state();
void motor_mapping(int);
void calculate_new_energy(int);
double add_noise_sensor();
void get_intensity(int,int);
void plastic_changes();
double modulus(double,double);
double myrand();
void random_lamp(int,double,double);
double calculate_distance(int);
double plastic_factor(double);

/*****
MAIN PROGRAM
*****/

void main(void)
{
    //srand48(time(NULL));
    srand(time(NULL));

    initialize();
    simulation_loop();
}

/*****
FUNCTIONS
*****/

void simulation_loop()
{
    int end=0,t,i;
    double t_lamp[NUM_LAMPS]={0},//time lamp has been on
           dist[NUM_LAMPS], //distance to each lamp
           food[NUM_LAMPS]; //counter for food eaten so far
    FILE *file;

```

```

file=fopen("robot_path.dat","w");

for(t=0;t<tmax;t++)
{
    calculate_new_state(); //updates world
    plastic_changes(); //if energy is low it applies changes

    /*saves information */
    if(t%WINDOW==0)
fprintf(file,"%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t%lf\t",
animat.angle,
animat.x,animat.y,
animat.sensor[LEFT].x,
animat.sensor[LEFT].y,
animat.sensor[RIGHT].x,
animat.sensor[RIGHT].y);

        for(i=0;i<NUM_LAMPS;i++)
{
    dist[i]=calculate_distance(i);
    if((t_lamp[i] > lamp[i].T)//||(dist[i]<SIZE_FOOD))
        /* if lamp is over */
        {
            random_lamp(i,animat.x,animat.y); //new lamp
            t_lamp[i] = 0;
            //This is when energy sources are discrete
            /*****
            if(dist[i]<SIZE_FOOD)
{
animat.E[i]+=STOMACH;
food[i]++;
}

            *****/
            //goggles!=(goggles); //put on and take off inverting goggles
        }
        t_lamp[i]++;

        // if(dist[i]<SIZE_FOOD)
        // animat.E[i]+=STOMACH;

if(t%WINDOW==0)
{
    //save more stuff
    fprintf(file,"%lf\t%lf\t%lf\t%lf\t%lf\t",
lamp[i].x,lamp[i].y,
animat.E[i],dist[i],food[i]);
    food[i]=0;
}
}
}

```

```

    }
}
    fprintf(file, "\n");

    /* inverting goggles */
    //if(t > TINV)
//goggles=1;

    }
fclose(file);
//fclose(file2);
}

/*****

void initialize()
    //this function initializes the animat, the lamps
    //the connections, etc
{
    int i,j;
    animat.x=1;
    animat.y=2;
    animat.angle=2;

    for(i=0;i<NUM_LAMPS;i++)
    {
        animat.E[i]=Emin;//rand()%20;
        random_lamp(i,animat.x,animat.y);
    }

    for(j=0;j<(2*NUM_LAMPS);j++)
    {
        for(i=0;i<2;i++)
        {
animat.connection[i][j].amplitud = 10; //rand();
animat.connection[i][j].w = myrand();//* PI ;
animat.connection[i][j].phi = ( myrand() - 0.5) * 2 * PI;

// printf("con %d %d \tamp %lf\tw %lf\tphi %lf\n",i,j,animat.connection[i][j].amplitud,animat.connect
        }
    }
}

/*****

void calculate_new_state(void)
    //in the function the position and state (sensors, motors, energy )
    // of the robot is udpatedrobot{
    int i;
    double v,w,sensor_angle,temp;

```

```

// Euler first order integration
v = 0.5 * ( animat.motor[LEFT] + animat.motor[RIGHT] ); //linear velocity
w = ( animat.motor[LEFT] - animat.motor[RIGHT] ) / R; //Angular velocity

animat.x = animat.x + sin(animat.angle) * v * dt;
animat.y = animat.y + cos(animat.angle) * v * dt;

animat.angle = animat.angle + w * dt;
animat.angle = modulus( animat.angle , 2*PI);

sensor_angle = add_noise_sensor();

/***** sensor COORDINATES *****/
/** with respect to the robot *****/

animat.sensor[LEFT].x = sin( animat.angle - sensor_angle ) * R;
animat.sensor[LEFT].y = cos( animat.angle - sensor_angle ) * R;

animat.sensor[RIGHT].x = sin( animat.angle + sensor_angle ) * R;
animat.sensor[RIGHT].y = cos( animat.angle + sensor_angle ) * R;

/***** NEW SENSOR VALUE *****/

for(i=0;i<NUM_LAMPS;i++)
    //for both types of batteries
    {
        get_intensity(LEFT,i);
        get_intensity(RIGHT,i);

        /***** Calculate new energy level *****/
        calculate_new_energy(i); //update energy
    }

if(goggles==1)
    //if goggles are on
    {
        /***** SENSOR INVERSION *****/
        temp=animat.sensor[LEFT].value[0];
        animat.sensor[LEFT].value[0]=animat.sensor[RIGHT].value[0];
        animat.sensor[RIGHT].value[0]=temp;
        /***** SENSOR DISRUPTION *****/
        animat.sensor[LEFT].value[0]*=0.1; // +=3;
        animat.sensor[RIGHT].value[0]*=0.1;
    }
/***** Calculate motor output *****/
motor_mapping(LEFT);
motor_mapping(RIGHT);
}
/*****/

```

```

void motor_mapping(int side)
    //This function maps the sensor inputs to the activation in the motors
{
    int i;

    animat.motor[side]=0;
    //for(j=0; j < 2 ;j++)
    for(i=0; i < (2*NUM_LAMPS); i++)
    {
        animat.motor[side] += animat.connection[side][i].amplitud
* sin( animat.connection[side][i].w * animat.sensor[i].value[i%2]
+ animat.connection[side][i].phi );
    }
    animat.motor[side] /= (2*NUM_LAMPS); //takes an average
}
/*****/
void calculate_new_energy(int i)
    // This function calculates the new energy, the if statement
    // is in case you want the 2 batteries to behave differently.
    //eg. one with discrete charging, the other continuous.
{
    if(i==0)
        //comment the first line if you don't want sensors to charge batteries
        //uncomment the second line if you want motor to spend energy in movement.
        animat.E[i] += ( ( 5 * (animat.sensor[LEFT].value[i] + animat.sensor[RIGHT].value[i])
        //- 0.01 * fabs( animat.motor[LEFT] + animat.motor[RIGHT])
        - animat.E[i]
        ) * dt) / taoE;
    else
        //comment the first line if you don't want sensors to charge batteries
        //uncomment the second line if you want motor to spend energy in movement.
        animat.E[i] += ( ( 5 * (animat.sensor[LEFT].value[i] + animat.sensor[RIGHT].value[i])
        //- 0.01 * fabs( animat.motor[LEFT] + animat.motor[RIGHT])
        - animat.E[i]
        ) * dt) / taoE;
}
/*****/
double add_noise_sensor()
    //returns a sensor angle with a small random noise
{
    return pi_1_3 + 0.05 * ( myrand() - 0.5 );
}
/*****/
void get_intensity(int side,int num)
    //This function calculates the intensity that the sensor receives
    //from the distance from the source to the sensor. REceives the side
    //(left right) and num specifies the battery type (food, water)
{
    double distance, val, dot, mag , phi;

```

```

//distance from lamp to sensor
distance = (lamp[num].x - (animat.sensor[side].x + animat.x))
          * (lamp[num].x - (animat.sensor[side].x + animat.x))
          + (lamp[num].y - (animat.sensor[side].y + animat.y))
          * (lamp[num].y - (animat.sensor[side].y + animat.y));

// dot product of normal vector and vector from sensor to lamp
dot = (animat.sensor[side].x) * (lamp[num].x - animat.sensor[side].x - animat.x )
      + (lamp[num].y - animat.sensor[side].y - animat.y ) * (animat.sensor[side].y);

//magnitud of the 2 vectors used above
mag = sqrt(distance) * R;

//angle between normal at the sensor, and the vector from lamp to sensor
phi=acos(dot/mag);

//value for intensity
val=(lamp[num].I)/distance;

if( phi > (PI/2) )
    //obstruction
    animat.sensor[side].value[num] = 0;
else if( val > 10 )
    //to high
    animat.sensor[side].value[num]= 10;
else
    animat.sensor[side].value[num] = val;
}
/*****
void plastic_changes()
    // gives new parameters to the sensorimotor connections
{
    int i,j,num;
    double factor;

    for(num=0;num<NUM_LAMPS;num++)
        //for the different types of energy sources
        {
            if (animat.E[num] < Emin || animat.E[num] > Emax)
//if it is out of bounds
{
            if(SOFT)
                factor=plastic_factor(animat.E[num]);
            else
                factor=delta_all;

            for (i=0;i<2;i++)//left-right
                for(j=(2*num);j<(2*num+2);j++)//2 first battery,second 2 battery 2
                    {

```

```

animat.connection[i][j].amplitud += (myrand() - 0.5) * deltaA * factor;
animat.connection[i][j].w += (myrand() - 0.5) * deltaW * factor;
animat.connection[i][j].phi += (myrand() - 0.5) * deltaPhi * factor;
    }
}
}
/*****/
double modulus(double x , double y)
{
    double value;

    value = ( x - ( (int)( x / y ) * y ));
    return value;
}
/*****/
double myrand()
{
    return (rand() % 100) / 100.0f;
}
/*****/
void random_lamp(int num,double x,double y)
    //initializes a lamp with random properties
{
    double angle,distance;

    angle = myrand() * 2 * PI;
    distance = myrand() * 50 + 75;

    lamp[num].x = x + distance * cos(angle);
    lamp[num].y = y + distance * sin(angle);
    lamp[num].I = myrand() * 1000 + 500;
    lamp[num].T = (myrand() * 0.5 + 0.75) * MAXT;
}
/*****/
double calculate_distance(int num)
    //distance from center of robot to a lamp, receives lamp index
{
    return sqrt((lamp[num].x - animat.x) * (lamp[num].x - animat.x)
        + (lamp[num].y - animat.y) * (lamp[num].y - animat.y));
}
/*****/
double plastic_factor(double x)
    //this is the plastic factor function
{
    if(x < E1 || x > E2)
        return 1;
    else if(x < Emin)
        return exp( ( - x + E1 ) / ( taoPmin * ( Emin - E1 ) ) );
}

```

```

else
    return exp( ( x - E2 ) / ( taoPmax * ( E2 - Emax ) ) );
}
/*****
////////// THE END !! :)//////////
*****/

```

B APPENDIX: MATLAB CODE

B.1 load_sim.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program loads the data from the simulation
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all
a=load('robot_path.dat');

teta=a(:,1); %orientation
x=a(:,2);    %x coordinate
y=a(:,3);    %y coordinate

%sensors coordinates
slx=a(:,4);
sly=a(:,5);
srx=a(:,6);
sry=a(:,7);

% Ligth sources positions
lx=a(:,8:5:end);
ly=a(:,9:5:end);

le=a(:,10:5:end); %energy levels
ld=a(:,11:5:end); %distance from sources
lf=a(:,12:5:end); %food eaten.

```

B.2 draw_circle.m

```

%Receives the center the radius and the orientation of the robot
%this function is used to draw lamps sometimes (using angle=0)

function draw_circle(x,y,angle,R)

teta=0:0.05:2*pi;

v=[R*sin(teta') + x ,R*cos(teta') + y];

```

```

plot(v(:,1),v(:,2))
hold on
line([x (R*sin(angle)+ x)], [y (R*cos(angle) +y)])

```

B.3 picture.m

```

%this program draws the arena, the robot the lamps,
%within tmin and tmax (variables which are specified
%in the program plot_energy

```

```

close all

```

```

figure(1)

```

```

plot_energy %this program plots the energy and the distance to food sources

```

```

%lamps

```

```

figure(2)

```

```

%draw LAMPS

```

```

for c=tmin:tmax

```

```

    draw_circle(lx(c,1),ly(c,1),0,3);

```

```

    hold on

```

```

    draw_circle(lx(c,2),ly(c,2),0,3);

```

```

    hold on

```

```

end

```

```

plot(lx(tmin:tmax,1),ly(tmin:tmax,1),'kx')

```

```

hold on

```

```

plot(lx(tmin:tmax,2),ly(tmin:tmax,2),'kd')

```

```

hold on

```

```

plot(x(tmin:tmax),y(tmin:tmax),'-r')

```

```

hold on

```

```

i=tmax

```

```

%plot(x(:i),y(1:i),'r')

```

```

draw_circle(x(i),y(i),teta(i),4);

```

```

plot(slx(i) + x(i) , sly(i) + y(i),'k.')

```

```

hold on

```

```

plot(srx(i) + x(i) , sry(i) + y(i),'k.')

```

```

hold on

```

```

axis equal

```

B.4 plot_energy.m

```

%this function plots the energy, the distance
% by a robot uin hte simulation
% from tmin to tmax

```

```

close all
figure(1)

tmin=15000;
tmax=15010;

subplot(2,1,1)
plot(tmin:tmax,le(tmin:tmax,1))
hold on
plot(tmin:tmax,le(tmin:tmax,2),'g')
hold on
%plot([1 length(le)], [3 3],'k-.')
plot([tmin tmax], [10 10],'k-.')

hold on
%plot([1 length(le)], [15 15],'k-.')
plot([tmin tmax], [40 40],'k-.')
%hold on
%plot([50000 50000], [0 20],'k-.')

xlabel('Time')
ylabel('Energy')

subplot(2,1,2)
plot(tmin:tmax,ld(tmin:tmax,1))
hold on
plot(tmin:tmax,ld(tmin:tmax,2),'g')
%hold on
%plot([50000 50000], [0 600],'k-.')

xlabel('Time')
ylabel('Distance from source')

```

References

- [1] Ashby, W. R. (1960). *DEsign for a Brain: The Origin of Adaptive Behaviour*. London: Chapman and Hall.
- [2] Braitenberg, V. (1984). *Vehicles: experiments in synthetic psychology*. Cambridge, MA: MIT Press.
- [3] Di Paolo, E. A., (2000). Homeostatic adaptation to inversion of the visual field and other sensorimotor disruptions. Proc. of SAB'2000, MIT Press.
- [4] Di Paolo, E. A., (2003). Organismically-inspired Robotics: homeostatic adaptation and teleology beyond the closed sensorimotor loop.
- [5] Held, R. (1965), Plasticity in sensory-motor systems, *Scientific American*, November 1965.
- [6] Kohler, I. (1962), Experiments with goggles, *Scientific American*, May 1962.
- [7] Maes, P. (1991) A bottom up mechanism for behavioural selection in an artificial creature. Proc. SAB 90, pp 238–246
- [8] Seth, A. (1998) Evolving action selection and selective attention without actions, attention or selection. Proc SAB 5, pp 139-147
- [9] Taylor, J. G. (1962). *The Behaviour Basis of Perception*. New Haven: Yale University Press.
- [10] Tyrel, T. (1993). *Computational Mechanisms for Action Selection*. Phd Thesis, University of Edinburgh.