Remembering the past: recipe for the ultimate survivor?

The value of multiple timescales in a recurrent neural network for self-organization of survival behavior in random versus structured environments

Lysanne Sloff (0815411) - December 20, 2011

Bachelor's thesis
Author: Lysanne Sloff
E-mail: L.Sloff@student.ru.nl
Student number: 0815411
Supervisors: Ida Sprinkhuizen-Kuyper & Pim Haselager
Radboud University Nijmegen
December 20, 2011

Remembering the past: recipe for the ultimate survivor?

The value of multiple timescales in a recurrent neural network for self-organization of survival behavior in random versus structured environments

Lysanne Sloff
Department of Artificial Intelligence, Radboud University Nijmegen
December 20, 2011

Abstract

It is generally thought in cognitive neuroscience that the concept of functional hierarchy - the notion that complex things can be decomposed into simpler elements and that simpler elements make up a complex system - plays an important role in the production of skilled (motor) behavior and situations that require cognitive control. According to schema theory, behavioral elements make up behavioral primitives, which can be sequenced to achieve a global goal. In robotics, there have been many different attempts to design paradigms for such behavior productions but often a distinction is made between reactive and deliberative robots. Hybrid systems incorporate both kind of behaviors, in which a higher level system controls lower level reactive layers to produce behavior (e.g. the traffic regulator concept). Since it is not really clear how such functional hierarchy is actually organized in the brain, it would be interesting to see how this functional hierarchy can self-organize. In the current thesis, a recurrent neural network model was used for such self-organization. Context units with different multiple timescales were used, to incorporate the temporal organization of behavior. The goal was to test how well such an MTRNN agent performed and what kind of behavior was shown as compared to a traffic regulator on a survival task in a day-night environment, with obstacles and food sources. Furthermore, since hybrid robots are consistent with embodied embedded cognition, it would be interesting to see what kind of role environment type plays for the behavior of the MTRNN agent. Therefore the behavior and performance of the MTRNN was tested in two different environments, varying in the amount of structure. It was found that the MTRNN agent performed worse than the other tested agents, but that performance was better in more structured environments. This implicated that the environment is an important factor but that the MTRNN agent is less suited to random environments. As for the self-organization of functional hierarchy, it did not emerge through the use of different timescales but the complexity of behavior was dependent on the right amount of food in the environment. The results indicated that in order to achieve functional hierarchy and perform well, the agent needs clear goal-directed tasks and structured environments.

Keywords: functional hierarchy, skilled behavior, reactive, deliberative and hybrid robotics, embodied embedded cognition, recurrent neural network, multiple timescales

1 Introduction

Every day humans encounter many situations in which difficult decisions must be made, complex tasks must be completed and appropriate behavior in different contexts is required to reach certain goals and in the end, survive through the day. In order to do that, humans (and other organisms as well) perform skilled behavior - such as speech production, planning, reasoning, a well as complex motor patterns. In cognitive neuroscience, it is widely believed that such skilled behavior is possible because the brain is a system with functional hierarchy (Braver, Paxton, Locke, & Barch, 2009; Botvinick, 2008; Ardila, 2008). The concept of functional hierarchy in cognitive neuroscience can be defined as the notion that complex systems can be decomposed into more simple, primitive elements and, the other way around, these simple elements can be integrated to make up a more complex system. The concept of functional hierarchy is one that is encountered a lot within the field of cognitive neuroscience. A good example is the motor behavioral production system because it can be seen as a system with a functional hierarchical organization; motor elements (such as moving left, turn 50 degrees, etc.) are integrated to compose a behavioral primitive - a set of motor elements to reach a certain sub-goal. Such behavior primitives can then be reused to make up a sequence of motor primitives to reach a certain global goal. Agents with such functional hierarchy can thus adapt very well to different situations. This idea has been expressed in the concept of (motor) schema theory (Schmidt, 1975; Arbib, Erdi, & Szentagotha, 1988), in which different primitives make up a schema in order to reach goals.

An important concept when it comes to skilled behavior and its adaptive properties is cognitive control - the ability to act accordingly to internal goals and the current perceptual context (Braver et al., 2009; Koechlin, Ody, & Kouneiher, 2003; Badre & Wagner, 2007; Badre, 2008; Egner, 2009). This means that different behavior is required in situations that are perceived the same but in which the context is different. Therefore, in order to perform skilled behavior, an agent must be able to differentiate such situations and choose and sequence the appropriate behavior primitives to handle the current context. The context therefore plays an important role in the way behavioral primitives in motor schema theory are sequenced.

In robotics, different approaches have been pursued to model behavior production. Within this field of research, often a distinction is made between reactive behavior on the one hand and deliberative behavior (such as planning, reasoning and strategy-driven tasks) on the other hand (Murphy, 2000). The reactive paradigm (Brooks, 1986, 1991) deviates from the belief that behavior should be decomposed into functions (in which output goes from one functional module to the other until the final output is produced) and instead is based on a more vertical decomposition of behavior into different activities that all have their own goal. To be more specific, each layer has its own goal, takes its own input and reacts without intermediate processing of information. The layers are organized in a vertical manner and higher level behavior layers can inhibit or suppress lower layers, thereby forming a subsumption architecture. Through this paradigm, an agent performs different combinations of behaviors, can react very fast to changes in the environment and is therefore very flexible.

The reactive paradigm arises from the concept of embodied embodied cognition; the brain

should not be seen as the central control system, but instead cognition arises through the interaction between the agent's body, it's brain and the environment (Brooks, 1991; Haselager, Dijk, & Rooij, 2008; Van Dijk, Kerkhofs, Van Rooij, & Haselager, 2008). This is exactly what the reactive paradigm does: it uses the world as its model. This is consistent with the work by Willems and Haselager (2003); they showed that the emergence of cooperative and strategic behavior was dependent on the nature of the environment, which indicates the role environment can play in behavior production.

Although reactive agents work very well in many situations, in more complex situations also some deliberative behavior and cognitive control is required to also maintain a global goal. The hybrid robotic paradigm incorporates both reactive and deliberative behavior and such an agent is thus able to perform well in local situations while sequencing behavior primitives to maintain a global goal as well (Aaron & Admoni, 2010; Huq, Mann, & Gosine, 2008; Peterson, Duffy, & Hooper, 2011). The traffic facilitator (Haselager et al., 2008; Van Dijk et al., 2008) is a nice concept that illustrates how a hybrid system can incorporate functional hierarchy and be consistent with embodied embedded cognition: at the lower level the behavioral layers produce behavior primitives according to the current perceptual input, while at a higher level a control system inhibits certain layers to maintain a global goal. The higher control structure thus decides which behavior primitives become active and are thus performed. This notion is consistent with the general belief that the prefrontal cortex (PFC) plays an important role in action selection, task sequencing and cognitive control and the literature covering this topic is almost endless (Braver et al., 2009; Ardila, 2008; Badre & Wagner, 2007; Badre, 2008; Botvinick, 2008; Petrides, 2005; Egner, 2009; Koechlin et al., 2003; Fuster, 2001). The PFC basically acts like the higher control system that controls lower level systems, again stressing the idea of the functional hierarchical organization of the brain.

The effectiveness of the traffic facilitator/regulator was shown in a study by Lagarde (2009). He compared reactive agents and control agents (based on the hybrid traffic facilitator) in an environment that had a day-night rhythm. All the agents had to survive as long as possible by searching for food, avoiding obstacles and go to sleep at appropriate times since sleep preserves energy (Berger & Phillips, 1995). In the control agents, the reactive layers could all individually be inhibited by a higher control structure (a multilayered perceptron). It was shown that the control agents were able to develop a day-night rhythm without this rhythm being hardcoded and this enabled these agents to outperform the reactive agent without hardcoded sleeping behavior and performed equal to the agent that had a hardcoded sleep rhythm.

Although the literature seems to indicate that behavior and cognition has a functional hierarchy, it is not really clear how this functional hierarchy is actually organized in the brain despite all the hybrid models out there. The work by Yamashita and Tani (2008) and Paine and Tani (2005) showed that functional hierarchy can self-organize in neural network models without constraints on how this functional hierarchy is structured in the architecture. Instead, both models include temporal aspects which is consistent with the notion that many behavior depends on time as well (Fuster, 2001; Rajah, Ames, & D'Esposito, 2008; Smith,

Ghazizadeh, & Shadmehr, 2006; Montebelli, Herrera, & Ziemke, 2008; Kiebel, Daunizeau, & Friston, 2008). Although the two studies included similar constraints and had similar goals, different network models were used. In Paine and Tani (2005), a network model with a bottleneck architecture was created that incorporated topological constraints. These constraints ensured that different network parts developed responsibility for different parts of the robot's behavior; one part developed fast dynamics producing behavior primitives, while the higher part developed slow dynamics, thereby keeping a global goal in mind and sequencing the behavior primitives. In this particular study, different timescales evolved indicating that these different parts do not work equally fast. In the second study (Yamashita & Tani, 2008), a kind of recurrent neural network was proposed in which functional hierarchy self-organized through the use of two different context units, each with their own time properties. The context units with a fast timescale became involved in the generation of behavior primitives, while the context units with a slow timescale were responsible for sequencing behavior primitives to achieve a global goal. The different timescales ensured that the units' activity is not only influenced by the current input, but also by previous time states; in the fast context units the activity is dependent on less previous states than in the slow context units, acting like a short- and long-term memory.

Both the traffic facilitator and the work of Yamashita and Tani (2008) produce agents with a functional hierarchy to produce behavior, both in their own way. This traffic facilitator was already shown to be effective in the day-night environment of Lagarde (2009) and therefore it would be very interesting to see how well the agent with a model like that of Yamashita and Tani (2008), a recurrent neural network with multiple timescales - called MTRNN from this point on - is able to perform, i.e. survive as long as possible, in the same environment. Therefore the first research question of the current thesis is:

(1) How well will the MTRNN agent survive as compared to reactive and hybrid agents in a random day-night environment, such as proposed in Lagarde(2009)?

To answer this question, the MTRNN agent will perform the same task in the same environment as the reactive (Reactive & Reactive-DN agents) and control agents (Control agent) of Lagarde (2009) and with two agents with simpler network structures, a perceptron and multi-layered perceptron (called Perceptron and MLP agent, respectively).

Furthermore, it seems that the kind of environment plays an important role when it comes to development of behavior structures. This indicates that it would also be very interesting to know more about what kind of behavior the MTRNN agent will show in different environments and also to what kind of environments the MTRNN is best adapted. This proposes two other research questions which will be investigated here:

(2) What kind of behavior and performance will the MTRNN agent show in random (such as proposed by Lagarde (2009)) as compared to more structured environments?
(3) In what kind of environments and tasks will the MTRNN agent be able to benefit (show effective survival behavior) from having multiple timescales, i.e. remember previous time

states?

Question (2) will be answered by observing all six agents perfom the survival task in two different environments. The first environment will be a random environment with food and obstacles and is actually the same as used by Lagarde (2009). The other is a more structured environment that incorporates no obstacles but cues to where the food is positioned. It is expected that the MTRNN agent is able to benefit of this structure, by memorizing patterns. The third question is answered by looking at the results of questions (1) and (2), but also at the type of environments and tasks that were used in the work of Yamashita and Tani (2008) and Paine and Tani (2005).

Finally, since the functional hierarchy should be self-organized in the model that is currently investigated, it is also necessary to investigate if this is also really the case. The fourth and final research question is therefore:

(4) Is the agent able to achieve functional hierarchy through different timescales in the current survival task and environments?

This final question is answered by observing and comparing the behavior of the MTRNN agent against the behavior of the MLP and Perceptron agents in both the environments; if a sequence of different behavior primitives and patterns (such as a distinction between day and night) is observed, it is likely that the agent uses a functional hierarchy of behavior to produce and control behavior. However, if the same kind of behavior is observed in the simpler network agents as well, the MTRNN agent may still have developed functional hierarchy but this then would not be due to the specific neural network architecture and use of different timescales.

In the following sections, first the methods to answer the various research questions are explained. Second, the results of the various simulations and comparisons are described and evaluated to finally draw a conclusion and discuss the many possibilities of future research.

2 Method

In this section is explained how the experiments to answer the four research questions are set up. The experiment includes six different agents: Reactive, Reactive-DN, Control, MTRNN, Perceptron and MLP agents. The characteristics of each agent will be described, as well as the training methods for the Control, MTRNN, Perceptron and MLP agents. Furthermore, the task, environments and simulations that were used are explained in detail. Finally, a short overview of alternative training strategies will be given. These strategies were tested in the same task setting but did not improve results so were eventually not further used in the experiments, but are still worth noting.

The task, the random type environment (see Section 2.2.1) and the Reactive, Reactive-DN and Control agent were first used in the work of Lagarde (2009) and later in Bax (2010).

In the current experiment also a more structured environment was used and the behavior of three new agents (the Perceptron, MLP and MTRNN agent) was evaluated. Environments can obtain food sources, obstacles and sign posts. The latter object is a sort of pointer that indicates that a food source is somewhere near in the environment.

2.1 The task setting

For every type of agents holds that it should perform the same task in every type of environment. The agents have to survive as many time steps as possible in a world with randomly placed food sources and obstacles. In order to survive in this context, the agents must try to keep their energy level above zero as long as possible. Naturally, consuming food will result in energy gain and walking into an obstacle will result in energy loss. Also, if the agent moves it will lose some of its energy but resting also costs energy, although of course not as much as moving. Thus, the agent can keep its energy level as high as possible by searching for food, avoiding obstacles or resting at appropriate times, dependent on the current states. Whether or not the agents will in fact be able to survive successfully for quite some time depends on the type of system (or paradigm) the agent uses to produce and control behavior.

2.2 The simulation environments

Since one of the goals of the current experiment is to find out more about the behavior of the MTRNN controlling an agent in different environments, two types of environments were used in the simulations.

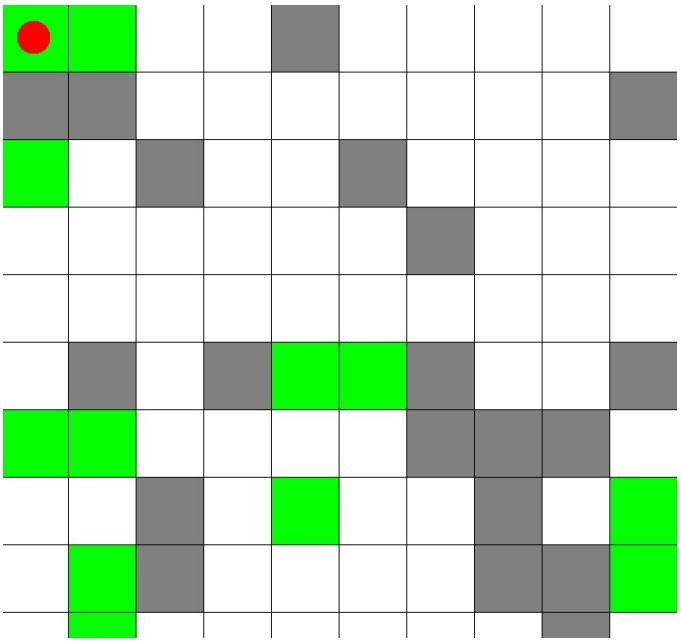
Each environment consists of hundred grid cells forming a two-dimensional ten by ten grid. At the start of a simulation, the agent is always placed at the same grid cell (coordinates (0,0)) or as near as possible if this grid cell is already occupied by another object. Food sources are the only objects that are used in both simulation environments. Agents can consume food by stepping on a cell containing a food source. Consuming food will result in an energy gain of 10.

All environments have the same build-in day and night rhythm; one day takes 30 steps of which half is in day conditions and the other in night conditions. Each agent is able to move to one of the adjacent cells of the grid cell the agent is currently standing on. Furthermore, the environment has no boundaries so if an agent seems to walk off the edge of the environment, the agent will reappear at the other side. Therefore the environment does not seem to be flat and simulates a torus, a 3-dimensional donut-shaped world.

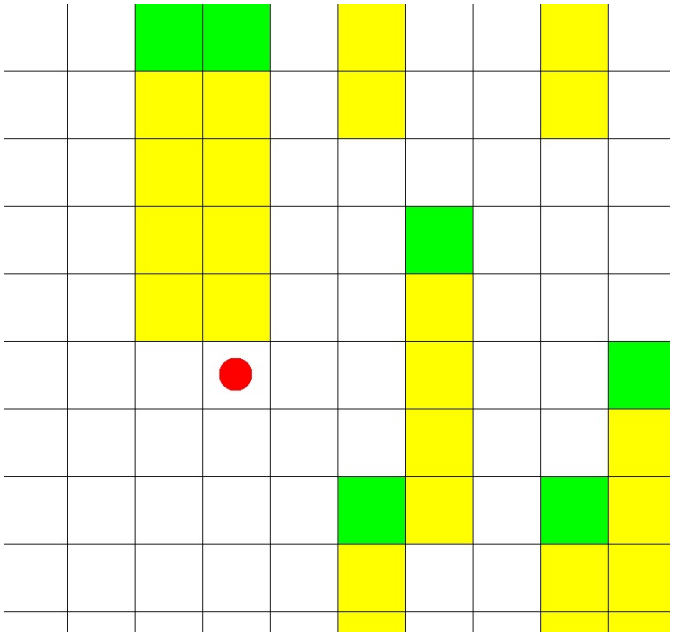
The two different types of environments are now described in more detail.

2.2.1 Random environment

An example of a random environment as used in the current experiment is depicted in Figure 1(a). In this specific type of world a cell can be occupied by empty ground or an object, which is either a food source or an obstacle. A set number of obstacles and food sources are randomly distributed across the grid so that the environment varies between simulations.



(a) Example of random environment



(b) Example of structured environment

Figure 1: *Examples of the two different types of environments. In the upper figure, dark gray cells are obstacles, the lighter grey or green cells are food sources, white grid cells are empty ground cells and the circle is the agent. In the lower figure, the grey or green cells are food sources, the yellow or light grey cells are food indicators or sign posts, white cells represent empty ground and the circle is the agent*

If food is consumed in this random environment, the food disappears and reappears at a random place somewhere else in the grid. This ensures that the agent does not linger around the same spot and that the amount of food remains constant during a simulation. Obstacles, however, remain where they are during a simulation and their locations only differ between simulations. Obstacles can be seen like quicksand or little swamps; it is possible to walk through them but the agent's energy level will then be decreased by 15.

This random environment does not require the agent to remember previous steps, because each world is different from the next and there are no specific cues there to indicate the current situation. Decisions can thus be made based on the current state the agent is in and not let previous steps influence that decision, since previous steps are independent from the current situation.

2.2.2 Structured environment

The second type of environment is, in contrast to the random environment described before, more structured. To be more specific, the environment includes several cues that indicate that food lies ahead. Therefore it might be useful for an agent to remember some of the previous steps, because now the previous situations can say something about the current situation (namely situations where the agent consumes food and has visited a sign post in previous steps).

A cell can be empty or occupied by an object, which is either a food source or a food indicator. In the current experiment, such a food indicator is called a sign post and is so called because agents might learn that such an object points the agent into the direction of a food source. Figure 1(b) shows an example structured environment. As can be seen in this figure, more structure is created by forming vertical strips of four sign posts and at the top a food source is placed. The presence of such a sign post thus indicates that above lies a food source, so it is expected that the MTRNN agent will move up as soon as it detects a sign post.

If food is consumed, the whole vertical strip of sign posts and food sources will disappear and is relocated to another point in the environment. This replacement is not so random as in the first environment however; in every simulation in this structured environment, a grid column can only contain one vertical strip of sign posts and food and a food source can only be replaced in an empty column (including the column it was just consumed in). This also means that if there are ten food sources, there will be a food source in every column during the whole simulation. Another possibility is to only replace the food source after food consumption, but the problem then is that the agent may be misled by sign posts that falsely indicate that food lies ahead.

In contrast to obstacles, sign posts do not cost the agent energy on top of the cost of moving or resting since then the agents that learn to follow the sign posts would be punished. This would result in the agents avoiding the sign posts and therefore also the food source, which is the opposite of the task goal.

2.2.3 Perceptual range

Each type of cell has its own unique pattern of light reflection, while empty ground does not reflect any light. Thus agents are able to distinguish between obstacles, food sources, sign posts and empty ground cells. At daylight, this classification of grid cells is flawless. However, in night conditions the light reflections of the various grid cells become less clear and thus the classification between objects less accurate. The image the agent than has of its surrounding is not always correct, so the agent receives very uncertain and fuzzy information. The information about the cell the agent is currently standing on is not fuzzy and is always accurate at both day and night.

Although food, obstacles and sign posts can be classified correctly at daytime, the range in which food can be perceived differs from the range in which obstacles and sign posts can be perceived. Food is always perceived within a range of Manhattan Distance 2, while ground cells, obstacles and sign posts are only perceived within a range of Manhattan Distance 1. This means that food can be perceived behind obstacles and sign posts and therefore the agent actually has two different ways of perceiving the current surrounding: with vision and with smell. The agent is able to distinguish between food, obstacles, sign posts and ground through vision (at Manhattan Distance 1) and distinguishes food from other types of cells through smell (at Manhattan Distance 2).

It must be noted that due to the poor light reflection at night, food is not correctly classified within both smell and visual range (although that is not something you would expect in real life).

2.2.4 Discrete environments

Both random as structured environments are discrete, so time passes in discrete steps. Furthermore, the environments are quite simple in the sense that grid cells can only be one of three different types (in any type of environment) and the agent can only move in a small number of directions, so only a small number of actions can be performed. This and the discrete time factor make it possible that the agent can perform a discrete action every time step and it does not matter that subsequent actions are very different from each other (e.g. it does not matter if the agent goes up one time step and the next step in a completely different direction).

2.3 Agents

In the current experiment, six different agents will be tested on their performance on the task and in both the random as the structured environment. Although all agents use a different kind of control architecture or behavioral paradigm, they do have some characteristics in common; each agent starts out with an energy level of 250, which is also the maximum energy level that can be achieved. Furthermore, moving always costs energy as well as resting but the energy costs for these actions differ between agent types which corresponds with the relation between energy preservation and sleep of organisms (Berger & Phillips, 1995). Each

Parameter	Value
Environment width	10
Environment height	10
Energy loss for obstacle	15
Energy gain for food source	10
Energy gain/loss for sign post	0
Number of obstacles	if world I: 0 to 40 obstacles, if world II: 0
Number of food sources	if world I: 0 to 40, if world II: 0 to 10
Number of sign posts	if world I: 0, if world II: food x 4
Start position of agents	(0,0) or as close as possible

Table 1: Summary of different environment parameters

agent has a motor system that allows it to turn and move in 8 different directions (UP, UP RIGHT, RIGHT, DOWN RIGHT, DOWN, DOWN LEFT, LEFT, UP LEFT) or stand still at each time step. Furthermore, the sensory system of the agents includes a light sensor, to see ground, objects, food and sign posts and smell food within the corresponding perceptual ranges.

In his bachelor's thesis, Lagarde (2009) tested the performance of four agents, which could be divided in reactive types and control types. The reactive types behaved according to the reactive paradigm with behavioral layers (Brooks, 1986, 1991; Murphy, 2000) but in the control systems, a higher control structure is present that can inhibit the lower level behavioral layers when necessary. In the current experiment, the second type of control agent is omitted from testing (why is explained below) but three new agents (on top of the two reactive agents and the control agent) will be tested. In the latter three types, the whole behavioral structure is replaced by a neural network. The first of these types uses a recurrent neural network with multiple timescales (MTRNN) as described by Yamashita and Tani (2008), with both slow context and fast context units. The second type uses a simple perceptron and the third a multilayered perceptron, to compare the MTRNN to simpler net structures used as a system to produce behavior.

To give an indication of the performance of the three neural network controlled agent types in this experiment, their performance will be compared to the performance of the three types already assessed before by Lagarde (2009). Overall, the following agents were used and tested for their performance in the current experiment:

- **Reactive:** An agent producing behavior following the reactive paradigm (see below) and without a higher control structure.
- **Reactive-DN:** A reactive agent with a build-in day and night rhythm and no higher control structure.
- **Control:** An agent with a multilayered perceptron as a higher control structure that inhibits several behavioral layers of the reactive system when necessary.

- **MTRNN:** An agent that uses a recurrent neural network with multiple timescales as a structure to produce behavior and predicts the best next motor action according to current sensory and motor information. Furthermore, the net uses fast context units and slow context units (hence the multiple timescales) that serve as a memory.
- **Perceptron:** An agent that uses sensory inputs to learn the next appropriate motor action.
- **MLP:** An agent that also uses sensory inputs to learn the next appropriate motor action, but uses a hidden layer between input and output.

The Control-2 agent (Lagarde, 2009) is not tested in this environment, because it only differs from the Control agent in the sense that additional costs are added for every output link of the higher control structure that inhibits a behavioral layer. Since the goal of the current experiment is to compare different functional hierarchical structures in this specific setting, comparing the MTRNN agent with the Control agent will suffice.

2.3.1 Reactive and Reactive-DN agents

Reactive agents produce behavior according to the reactive paradigm, a paradigm with a sense-act organization; the agent senses (part of) its environment and based on this sensory information an action is immediately produced without extra intermediary processing or planning (Brooks, 1986, 1991; Murphy, 2000). The paradigm consists of one or more of these sense-act coupling or behavioral layers, structured in default and higher-level behavior. If higher level layers are on, they may inhibit lower-level layers but still more than one layer can be active at the same time. Through this paradigm, the agent will produce emergent behavior; a combination of output of several behavioral layers.

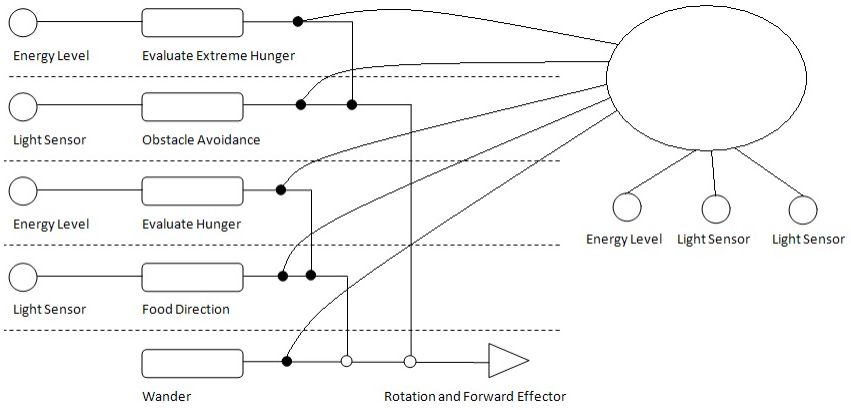


Figure 2: *The behavioral layers of the Reactive, Reactive-DN and Control agent and the higher control structure of the Control agent (Lagarde, 2009)*

Figure 2 shows the organization of the layers as also used and described in Lagarde (2009) and Bax (2010). Each layer is briefly described:

- The lowest layer is the *Wander layer*. Because this is the default layer, it takes no input but only produces motor output. It moves the robot in a random direction or randomly decides to rest at the same spot.

- On top of the Wander layer is the *Food Direction layer*, which takes as input its surroundings at that time step. Based on this information, it determines whether or not there is food within range and if there is, determines the location of a food source and activates the wander layer to turn in the required direction.

- The third layer is the *Evaluate Hunger layer*. This layer inhibits its lower level, the Food Direction layer, when the agent's energy level is higher than a certain hunger threshold. If the energy level becomes lower, the Food Direction layer is not inhibited anymore and thus the agent is able to search for food.

- The next layer is the *Obstacle Avoidance layer*, which takes as input the current situation (existing of the four cells that are directly adjacent to the cell the agent is currently standing on) and tries to detect whether or not there are no obstacles in its surrounding. If the agent detects obstacles, it wanders in a direction where the agent sees none.

- If the agent is really hungry, i.e. the agent's energy level becomes lower than a certain extreme hunger threshold, the *Evaluate Extreme Hunger layer* inhibits its lower layer which is the Obstacle Avoidance layer. This means that the agent is then able to move through an obstacle to reach food.

Both the Reactive and Reactive-DN agent use this paradigm. However, because the Reactive agent does not have a day and night rhythm, it will try to behave at night the same as in daylight. It will thus make more mistakes, since at night the sensory information is inaccurate. The Reactive-DN agent does have a build-in day night rhythm, so the agent will rest at night to preserve as much energy as possible and not make costly mistakes.

The costs for moving and resting are not equal for these two agents: moving costs Reactive agent 2 and resting 1, while the Reactive-DN loses 3 when moving and 2 when resting.

2.3.2 Control agent

The control agents use the behavioral reactive layers as well, but differ from the reactive agents in that there is higher control structure that is able to inhibit every reactive layer. To be more specific, this higher control structure is a multilayered perceptron which has an inhibitory output link to every reactive layer. Therefore there are five output units of the MLP. Which behavioral layers are inhibited, depends on the sensory information the network extracts from its environment. The sensory information is based on four sensory inputs; (1) the energy level, (2) the log-likelihood of the agent's surrounding, (3) whether or not the agent stands on an obstacle and (4) whether or not the agent is standing on a sign post cell. The fourth input was not used in Lagarde (2009), but was especially added to the MLP for the Control agent to be able to handle the structured environment as well. The

first, third and fourth input are internal information, so the input values are always accurate. The second input, the log-likelihood of the current surrounding, is external information and it depends on the time of day what this value is: at day this value is always 0, but at night the visibility values of all the objects become unreliable and thus the log-likelihood of the current surrounding will then not be equal to 0. Between the input and output layer is a hidden layer consisting of six hidden units. The input and hidden layer are fully connected, as well as the hidden and output layer. Each node layer has its own activation function:

- Input nodes use the identity function
- Hidden nodes use the hyperbolic tangent (tanh)
- Output nodes use the logistic sigmoid function

The output values lie between 0 and 1 and represent the probabilities of whether or not the inhibitory link to a behavioral layer is activated. The weights between the layers evolve according to an evolutionary algorithm (see Section 2.4).

Just as the Reactive-DN agent, the Control agent loses energy when moving (energy decrement of 3) and a little less when the agent rests (decrement of 2).

2.3.3 MTRNN agent

The MTRNN agent uses a structure that supports the same functional hierarchy of organization as the control agents, but by using a different way of producing behavior, namely through a multiple timescale neural network. The network used here is different from the network developed by Yamashita and Tani (2008) in the way that it is adapted to the current task and environments. However, the architecture is mainly the same and the current network still covers the aspect of multiple timescales and uses the same kind of network nodes. This type of agents has the same energy decrements as the Control and Reactive-DN agents: moving costs 3, while resting costs 2.

In Figure 3 the structure of this MTRNN is shown. As can be seen, the network consists of three different parts:

1. The input-output part that interacts directly with the environment by extracting sensory and motor input from the agent's surrounding and outputting the best next motor action;
2. The fast context units that serve as a short-term memory and are influenced by the actions of a number of previous time steps;
3. The slow context units that serve as a sort of long-term memory and are influenced by more previous steps than the fast context units.

Below each of these separate parts are described in more detail and also the interaction between the different network parts is explained.

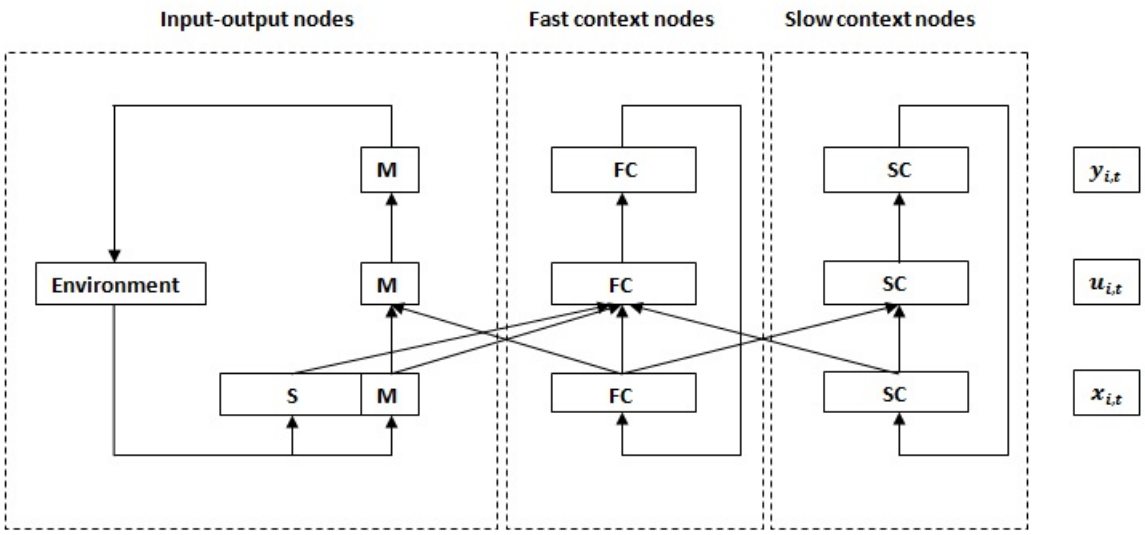


Figure 3: *Structure of the recurrent neural network with multiple timescales as used for the MTRNN agent. M = motor nodes, S = sense nodes, FC = fast context nodes, SC = slow context nodes,* $x_{i,t}$ *is the input activation for unit i,* $u_{i,t}$ *is the membrane potential for unit i and* $y_{i,t}$ *is the output of unit i*

Input-Output layer

The left most part is the input-output layer. This is the part of the network that interacts directly with the environment. The MTRNN takes 24 different sensory inputs and 9 motor inputs. The motor input is very simple: each input corresponds to one of the nine possible directions (up, up right, right, down right, down, down left, left, up left and center/stay put). If a direction is the direction the agent has moved to in the previous time step, the corresponding input is 1, otherwise the input value is 0. So for example, if the agent has moved up, the motor input is the vector (1 0 0 0 0 0 0 0 0).

The sensory module of the MTRNN is somewhat more complex, because the network does not use behavioral layers that react on sensory inputs. Therefore the input to the MTRNN has to be very elaborate to give this agent all the necessary information the reactive and control agents receive through their behavioral layers. The 24 different sensory inputs are:

- The first and second input are the log-likelihood of the surrounding and the type of cell the agent is currently standing on (the same as in the Control agent)
- The third and fourth input represent the hunger and extreme hunger thresholds: input three is 1 if the agent's energy level drops below 240 and 0 otherwise; input four is 0 if the energy level is above 40, but 1 if the energy level drops below 20. For energy levels between 20 and 40, input four increases linearly from 0 to 1 as the energy becomes lower.

- The fourth to eighth input represent the direct surrounding of the agent, i.e. all cells that lie at Manhattan Distance 1 from the agent. This sensory input represents the vision of the agent. The agent discriminates obstacles on the one hand and food and ground on the other hand, so that the agent can handle environments with food sources and obstacles (like the random environment).

- The ninth to twelfth input also represents the direct surrounding of the agent, but now a distinction is made between sign posts and other cells so that this agent can handle environments with sign posts and food sources (like the structured environment).

- The thirteenth to twenty-fourth input represent the surrounding within range of Manhattan Distance 2 (this includes cells at Manhattan Distance 1). The inputs represent the smell of the agent and the agent discriminates between food on the one hand and other type of grid cells on the other.

This agent also does not perceive surrounding grid cells well at night. Since the input-output units should not be affected by previous states, except indirectly through the context units, the timescale for these units is 1.

Fast context and slow context layer
The multiple timescales in the MTRNN are achieved by using two different kind of context units: fast context units and slow context units. The context units serve as a short-term and long-term memory, respectively. The context units do not interact with the environment but instead receive as input the output of the corresponding context units at the previous time step. In the current experiment, fifteen fast context units and two slow context units are used. The slow context units are set to an initial value when it becomes night and when day starts, in order to let the slow context units learn different behavior for night and day (a characteristic also used in Yamashita and Tani (2008)). Input and output values lie between 0 and 1.

Naturally, the two types of context units have different timescales: the fast context units have timescale 2, which means that the fast context states are influenced by the previous state; the slow context units have timescale 10 and thus their states are affected by the nine previous states.

Interaction and activation within the network
Arrows in Figure 3 indicate weight layers between the different network parts, i.e. which parts of the network interact with each other. It shows that sensory input and motor input are not directly linked to each other; instead interaction between these two parts happens indirectly through the fast and slow context units.

- Fast context units use weighted input from sense units, motor units, fast context units (including itself) and from slow context units to calculate their activation.

- Slow context units receive weighted input from slow context units (including itself) and fast context units to calculate their activation.

- Motor units use weighted input motor units (including itself) and fast context units to calculate their activation.
- Since the sense output is not necessary for the agent to calculate the best next action, the activations of the sense units are not calculated. This means that there are no weights from sense units and fast context units to the sense units (as is the case with motor units), since those weights are obsolete.

For all kinds of units, inputs $x_{i,t}$ for unit i at time t are values between 0 and 1. Membrane potentials $u_{i,t+1}$ at the next time step for all units are calculated according to the following formula:

$$u_{i,t+1} = (1 - \frac{1}{\tau_i}) \cdot u_{i,t} + \frac{1}{\tau_i} \cdot \left[\sum_{j \in N} w_{ij} x_{j,t} \right] \tag{1}$$

where w_{ij} is the weight from unit j to unit i, $x_{j,t}$ the input for unit i at time step t and τ_i the time constant for that particular kind of unit, which basically represents the timescale for that unit (e.g. τ_i for fast context units is 2). The activations $y_{i,t}$ for sense and motor units are calculated with the following formula:

$$y_{i,t} = \frac{\exp u_{i,t}}{\sum_{j \in Z} \exp u_{j,t}} \tag{2}$$

where Z are motor **or** sense units. Activations for the context units are calculated according to the conventional sigmoid function:

$$y_{i,t} = \frac{1}{1 + e^{-x}} \tag{3}$$

The activation formulas ensure that all outputs are also values between 0 and 1.

2.3.4 Perceptron and MLP agents

Although the MTRNN agent and Control agent both use a kind of architecture in which functional hierarchy is present, the way behavior is produced is quite different between the two agent types. Therefore the MTRNN agent is also compared with two other agents using a neural network structure to interact with the environment: the Perceptron and MLP agent.

The first agent uses a perceptron network to produce the best next motor action, as the name also implies. The second agent uses a multilayered perceptron to produce motor behavior. Both networks take as input the same sensory input as the MTRNN agent but the motor input is omitted, thus therefore the input consists of 24 nodes. Furthermore, the output of both networks are the probabilities for the nine different directions (including resting at the same spot), just as is the case with the MTRNN agent. The direction that has the highest probability will be the direction in which the agents will move. Naturally, the difference between the Perceptron and MLP agent is that the latter has a hidden layer consisting of 15 nodes between input and output layer.

Parameter	Value
Types of agents	Reactive, Reactive-DN, Control, MTRNN, Perceptron, MLP
Maximum energy level	250
Costs of moving/stand still	Reactive: -2/-1, Reactive-DN, Control, MTRNN, Perceptron, MLP: -3/-2
Visual range	Manhattan Distance 1
Smell range	Manhattan Distance 2

Table 2: Summary of the different agent parameters

The same formulas for updating of the membrane potentials and calculating activations of the MTRNN agent are used for both other network agents, but since no context units are used, the timescale will always be set to 1. This means that the formulas can be rewritten as follows:

- Updating of membrane potentials for unit i at time step t: $u_{i,t+1} = \sum_{j \in N} w_{ij} \cdot x_{j,t}$, where w_{ij} is the weight from unit j to unit i and $x_{j,t}$ is the input for unit j at time step t.
- Activations for motor output unit j at time step t: $y_{i,t} = \frac{\exp u_{i,t}}{\sum_{j \in Z} \exp u_{i,t}}$ where Z are motor units.

The Perceptron and MLP agents have the same energy decrements for movement and resting, namely 3 and 2, respectively.

2.4 Evolution and training

In the current experiment, an evolutionary algorithm is used to evolve optimal parameters for the network. The parameters of interest are the weights between the different types of network units. For the MTRNN holds that also the initial states of the slow context units for both day and night are evolved to find the optimal setting to define difference between day and night behavior. Since the structure for the network is relatively clear and fixed, the evolutionary algorithm is not used to find the optimal network structure. The algorithm is implemented using the JGAP package (Rostan, 2009). The Control, MTRNN, Perceptron and MLP agents all use evolution to find their optimal parameters.

In the evolution process, chromosomes are created in which the genes represent the different weights (and in the case of the MTRNN also the initial slow context states). For the MTRNN agent there are 24 sense units, 9 motor units, 5 fast context units and 2 slow context units which means there are 1000 weights and 4 (2x2 nodes) initial slow context states. For the Perceptron agent there are 216 weights (24 inputs and 9 outputs) and for the MLP agent there are 495 weights (24 inputs, 15 hidden nodes and 9 outputs). Finally, for the Control agent there are 63 weights (4 inputs + 1 bias node, 6 hidden + 1 bias node and 5 outputs). It was mentioned in Lagarde (2009) that adding bias nodes did not make

any difference to the performance of the agent. It was also tested if adding bias nodes to the MTRNN would improve the performance of this agent, but as this was also not the case, bias nodes were not used in evolution of the MTRNN.

In Lagarde (2009) the search space of weights of the Control agent was restricted by using integer genes in the range [-300,300] and each value was later divided by 100 so that weight values would lie in the interval [-3,3]. A similar strategy was used for the MTRNN, Perceptron and MLP agent, but instead weight values eventually lie in the interval [-5,5] and initial slow context states in the interval [0,1]. Since there are quite some weights to evolve in the MTRNN agent, the population size is set at 750. For the Perceptron agent, the MLP agent and the Control agent the population size is 250, 500, and 125, respectively.

During evolution, each chromosome of weights is transformed into an agent. For all agents that are evolved (the Control, MTRNN, Perceptron and MLP agent) the fitness value of a specific chromosome is the average of the amount of steps the corresponding agent is able to survive in 5 different runs and thus five different environments. This is done to ensure that a more accurate value of the agent's performance is calculated, since if the performance would be based on one environment the agent might do very well in one environment but perform worse in another. Since each simulation has a maximum duration of 750 steps, the maximum fitness is also 750. If during the training/evolution phase the settings of optimal parameters reaches the maximum fitness value, the agent's fitness will be increased with 10 to emphasize its good performance. In simulations after the best agent has been picked (i.e. in the testing phase), the maximum fitness value is 750 (see Section 2.5).

After every agent in a generation has been evaluated, the group of chromosomes with the highest fitness are used to create a new population. The genetic operators used to do this are mutation and recombination. For all agents, the algorithm will stop after 100 generations.

2.4.1 Alternative fitness functions for MTRNN agents

At the beginning of experimenting, different fitness functions for the MTRNN agent were evaluated to find out which training strategy would result in the highest performance for this specific agent and would thus be best fitted to use in the current experiments. The different fitness functions were tested in the random environment.

The first fitness function, the one that is eventually used to evolve the MTRNN agents, is the average amount of steps the agent was able to survive in a number of environments. However, this fitness function provides a fitness value that is based on the overall fitness in the whole simulation and therefore has a global character. An alternative method was tested, for which the fitness value was the average amount of correct decisions the agent was able to make in a number of environments. Correct decisions were for example avoiding obstacles when perceiving an obstacle, or moving in the direction of a food source. The goal of this latter fitness function was to provide a more locally based fitness value.

In Yamashita and Tani (2008) it is claimed that an agent using the MTRNN can learn various tasks and also sequence these tasks in appropriate order. From this it follows that it should be possible that the MTRNN can switch between different contexts to produce better behavior as well. Therefore two alternative fitness functions were created that focused on

Parameter	Value
Number of weights	Control: 63, MTRNN: 1000, Perceptron: 216, MLP: 495
Range of weight values	Control: [-3, 3], MTRNN, Perceptron, MLP: [-5,5]
Range of initial slow context state values	MTRNN:[0,1]
Population size	Control: 125, MTRNN: 750, Perceptron: 250, MLP: 500
Number of evolutions	Control: 100, MTRNN, Perceptron, MLP: 100
Maximum fitness	750

Table 3: Summary of the evolution parameters for the different agents

learning distinguishing different contexts and thus evolve different behaviors. Both of these alternative functions used the same three contexts:

- Context 1: food is directly within visual range, so the agent only has to move one step in the appropriate direction
- Context 2: food is within smell range, but a direct path may be blocked by an obstacle
- Context 3: No food is within range, so the agent can only be surrounded by obstacles, sign posts or empty ground

In the case of the first fitness function that focused on different contexts, the following happened: as soon as the agent perceived that it was in a certain context, the states of the slow context units were set to corresponding values. This is a variation on what happens in the fitness function that is currently used; in that case as soon as it becomes day or night, the slow context states are set. The second fitness function that incorporated different contexts enabled the agent to train on the three contexts separately instead of on the whole environment. Therefore also three different fitness functions were used: in Context 1, fitness was based on whether or not the agent was able to get the food in one step; in Context 2, a simulation had a time duration of four steps and fitness was based on whether or not the agent was able to get to food without bumping into obstacles; in Context 3, four step simulations were also used and fitness was based on how far away the agent was able to walk without bumping into obstacles.

However, all alternatives did not improve performance of the MTRNN agent and thus it was decided to stick with the old fitness function based on life duration.

2.5 Simulations

After the optimal parameter settings with the best fitness has been established for the agent that is currently tested, the agent with these parameters is run in 5000 simulations and thus

Parameter	Value
Number of simulations	5000
Variables	Number of food sources, number of obstacles, type of environment

Table 4: Summary of simulation parameters

environments. In each simulation the environment consists of a 10 by 10 grid. Simulations are run in both random and structured environments, but the type of environment remains consistent over these 5000 simulations (i.e. each agent is tested in both environments in 5000 simulations). At the start of each simulation, each agent has an energy level of 250. The final fitness result is the average of the fitness values of these 5000 simulations. A simulation has a maximum duration of 750 steps. Furthermore, a constant factor in simulations is the day and night rhythm and thus in all simulations the same rhythm is used.

Simulations are run to test the effect of various characteristics or variables. First of all, the performance of each agent is assessed in each type of environment. In the random environment, different ratios of food and obstacles are tested to get a clear idea of how the different agents perform under different environmental constraints. In the second or structured environment, the amount of obstacles is always zero and only the amount of food sources (and thus the amount of sign posts) is varied. Second, next to the fitness value of the agents also the observed behavior is analyzed, since the fitness value does not explicitly show what kind of behavior the different agents are producing.

2.6 Summary of methods

Table 5 lists all the experiments and variables that are of interest in the current thesis. For each test the performance value is varied over 5000 simulations.

3 Results

In this section the most important results will be shown and described. A complete and detailed overview of all the results can be found in Appendix B. Results will be provided in the form of performance graphs and a description of the visual behavior. The performance measure is defined as the amount of steps the agent is able to survive, i.e. the fitness value as described in the previous section.

3.1 MTRNN agent versus other agents in random environments

In this section the results answering the first part of the research question are shown and evaluated. The first research question is how well the MTRNN agent will perform as compared to other agents in a day-night environment, such as proposed in Lagarde (2009). This question is answered by evaluating the performance of the Reactive, Reactive-DN, Control,

Design	**Variables involved**
Each agent is tested in the random environment with varying ratios obstacles/food	Agents: Reactive, Reactive-DN, Control, MTRNN, Perceptron, MLP Environment: random Obstacles: 0, 10, 20, 30, 40 Food: 0, 10, 20, 30, 40
Each agent is tested in the random environment with varying food sources and no obstacles	Agents: Reactive, Reactive-DN, Control, MTRNN, Perceptron, MLP Environment: random Obstacles: 0 Food sources: 0, 2, 4, 8, 10
Each agent is test in the structured environment with varying food sources (and thus sign posts)	Agents: Reactive, Reactive-DN, Control, MTRNN, Perceptron, MLP Environment: structured Obstacles: 0 Food sources: 0, 2, 4, 8, 10 and sign posts: 4 * amount of food sources

Table 5: Summary of methods

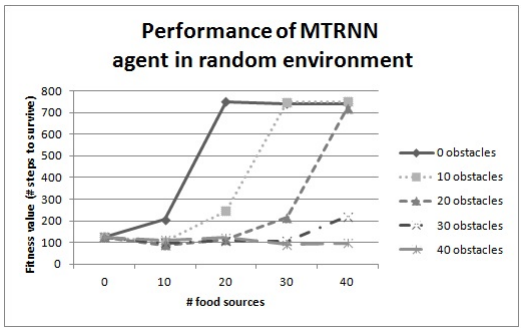
MTRNN, Perceptron and MLP agent in the random type of environment with varying ratios of food and obstacles. Specifically, the results of the MTRNN agent is compared to the results of the Reactive, Reactive-DN and Control agent which were already assessed by Lagarde (2009). Furthermore, the MTRNN agent is also compared to the Perceptron and MLP agent, to get an indication of how well the sensory information works in simpler network structures.

3.1.1 Performance graphs

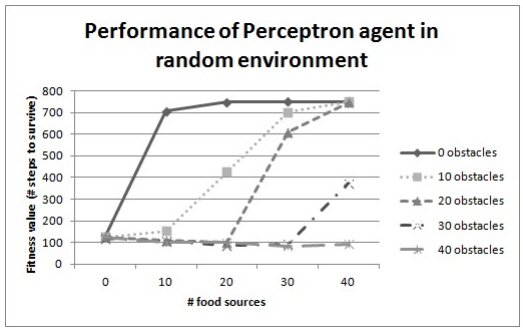
In Figure 4(a) to 4(f), the performance of each agent is set out against different numbers of food and obstacles.

As was analyzed and described in more detail by Lagarde (2009), the Reactive, Reactive-DN and Control agent performed quite well and as to be expected for all combinations of food and obstacles. To be more specific, these three agents follow a rather sensible line of performance: as the number of food sources increases or the number of obstacles decreases, the performance also increases until it reaches the maximum performance level at certain food/obstacle ratios and vice versa. Difficult environments - with many obstacles - are a lot harder to survive in than in environments where there is more to gain, which makes a lot of sense.

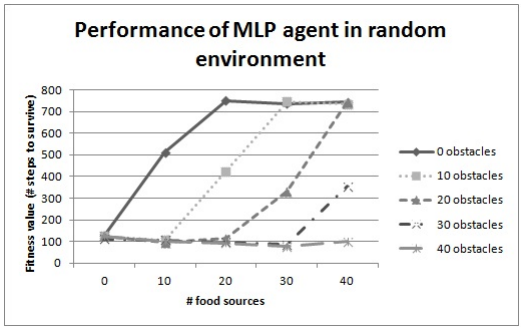
In very easy environments that only contain food and no obstacles, the Reactive Agent performs best of all agents (see Figure 5). In such environments, there is no need to go to sleep because there is no danger of bumping into obstacles at night. Although the agents do



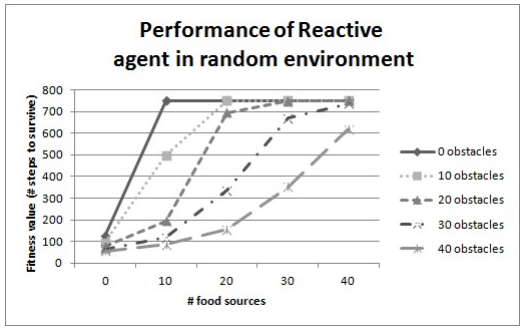
(a) Performance of MTRNN agent in environment I



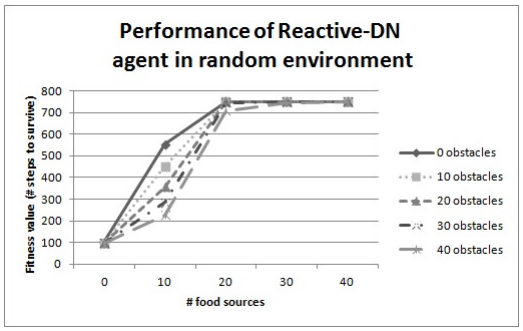
(b) Performance of Perceptron agent in environment I



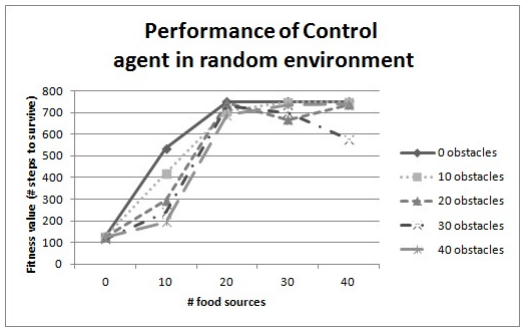
(c) Performance of MLP agent in environment I



(d) Performance of Reactive agent in environment I



(e) Performance of Reactive-DN agent in environment I



(f) Performance of Control agent in environment I

Figure 4: Performance of all the agents in the random environment with several different food/obstacles ratios

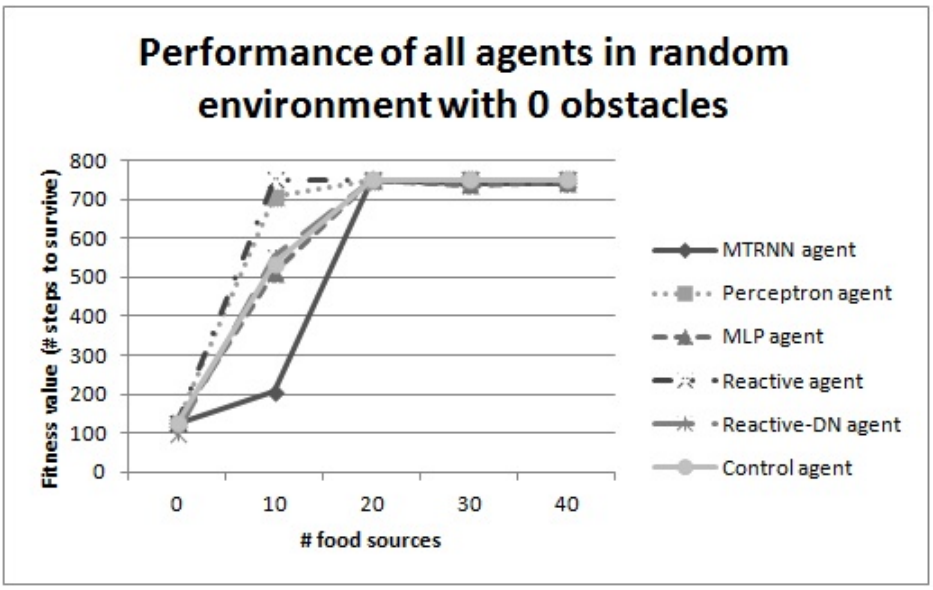


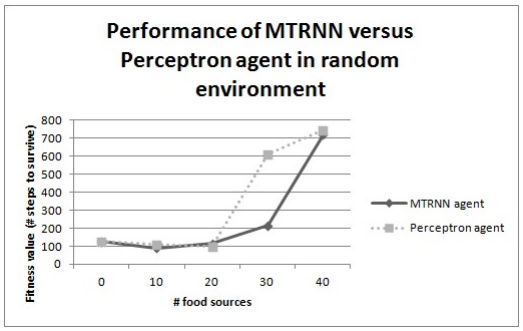
Figure 5: Performance of all agents in the random environment with 10 obstacles set out against the amount of food sources

not perceive their surrounding correct at night, the agent still collects enough food at night to not go to sleep. The Control agent will learn to walk at night in such situations, but the Reactive agent has lower costs for moving and resting than this agent. Therefore the Control agent still performs worse than the Reactive agent.

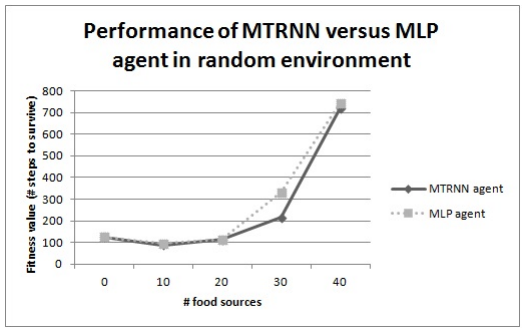
Figures 4(a) to 4(c) show the lines of performance for MTRNN, Perceptron and MLP agents, respectively. One can easily observe that these three agents have a similar learning curve; performance varies around 125 until the amount of food is higher than the amount of obstacles. After this threshold is passed, the performance increases linearly as the amount of food also increases. In contrast to the case of the Reactive, Reactive-DN and Control agent, this is not what one would expect for agents to show. Clearly, the ratio obstacle/food plays a very important role in the evolution of these agents.

Figure 6(e) shows that for 20 obstacles and several amounts of food sources, the Control agent performs equal or higher than the MTRNN agents. In situations where the environment is completely empty, i.e. no obstacles and no food sources, both agents will use their neural networks to figure out that to stay at the same spot during a simulation is the best strategy and thus the agents perform equally well. Almost the same holds for the Reactive and Reactive-DN agents compared to the MTRNN agent, but in situations where there is no food and no obstacles, the MTRNN agent will live longer on average. This can be explained by the fact that the reactive agents still wander around even when there is no food, which costs more than resting.

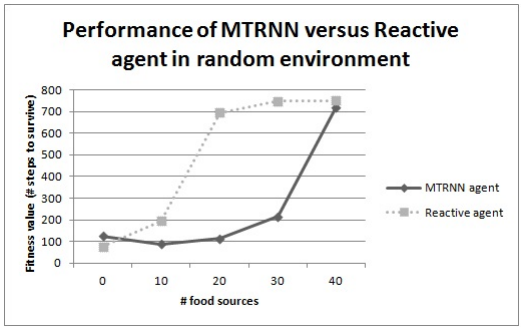
In situations where the amount of food is higher than the amount of obstacles, both the Perceptron and MLP agent have higher performance than the MTRNN agent, as can be seen



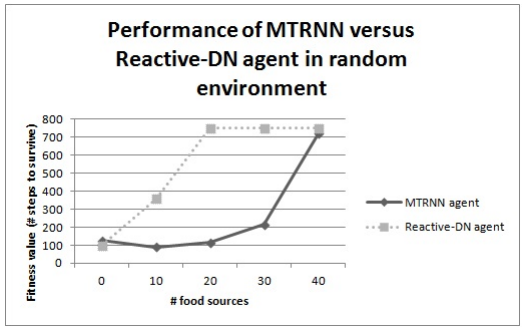
(a) Performance of MTRNN agent versus performance of Perceptron agent



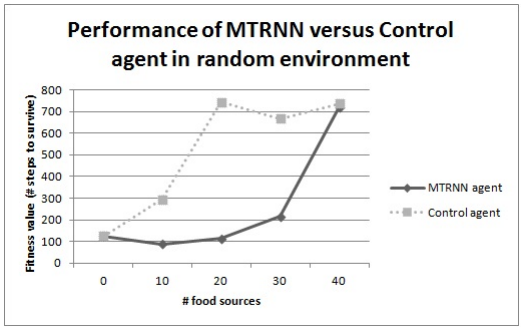
(b) Performance of MTRNN agent versus performance of MLP agent



(c) Performance of MTRNN agent versus performance of Reactive agent



(d) Performance of MTRNN agent versus performance of Reactive-DN agent



(e) Performance of MTRNN agent versus performance of Control agent

Figure 6: Performance of MTRNN versus all other agents with 20 obstacles and varying amount of food sources in random environment

in Figure 6(a) and 6(b). However, the difference in performance is bigger in the case of the Perceptron. These figures also show that this difference is not observed anymore when the amount of food (40) is a lot higher than the number of obstacles; in that case, all agents will approach maximum performance value. In opposite situations, so in environments where the amount of food is smaller than or equal to the amount of obstacles, the three agents have equal performance. This all suggests that the hidden layer in the MLP agent and the different context units in the MTRNN agent do not add anything to the effectiveness of the agent.

Overall the six agents can be divided in two groups: on the one hand the Reactive, Reactive-DN and Control agents that perform quite well in different circumstances, even when the amount of obstacles is higher than the amount of food; on the other hand the MTRNN, Perceptron and MLP agents, for which performance only increases when the amount of food is larger than the amount of obstacles and is quite constant otherwise.

3.1.2 Observed behavior in random environments

Below the behavior that was observed for all agents in the random environment is described in detail.

Reactive, Reactive-DN and Control agents
Since the default behavior is to wander randomly, the Reactive agent is in motion most of the time, even at night. In daylight, obstacles are always avoided - except when the Evaluate Extreme Hunger layer is activated - but at night, when grid cell types become hard to distinguish, the Reactive agent bumps into obstacles regularly. As mentioned before, in situations where there are no obstacles and just food, this strategy is very successful.

Due to its build-in day and night rhythm, the Reactive-DN agent moves only at daytime and rests during the night. Its behavior at day is the same as that of the Reactive agent, namely searching for food while avoiding obstacles.

The Control agent is able to develop different day and night behavior without a hardcoded day and night rhythm. Therefore in many situations the behavior of the Control agent is similar to that of the Reactive-DN agent. However, different behavior for Control and Reactive-DN is shown when there is enough food ($>$ 10 food sources) and no obstacles; in such situations, the Reactive-DN will still go to sleep at night while the Control agent will search for food the whole time. To preserve energy, the Control agent takes short naps during the day in some situations, as also reported by Lagarde (2009).

It must be noted in the current context that all of these agents show obstacle avoidance behavior. This is of course due to the Obstacle Avoidance layer, but it is an important point to keep in mind since the MTRNN, Perceptron and MLP agent do not use such a behavioral layer and may not behave in such a straightforward way.

The MTRNN, Perceptron and MLP agents
In this random type of environment, the MTRNN agent shows a lot less nuanced behavior. The performance graph of this agent (Figure 4(a)) already suggested that the ratio

food/obstacles is very important and this notion is further supported by the behavior that is observed: if there are more or equal obstacles as compared to the amount of food, the agent will not move throughout the whole simulation, except for some idle movements in any direction. In opposite situations, the agent will always move around in the same kind of pattern (for example, 2 steps to the right, 1 up). Therefore, the agent heads in the same direction throughout the simulation (for example, from the upper left corner to the upper right corner). The same strategy is thus used during the whole simulation. In very few cases, the MTRNN agent is able to develop different behavior for day and night; the agent will walk at day and rest at night. This behavior is only observed for special ratios of food/obstacles, e.g. if this ratio is 20/20.

The MLP agent shows behavior similar to that of the MTRNN agent, except with none of the ratios food/obstacles that were evaluated does the MLP agent show different behavior for day and night. The Perceptron agent also shows the same kind of behavior pattern throughout one simulation. Furthermore, the Perceptron agent is able to show different behavior for day and night for specific food/obstacle ratios as well. The latter indicates that the fact that this is not observed for the MLP agent is a matter of parameter choice: the MLP agent probably will show different day and night behavior for specific ratios as well, just not for the ones that were evaluated in this experiment.

An interesting observation is that all these three agents never show obstacle avoidance behavior, not even when it is day and obstacles are correctly perceived. Since also Perceptron and MLP agents show this specific behavior, it cannot be due to the fact that sensory information is processed indirectly by fast context units in the MTRNN agent. A possible explanation for this observation could be that these three agents are only able to develop very global behavior (such as the same behavior patterns shown throughout a simulation) and cannot handle local situations.

3.1.3 Conclusion

The results showed that in the majority of the different food/obstacle ratios, the MTRNN agent has equal or lower performance than every other of the five agent types. The only exception is in situations where there are only obstacles and no food sources; the MTRNN agent than is able to survive longer on average than Reactive and Reactive-DN agents and has equal life duration as the Perceptron, MLP and Control agent. This indicates that the random environment just may be too random for the MTRNN to benefit from its multiple timescales, i.e. to remember the past. This notion is also supported by performance graphs 6(a) and 6(b), which furthermore show that the MLP and especially the Perceptron agent perform better than the MTRNN agent in situations when the amount of food sources is larger than the number of obstacles.

Both the performance graph as well as the observed behavior support the suggestion that the amount of food versus the amount of obstacles plays an important role in what kind of behavior is shown. Only if there are more food sources than obstacles, the agent will learn to move during the day and with fewer specific ratios will the agent show a day-night rhythm.

3.2 Random versus structured environments

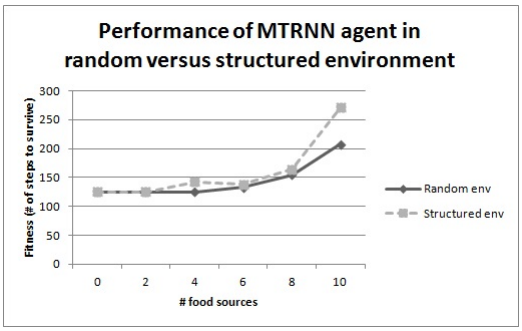
The second research question - will the MTRNN agent behave more effectively in an environment with more structure versus purely random environments - is answered by comparing results for all agents in random and structured environments. In the random environments also no obstacles are placed, so only the presence of sign posts is put to the test. Results for the Reactive, Reactive-DN and Control agent will only be mentioned quickly, since these agents do not have a behavioral layer to handle sign posts. Therefore the main focus will be on comparing the results for the MTRNN agent in different worlds and to the results of the other network agents as well.

3.2.1 Performance graphs

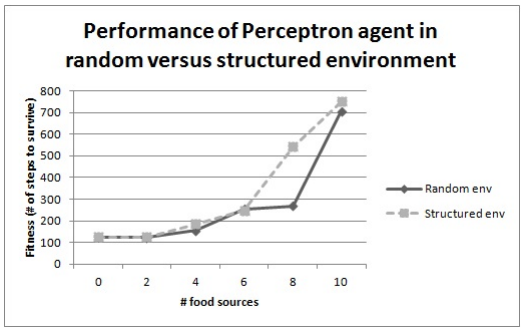
Figure 7(d) shows that the performance of the Reactive agent is in both environments, the random and structured environment, almost completely the same. This was to be expected, since these agents cannot distinguish sign posts from ordinary ground cells because there is no reactive layer that handles sign posts. The results of this agent in the structured environment are thus not very useful to get to know something about the environment. Still, the results can be used to find out how effective walking around randomly searching for food is in the structured environment as compared to the performance of the MTRNN agent. The same holds for the Reactive-DN agent (see Figure 7(e)). Although the Control agent is in fact able to recognize situations in which it stands on a sign post cell, it still misses a reactive layer that commands specific behavior when sign posts are encountered. The Control agent might still be able to develop different behavior than in the random environment, but observed behavior will be evaluated in the next section. However, Figure 7(f) suggests that no different behavior is shown in the different environments, except in the case of 10 food sources: the performance in the structured environment is higher in such a case as compared to the random environment. See below for a discussion of this observation.

In 7(a) it can be observed that the MTRNN agent performs the same in both environments for the majority of the evaluated situations. However, two points provide an interesting exception: the fitness value of the MTRNN agent in the structured world is higher than in the random world when the amount of food sources is 4 or 10. The latter fluctuation is also observed in other agents and will be discussed below. However, something interesting seems to happen when the amount of food sources is 4. It remains to be seen in the next section when observed behavior is evaluated what might be the reason for this, since this cannot be derived from the performance graph. Figures 7(b) and 7(c) show the comparison of the performance in the different environments for the Perceptron and MLP agent. The fitness value of the Perceptron agent is almost always higher in structured environments, but this difference becomes bigger in the case of 8 food sources. For the MLP agent performance is only higher in the structured environment when there are 8 food sources or more. Again, the next section should explain more about the possible reasons for these fluctuations.

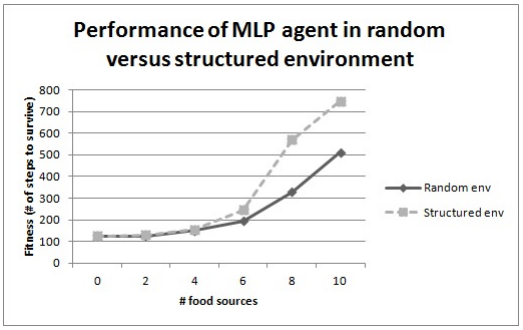
It must be noted that the performance graph of the MTRNN agent (Figure 7(a)) scales differently than the other performance graph in this figure; the y-axis for the MTRNN agent



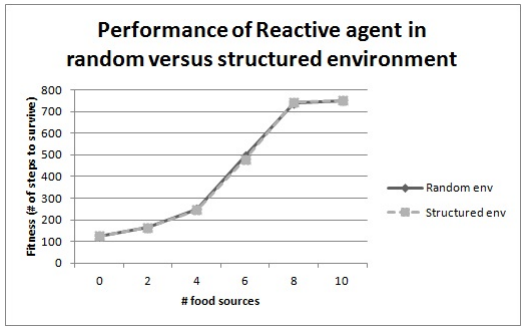
(a) Performance of MTRNN agent in the random versus the structured environment



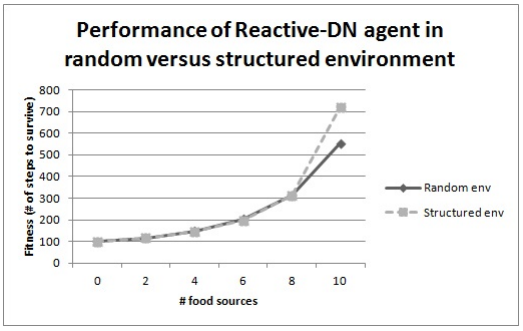
(b) Performance of Perceptron agent in the random versus the structured environment



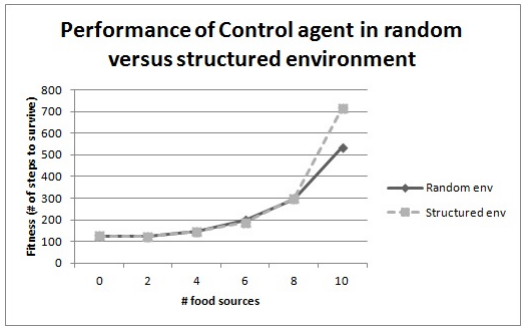
(c) Performance of MLP agent in the random versus the structured environment



(d) Performance of Reactive agent in the random versus the structured environment



(e) Performance of Reactive-DN agent in the random versus the structured environment

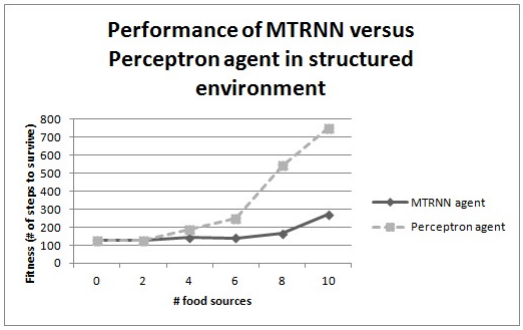


(f) Performance of Control agent in the random versus the structured environment

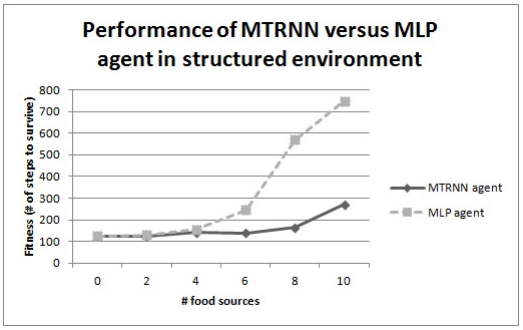
Figure 7: Performance of all the agents in the random environment versus the structured environment

only goes to 300, while the y-axes of the rest of the figures go to 800. This shows that the MTRNN agent performs quite worse than other agents in both environments, but it also implicates that the differences observed are not as big as differenes in other plots.

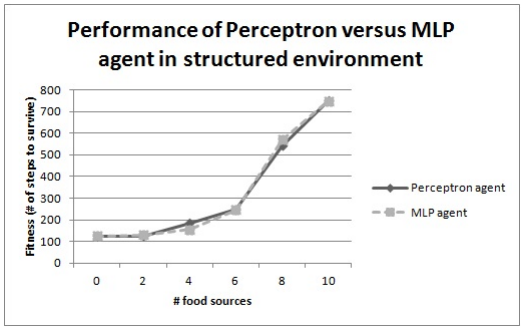
Figures 8(a) to 8(c) show the performance plots of the MTRNN, Perceptron and MLP agent against each other in the structured environment. Clearly, both the Perceptron and MLP agent perform better than the MTRNN agent. This indicates that there is some difference in the sort of behavior the MTRNN agent performs and might suggest that the various timescales do have an effect on this behavior, albeit a negative effect.



(a) Performance of MTRNN agent versus Perceptron agent in the structured environment



(b) Performance of MTRNN agent versus MLP agent in the structured environment



(c) Performance of Perceptron agent versus MLP agent in the structured environment

Figure 8: Performance of Perceptron, MTRNN and MLP agents compared with each other in the structured environment

As briefly mentioned above, another interesting point that can be observed in Figure 7(a) to 7(f) is that all agents (except for the Reactive and Perceptron one) have higher fitness values in the structured environment when there are 10 food sources. Because also the performance of the Reactive-DN agent is improved in that situation, this increase is probably not due to the fact that there is more structure in the environment. Instead, it must be noted that when there are 10 food sources there will be a food source in every grid column. Therefore the food is much more evenly distributed than in the case of 10 food sources in the random environment and thus walking around searching for food will result

in more food consumption in the structured environment. This difference is not observed for the Reactive and Perceptron agents, but the performance for these agents has already reached maximum value for both environment types.

3.2.2 Observed behavior in random versus structured environments

Below the behavior that was observed in both the environments is described in detail for all agents, although the focus will be on the MTRNN, Perceptron and MLP agents. It must be noted that the different amounts of food sources tested here are quite low. Such low amounts were not tested for the random environment in the previous sections, so it is possible that new behavior in this environment is observed as well.

The Reactive, Reactive-DN and Control agents
As the Reactive agent performs the same in both worlds, the behavior of this agent is also similar in the random and structured environment. Both day and night the agent walks around randomly and if a food source is perceived, the agent will try to move towards it. In the structured environment there are no obstacles, so no obstacle avoidance behavior is needed and thus is not observed. Furthermore, since the agent does not use a reactive layer that handles sign posts, these type of grid cells are classified as empty grid cells and thus no specific behavior for sign posts is observed.

The Reactive-DN agent also does not show different behavior in the two environments; in both the random and structured world the Reactive-DN agent will walk around randomly at day searching for food while ignoring sign posts. When night falls, the agent will rest at the same spot. Again, no obstacle avoidance behavior is needed.

Although the Control agent can identify sign posts if it stands on such a grid cell, the agent behaves almost always the same in the structured environment as in the random environment. For both environments holds that if the amount of food is quite low ($\pm$ 2 food sources), the agent will only minimally move at day and definitely stand still at night. There are just too few food sources for the agent to risk wandering around a lot. As the amount of food increases (e.g. 4 food sources), a difference in behavior for the two environments is observed. In this specific situation, agents in the random environment will sometimes have little naps at day, while for agents in the structured environment such naps are not observed. If food increases, this difference at day is not observed anymore. For all food amounts that were tested and in both environments, the agent will rest at night most of the time.

The MTRNN, Perceptron and MLP agents
The first observation that can be made is that there is some sort of default movement pattern underlying the behavior of the MTRNN agent. This holds for both the random and the structured environment. It is very situation dependent whether or not the agent will subsume this pattern with other behavior (e.g. to collect a food source in range). Moreover, the MTRNN agent shows quite some different sorts of behavior. It must be noted that behavior varies more between different amounts of food in the structured environment. The behavior of the MTRNN agent for different numbers of food sources:

- **0 or 2 food sources:** both in the random as in the structured environment, the MTRNN agent will not move (or only minimally) throughout the simulations.

- **4 food sources:** Figure 7(a) shows that the average fitness value for the MTRNN agent is higher in the structured environment as compared to the random environment. What actually is observed in the structured environment is that at day, the agent will move in a diagonal movement pattern until a sign post or food source is detected. If a sign post is encountered, the agent will follow the upward strip of sign posts until the food source at the top is consumed. The agent can abandon this upward movement and get food if food on the left or right side is closer than the food source at the top of the sign post strip. In the random environment, the agent will largely follow the same movement pattern throughout the simulation (since there are less interesting points - no sign posts - to abandon default behavior). Furthermore, the agent takes little naps at day, which is not observed in the structured environment. In both environments the agent rests at night.

- **6 food sources:** in the structured environment, the agent does not make use of the sign posts anymore. Instead, the agent adopts a movement pattern at day which approaches the food sources from above and thus the agent will not encounter many sign posts. At night, the agent will go to sleep. In the random environment the agent sticks to the same movement pattern throughout the simulation, even at night.

- **8 food sources:** again, the MTRNN agent does not make use of the sign posts explicitly in the structured environment. The agent will move at day in an upward diagonal movement pattern, which is not abandoned when a sign post is encountered. A similar movement pattern (although with almost no diagonal movements) is adopted in the random environment. In the structured environment the agent will go to sleep at night, while in the random environment the agent will keep on moving.

- **10 food sources:** Figure 7(a) shows that, again, the performance of the MTRNN agent in the structured environment is remarkably higher than in the random environment. As explained in the previous section, this is probably due to the equal distribution of the food sources. The behavior that is observed in the structured environment is that the agent will not leave a certain grid column; as soon as food is consumed, another food source will appear in the same grid column. In the random environment, the agent will walk around the environment. In both environments, the agent will also walk at night.

The behavior that is observed in the Perceptron is quite similar to that of the MTRNN agent, except for some small differences. In the structured environment, the Perceptron agent will use the sign posts when there are 4 food sources and not anymore when the amount of food is higher. However, the Perceptron agent will start to walk at night sooner than the MTRNN agent (this was observed for 6 food sources). Furthermore, if there are 10 food sources the Perceptron agent will not stay in the same grid column (except at night) but will explore

the environment searching for food. In the random environment, the Perceptron agent will adopt a specific movement strategy throughout the simulation. At night, the agent will move but takes little naps as well.

Contrary to the Perceptron and MTRNN agent, the MLP agent will not use the sign posts in the case of 4 food sources. Instead it adopts a similar movement pattern that was also observed in the MTRNN agent for 6 food sources. However, sign posts are used in the case of 8 food sources, especially at night. This may explain the higher performance in the structured environment in Figure 7(c). It must be noted that the agent then also adopts a movement pattern in upward direction and thus more sign posts are encountered. In the case of 10 food sources in the structured environment, the MLP agent will not stay in the same column (except at night) but occasionally will move to an adjacent column. In the random environment, the MLP agent will show similar behavior to that of the Perceptron agent.

Although observed behavior is thus at some points quite similar in these three agents, the movement patterns that the Perceptron agent and MLP agent adopt must be more effective than the patterns of the MTRNN agent since otherwise the performance of these agents would not be as high in the second environment (see Figure 7(b) and 7(c)).

3.2.3 Conclusion

The results described in the current section show that in the case of 4 food sources, the MTRNN agent has higher performance level in the structured environment than in the random environment. More specifically, the agent uses the sign posts to develop a more effective strategy to reach the food. In some other of these situations, the agent does have higher performance but develops a strategy that does not include sign posts. This suggests that if there is too much food, using the sign posts will not be effective anymore. Therefore the situation of 4 food sources poses an interesting case in the current context.

The MLP and Perceptron agent also followed the sign posts for 4 and 8 food sources, respectively. This suggests that all three agents were able to learn similar patterns - including different day and night behavior. This indicates that it is not really necessary to have multiple timescales in this specific environment to reach higher performance. However, Figures 8(a) to 8(c) show that the performances for the MLP and Perceptron agent are a lot higher than for the MTRNN agent. This may indicate that the MTRNN agent does use different cognitive skills than the other agents - namely using it's long- and short-term memory - although it is not really effective in the current task and environment. Since in situations where sign posts are used the performance is higher than in other situations for all three agents, this suggests that adding sign posts really improves the performance for network agents (albeit only for specific cases).

All agents - except for the Reactive and Perceptron agent which already reached maximum performance in the random environment - performed better in the structured environment when there were 10 food sources. As suggested earlier, this is probably due to the fact that in the structured environment the food is more evenly distributed across the environment. The performance increase is thus due to the structure of the environment

itself.

3.3 Optimal environment and task

In Sections 3.1 and 3.2, it was observed that the performance of the MTRNN agent was worse than the MLP and Perceptron agent in both types of environments (see Figures 6(a), 6(b), 8(a) and 8(b)). This highlights the importance of the next research question: In what kind of environments and tasks will the MTRNN agent be able to benefit (show effective survival behavior) from having multiple timescales, i.e. remember previous time states? Although Section 3.2 may suggest that the worse performance value of the MTRNN agent is due to the multiple timescales, this is actually not proved but it clearly suggests that the agent does not benefit from having multiple timescales. However, the results of the previous two sections and some background information should give some pointers to what kind of environments and tasks are suitable for the MTRNN agent.

In many random environments, the MTRNN agent will not move throughout the simulation unless there is enough food to overcome the ratio food/obstacle threshold. This suggests that the MTRNN agent does not extract any more information than whether or not enough food is available in the environment. The strategy the agent will pursue is thus too holistic (or global for that matter) to handle local situations as well. Although the structured environment was less random and it was observed that in some situations the sign posts were used, overall the MTRNN agent followed the same movement pattern throughout the simulation and again the strategy was holistic.

In contrast to the current experiment, the work of Paine and Tani (2005) and Yamashita and Tani (2008) shows that the MTRNN or a similar structure can be effective; the network architectures were able to develop a functional hierarchy through self-organization and multiple timescales and could develop different behavior primitives and sequences of such primitives. However, it is important to note that the task and environment of their experiments were quite different from the ones that are used in the current research. Specifically, in Yamashita and Tani (2008) the agent had to perform five different behaviors (such as move object left and right three times). The agent always started out in the same position and which behavior was to be performed depended on the initial states of the slow context units. In Paine and Tani (2005), the agent had to find different goals in a static maze; which goal was to be found was again dependent on the initial slow context states. In the current experiment, the task was to survive as long as possible which could be done by avoiding obstacles, consume food and resting. However, the environment was not static at all and thus the agent really had no benefit of remembering where food had been before consumption. Therefore the subgoals (food sources) to reach the more abstract goal (surviving) were also not static, while in Yamashita and Tani (2008) and Paine and Tani (2005) the goals were precisely that and thus much more concrete.

From all this the following can be derived:

1. The environment must contain a lot of structure, so that it is beneficial for the MTRNN agent to remember;

2. The goals the agent has to achieve in a task must be static between simulations

A third constraint could be that the task and environment require a global and holistic strategy for effective behavior, but this already follows from the two constraints mentioned; if the world is completely static the agent will always use the same behavior patterns for the same goals, again dependent on the initial slow context states.

An example of a task and environment in which the MTRNN agent might be able to benefit from its cognitive abilities is a task for which the agent must reach different goals from different points in a static world. One can think of some kind of quest in which the agent gets an initial goal and only gets to know the next goal if it has reached the previous goal. The subgoals are also not always in the same order. This is very much like the maze navigation task of Paine and Tani (2005), but the difference is that in their task the agents always navigated to a goal from the same point, while in this task the agents must be able to navigate to different goals from different starting points.

Another possibility might be to add a random factor to an otherwise static world by for example letting a predator loose while the agent has to survive by searching for food. The behavior of agents for such a task when a predator is present was already evaluated by Bax (2010), but only for the Reactive, Reactive-DN and Control agent. The same could thus be done for the MTRNN agent as well. It is expected that because the world is static, the MTRNN agent can handle the task well but the random factor would encourage the agent to produce more reactive behavior than in the current experiment.

A third possibility is to use a task and/or environment in which the agent has to follow a specific kind of order to complete certain subgoals. Because of the timescales, the agent might be able to develop subgoal-specific behavior through the fast context units. The example of the quest given above also includes ordering of behavior patterns, but other examples in which the order varies not much are also possible.

Although random environments are not really an option for the MTRNN agent, still many different environments and tasks are possible. These tasks and environment do have some constraints; the world must not contain too many random or unexpected things, is static or at least has much structure and contains clearly specified goals and sub goals.

3.4 Emergence of functional hierarchy

This section focuses on the fourth research question: is the agent able to achieve functional hierarchy through different timescales in the current survival task and environments? Again, this question can be answered by comparing the MTRNN agent with the Perceptron and MLP agent; these latter agents use the same kind of sensory information but will not have functional hierarchy of behavior through different timescales, implicit nor explicit. Furthermore, observation of the MTRNN agent's behavior in both environments should give some indication of whether or not this agent develops functional hierarchically organized behavior.

In Section 3.1 and 3.2 is described that in both environments the MTRNN agent performs worse than the MLP and Perceptron agent. However this is only the case for situations in

which the amount of food is higher than the number of obstacles (as shown in Figures 6(a), 6(b), 8(a), 8(b). This indicates that for all agents it holds that the ratio food/obstacles should be beneficial, otherwise the agents will not move at all. These situations should therefore not be included when evaluating emergence of functional hierarchy in the MTRNN agent. The fact that the Perceptron and MLP agent performed better than the MTRNN agent suggests that remembering previous states and use of different timescales have an effect on the performance, albeit a negative effect. It also suggests that the MTRNN agent develops different behavior patterns than the MLP and Perceptron agents, but again not very effective ones.

In the Methods section was described that the states of the slow context units were set to corresponding values as soon as it became night or day. This should have triggered the MTRNN agent to develop different behavior for day on the one hand and night on the other, but is this also the case? According to the observed behavior (as described in detail in Section 3.1.2 and 3.2.2): yes, again but only for some situations did the MTRNN walk at day and stay still at night. Furthermore, although not as much in the MTRNN agent, different day and night behavior was also observed for the MLP and MTRNN agents. This indicates that such different behavior patterns are not necessary due to the setting of the initial slow context states in the MTRNN agent but probably because all these network agents recognize the pattern of day and night.

One interesting observation must be noted; in the case of 4 food sources in the structured environment (the one with the sign posts), the MTRNN agent will most of the time hold on to a default movement strategy but abandons this as soon as a sign post or food source is perceived. When this happens, the agent will temporarily move to achieve this sub goal. This suggests there is some subsumption of different behaviors; as soon as an interesting surrounding is perceived (such as a food source), the agent will move towards this goal and thereby subsumes the default behavior patterns. In other situations and in particular in the random environment, the MTRNN agent sticks to one global strategy of movement and will not visibly react to particular situations, such as encountering a food source, obstacle or sign posts.

Since other network agents also show different behavior for day and night, it is not possible to conclude that this is due to emergence of functional hierarchy in the MTRNN agent, in the sense that these behavior patterns were not developed due the setting of initial slow context states and the multiple timescales. However, it cannot be denied that overall the MTRNN agent performs worse than the MLP and Perceptron agent and thus there must be some difference in the behavior that is produced by the MTRNN agent. Furthermore, in a particular case the MTRNN agent performs quite well and seems to show behavior as a result of some sort of subsumption architecture.

In all, the MTRNN agent does not really benefit from having multiple timescales in this type of environment and task, but the differences with other network controlled agents suggest that these timescales do have an effect on performance. Since other network agents also show different behavior patterns, the current context that is perceived seems to play a much bigger role than the architecture of the networks. The case of 4 food sources in the

structured environment does show some potential of emerging functional hierarchy, but in the current experiment there is not enough evidence to fully support this notion. It remains to be seen in future work if the MTRNN agent can really self-organize this functional hierarchy in similar tasks and environments.

3.5 Statistical significance

In the previous results sections, many differences between different agents or between different worlds were described. But are these differences statistically significant? In the following section an answer to this question is provided by giving an example of statistical testing.

3.5.1 Evolution statistics

During evolution of the Control, MTRNN, Perceptron and MLP agents, the fitness value was based on the average of the fitness values obtained in five different simulations and thus five different environments. However, there are many possible environments; take for example the situation with 10 obstacles and 20 food sources in the random environment. The amount of possible environments is then:

$$\binom{100}{10} \cdot \binom{100-10}{20} = 8,82 \cdot 10^{32}$$

so the population of environments can be considered virtually infinite. According to the Central Limit Theorem and the Law of Large Numbers (Billingsley, 1986), for a reliable approximation of the population average the sample size should be large enough - $N > 40$. This is clearly not the case in the current experiment because during evolution, the fitness value is averaged over only five different environments, which means that the average fitness value on which the fitness during evolution is based does not reflect the real average fitness well.

3.5.2 Comparison of different agents and environments

To answer the various research questions in this thesis, many comparisons were made between agents and between environments. One of the most interesting observations was that the performance of the MTRNN agent was higher in the structured environment as compared to the random environment. Whether this difference is also significant will now be assessed as an example.

Each type of agent with optimal parameters (and thus the best fitness value) for the current environment acquired through evolution was tested in 5000 different environments. The final performance value was the average fitness value of these 5000 environments. Because the sample size is this big, all differences that are found in the data will probably be significant.

In this specific example, the focus is on whether or not the performance of the MTRNN agent is significantly higher in the structured environment than in the random environment.

	Structured environment	**Random environment**
Average	$X_{gem}(A) = 142,4130$	$X_{gem}(B) = 124,7870$
Stdev	$s_A = 13,0497$	$s_B = 9,584...$
# of subjects	$n_A = 5000$	$n_B = 5000$

Table 6: The statistics acquired for the comparison of the performance value of the MTRNN structured and random environment with 4 food sources

We therefore have the following design and hypotheses:

Design:
Dependent variable = the performance of the MTRNN agent with 4 food sources and 0 obstacles
Independent variable = type of environment (structured, random)

Hypotheses:
$H_0 = \mu(Structured) \leq \mu(Random)$
$H_a = \mu(Structured) > \mu(Random)$

The following statistics were acquired for the two environments and can be found in Table 6. With help of the formulas listed in Appendix C, the p-value for the current statistic test is calculated. $p < 0.0005$, which means that there is significant proof that the null hypothesis can be rejected and thus that the performance of the MTRNN agent in the structured environment is significantly higher than in the random environment for 4 food sources.

4 Conclusion

In this thesis, the main goal was not only to investigate the effectiveness and possibilities of the MTRNN agent compared to other agents and in different environments, but also whether or not the agent develops functional hierarchy of behavior through self-organization and use of multiple timescales. The goal consists of four sub goals:

1. Evaluate and observe how effective the MTRNN agent performs in a random day-night environment (Lagarde, 2009) as compared to agents with different behavior paradigms;

2. Evaluate and observe the behavior of the MTRNN agents in the random environments as compared to a more structured day-night environment;

3. Discuss and derive from the results and other information in what kind of environment the MTRNN agent will benefit from having multiple timescales, i.e. remember previous time states;

4. Evaluate whether or not the MTRNN agent was able to achieve functional hierarchy in the current experiment.

To answer the first research question, six different agents - two reactive agents, one regulative controlled agent and three network agents controlled by a Perceptron, MLP or MTRNN network - were compared to each other in a randomized day-night environment with obstacles and food sources. The results showed that the performance and behavior of the MTRNN agent was in almost every case worse than the reactive, Control and Perceptron agents. In many cases, the MTRNN agent behaved and performed the same way as the MLP agents. This suggests that agents with a simpler architecture were better adapted to the random environment than an agent with a more complex architecture such as the MTRNN agent. The MTRNN agent was not able to extract more information of what it has seen in previous situations than whether or not there was enough food to risk walking around, so it seems that the random environment is just too random for the MTRNN agent to be effective.

To answer the research question concerning the second goal, the performance and behavior of the MTRNN was evaluated in two different environments. The first environment was the random world with food and obstacles, the second environment did not contain obstacles but instead used strips of sign posts leading to and therefore indicating a food source. These sign posts were used to create a more structured environment, although the strips of food sources and sign posts were still randomly placed in each simulation (under the constraint that maximally one food course could be placed in a column). Because of the structure, it was expected that the MTRNN agent (and the Perceptron and MLP agent as well) would be able to recognize patterns and the MTRNN agent could thus benefit from remembering previous states. However, this expectation was only partially supported by the results; the MTRNN agent was still constrained by the fact of whether there was enough food but the use of sign posts did improve performance in some specific cases. A very interesting example is the situation with 4 food sources; in the random environment the agent will take many naps at day to preserve energy, but in the structured environment the agent will move around more and make use of the sign posts to reach food and therefore performance will be higher in the latter situation. However, the agent still performs worse than the MLP and Perceptron agent. Also, the MTRNN agent will largely follow the same kind of movement strategy throughout a simulation, which suggests that the second type of environment is still not structured enough for the MTRNN agent to benefit from having a memory.

The third question - in what kind of environments and tasks will the MTRNN agent be able to benefit from its multiple timescales - was answered by evaluating the results obtained in the current experiment and other research in the same field of interest, specifically the work by Tani (Yamashita & Tani, 2008; Paine & Tani, 2005). The research suggested that an environment that is completely static throughout the experiment and with static goals or sub goals is a typical environment in which it is useful to remember previous states and also develop different behavior primitives that can be sequenced. However, these types of environments and tasks may not be that challenging because the situations stay the same. Therefore such environments can be expanded by introducing a random factor (such as a predator agent) or by introducing multiple static goals at multiple starting points. In any

case the environments and tasks should encourage the agent not only to have a global idea of the world, but also develop behavior that handles local situations.

The fourth research goal focused on whether or not functional hierarchy of behavior emerged in the MTRNN agent. This question was answered by comparing observed behavior of this agent against the MLP and MTRNN agent and by evaluating the different behavior patterns that were observed in the different situations. The results and observations showed that all network controlled agents were able to recognize different day- and night patterns, dependent on the environment settings, indicating that environment context plays a more important role than the structure of the network models and use of different timescales. However, differences between MLP and Perceptron agents on the one hand and the MTRNN agent on the other suggested that the multiple timescales did have an effect on the performance, albeit a negative effect. Furthermore, it seemed like there was some subsumption of behavior in the case of 4 food sources in the structured environment for the MTRNN, which is in favour of the notion of functional hierarchy but in the current experiment there is not enough evidence to support this notion. This, and the constraints on the environment and tasks derived from these experiments, again emphasize the need for further research in this field of interest.

5 Discussion & future research

In the previous sections different possible environments and tasks were proposed in which the behavior of the MTRNN agent might be able to act effectively. It would therefore be very interesting to evaluate and observe what kind of behavior the MTRNN agent will show in such environments and tasks.

In the current context, the concept of cognitive control (Aaron & Admoni, 2010; Badre & Wagner, 2007; Badre, 2008; Braver et al., 2009; Egner, 2009; Koechlin et al., 2003; Rajah et al., 2008) - the ability to react different to the same situation dependent on the context - is also important. For example, take the static world in which also a predator is present; the agent must be able to react to the same environment but in either a context of danger or a context in which no immediate threat is visible. The formalization by Vroon (2011) was also concerned with cognitive control. He formalized the idea that regulative controlled agents (the Control agent in the current experiment) would be better able to handle situations with different contexts than just plain reactive agents. It would be very interesting to test (for lack of a better word) to what extent this formalization also holds for the MTRNN agent.

Instead of finding ways to improve the task and environment, one could also try to alter the neural network model in such ways that the multiple timescales remain intact but the agent is able to act reactively as well. For example, the MTRNN model could be expanded with a third layer of nodes that decides when the slow context units should be on (to indicate shifting of contexts) and when the fast context units are on, so that local behavior can be learned. This last suggestion is probably very difficult and too far-fetched to research, but still it would be very nice to incorporate the best of both worlds - a little like regulative control does, but incorporating multiple timescales as well.

Finally, the memory context units play a large role in producing behavior of the MTRNN agent and therefore the agent comes up with a solution that is too global and too deliberative to have the agent react well to local situations. This poses an important follow-up research question: how can we ensure that the global memory filters and remembers the information that is needed to react in a chaotic environment such as the random environment used in this experiment? To investigate this, one needs a quantifiable measure. Such a measure might be entropy (i.e. information density or the amount of chaos (Martyushev & Seleznev, 2006)) - a measure that indicates the uncertainty one has about some set of possible outcomes. The higher the entropy, the less information is present and thus the more uncertain an environment is. In random environments, the amount of possible environments is enormous and thus entropy is very high. There are two ways to deal with this problem; both focus on decreasing entropy for the agent itself. One is to increase the number of units so that more different situations and information can be stored and memory is increased. However, it is not really natural to deal with all the information in the environment; the other solution is thus to filter out the relevant information to decrease entropy in the agent itself. The goal of decreasing entropy should be that the agent will handle obstacles and food sources well by using relevant information from the past.

In any case, this research was only a little step in the direction of finding out more about the behavior, characteristics and effects of different timescales of agents controlled by recurrent neural networks. There is still a lot more interesting research to be done in this field of science.

References

Aaron, E., & Admoni, H. (2010). Action selection and task sequence learning for hybrid dynamical cognitive agents. *Robotics and Autonomous Systems*, *58*, 1049-1056.

Arbib, M. A., Erdi, P., & Szentagotha, J. (1988). *Neural Organization: Structure, Function and Dynamic*. Cambridge, Massachusetts: MIT Press.

Ardila, A. (2008). On the evolutionary origins of executive functions. *Brain and Cognition*, *68*, 92-99.

Badre, D. (2008). Cognitive control, hierarchy, and the rostro-caudal organization of the frontal lobes. *Trends in Cognitive Sciences*, *12*(5), 193-200.

Badre, D., & Wagner, A. D. (2007). Left ventrolateral prefrontal cortex and the cognitive control of memory. *Neuropsychologia*, *45*, 2883-2901.

Bax, L. (2010). *Cognitive control in reactive agents: Surviving predators through the evolution of a circadian rhythm.* Bachelor's thesis. Nijmegen.

Berger, R. J., & Phillips, N. H. (1995). Energy conservation and sleep. *Behavioural Brain Research*, *69*, 65-73.

Billingsley, P. (1986). *Probability and Measure*. New York: Wiley.

Botvinick, M. M. (2008). Hierarchical models of behavior and prefrontal function. *Trends in Cognitive Sciences*, *12*(5), 201-208.

Braver, T. S., Paxton, J. L., Locke, H. S., & Barch, D. M. (2009). Flexible neural mechanisms

of cognitive control within human prefrontal cortex. In *Proceedings of the National Academy of Sciences of the United States of America* (Vol. 106, p. 7251-7356).

Brooks, R. A. (1986). A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, *2*(1), 14-23.

Brooks, R. A. (1991). Intelligence without representation. *Artificial Intelligence*, *47*, 139-159.

Egner, T. (2009). Prefrontal cortex and cognitive control: motivating functional hierarchies. *Nature Neuroscience*, *12*, 821-822.

Ellis, J. L. (2004). *Statistiek voor psychologie deel 1 & 2.* Amsterdam: Uitgeverij Boom.

Fuster, J. M. (2001). The Prefrontal Cortex - An Update: Time Is of the Essence. *Neuron*, *30*, 319-333.

Haselager, W., Dijk, J. van, & Rooij, I. van. (2008). A lazy brain? Embodied Embedded Cognition and Cognitive Neuroscience. In P. Calvo & T. Gomilla (Eds.), *Handbook of embodied cognitive science: An embodied approach* (p. 273-290). Oxford: Elsevier.

Huq, R., Mann, G. K., & Gosine, R. G. (2008). Mobile robot navigation using motor schema and fuzzy context dependent behavior modulation. *Applied Soft Computing*, *8*, 422-436.

Kiebel, S. J., Daunizeau, J., & Friston, K. J. (2008). A Hierarchy of Time-Scales and the Brain. *PLoS Computational Biology*, *4*(11), 1-12.

Koechlin, E., Ody, C., & Kouneiher, F. (2003). The Architecture of Cognitive Control in the Human Prefrontal Cortex. *Science*, *302*, 1181-1185.

Lagarde, S. (2009). *Evolving a circadian rhythm.* Bachelor's thesis. Nijmegen.

Martyushev, L., & Seleznev, V. (2006). Maximum entropy production principle in physics, chemistry and biology. *Physics Reports*, *426*(1), 1-45.

Montebelli, A., Herrera, C., & Ziemke, T. (2008). On Cognition as Dynamic Coupling: An Analysis of Behavior Attractive Dynamics. *Adaptive Behavior*, *16*(2/3), 182-195.

Murphy, R. R. (2000). *Introduction to AI Robotics.* The MIT Press.

Paine, R. W., & Tani, J. (2005). How Hierarchical Control Self-Organizes in Artificial Adaptive Systems. *Adaptive Behavior*, *13*(3), 211-225.

Peterson, G. L., Duffy, J. P., & Hooper, D. J. (2011). Dynamic Behavior Sequencing for Hybrid Robot Architectures. *Journal of Intelligent & Robotic Systems*, *64*, 179-196.

Petrides, M. (2005). Lateral prefrontal cortex: architectonic and functional organization. *Philosophical Transactions of The Royal Society*, *360*, 781-795.

Rajah, M. N., Ames, B., & D'Esposito, M. (2008). Prefrontal contributions to domain-general executive control processes during temporal context retrieval. *Neuropsychologia*, *46*, 1088-1103.

Rostan, N. (2009). *JGAP: The Java Genetic Algorithm Package.* Available from `http://jgap.sourceforge.net/` (Version 3.4.3)

Schmidt, R. A. (1975). A Schema Theory of Discrete Motor Skill Learning. *Psychological Review*, *82*(4), 225-260.

Smith, M. A., Ghazizadeh, A., & Shadmehr, R. (2006). Interacting Adaptive Processes with Different Timescales Underlie Short-Term Motor Learning. *PLoS Biology*, *4*(6),

1035-1043.

Van Dijk, J., Kerkhofs, R., Van Rooij, I., & Haselager, P. (2008). Can There Be Such A Thing as Embodied Embedded Cognitive Neuroscience? *Theory & Psychology*, *18*(3), 297-316.

Vroon, J. (2011). *Regulated Reactive Robotics: A Formal Framework*. Unpublished master's thesis, Department of Artificial Intelligence, Radboud Universiteit.

Willems, D., & Haselager, W. (2003). Cooperative behavior in simulated reactive robotics. In T. Heskes, P. Lucas, L. Vuurpijl, & W. Wiegerinck (Eds.), *Proceedings of the 15th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC)* (p. 355-362). Nijmegen.

Yamashita, Y., & Tani, J. (2008). Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: A Humanoid Robot Experiment. *PLoS Computational Biology*, *4*(11).

Setting	Control	Perceptron	MLP	MTRNN
Population size	125	250	500	750
Representation	Integer vector	Integer vector	Integer vector	Integer vector
Mutation	Addition of value in interval [-100,100]	Addition of value in interval [-100,100]	Addition of value in interval [-100,100]	Addition of value in interval] [-100,100]
Muation probability	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$	$\frac{1}{12}$
Recombination	One-point crossover	One-point crossover	One-point crossover	One-point crossover
Recombination probability	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
Parent selection	Roulette wheel	Roulette wheel	Roulette wheel	Roulette wheel
Survivor selection	Generational	Generational	Generational	Generational
Termination condition	100 generations	100 generations	100 generations	100 generations

Table 7: The JGAP settings for the different agents as used in the current experiments

Appendices

A JGAP Settings

The JGAP package was used to evolve the different weights of the Control, MLP, Perceptron and MTRNN agnets and initial slow context states of the MTRNN agent. Weights of the Control agent had values between the interval [-300, 300] and were later transformed to double values in the interval [-3,3]. The same was done for the other three agents, but then initial weights values were in the interval [-500,500] and the initial slow context states had values between -100 and 100. Since this project uses the same settings as Lagarde (2009) and he used the default configuration, the default configuration was also used here. The JGAP settings for the different agents can be found in Table 7.

B Results

Beside the different agents and different environments, there were many more variables that could be tested. However, due to time constraints and the fact that many of these variables were already assessed in the thesis of Lagarde (2009) (although not with the Perceptron, MLP and MTRNN agent), only the effects of different environments and different ages were obtained and are listed here in more detail.

Nr of obstacles	Nr of food	MTRNN	Perceptron	MLP
0	0	125	125	125
	10	207,3458	705,6486	511,1334
	20	749,4396	748,3348	750
	30	741,42	749,9064	735,7106
	40	741,5034	749,92	743,5982
10	0	125	124,9698	124,9126
	10	104,9766	151,861	106,0946
	20	245,8366	425,5718	423,8084
	30	747,0762	700,7696	746,3166
	40	750	749,6808	733,4012
20	0	125	124,463	124,9346
	10	89,1352	108,0912	94,6444
	20	114,4408	99,2096	113,9246
	30	214,5934	609,8378	332,2796
	40	719,3690	744,747	743,379
30	0	125	117,4204	111,7124
	10	96,5286	106,4912	105,9768
	20	108,059	86,1418	97,1512
	30	107,1362	93,8474	85,48
	40	219,6644	371,8028	354,5370
40	0	125	125	124,2224
	10	111,5772	102,8402	101,389
	20	123,1578	99,3136	92,974
	30	91,4706	82,9398	76,2564
	40	97,6158	92,7918	99,5224

Table 8: Fitness values for various combinations of obstacles and food for the MTRNN, Perceptron and MLP agent

B.1 Fitness of agents in environment I

In the current experiment, the agents were tested in the first environment with different food/obstacle ratios of which in Section 3 also graphs are showed. In Table 8 and 9 you can find the exact fitness values for the MTRNN, Perceptron and MLP agent, and the Reactive, Reactive-DN and Control agent, respectively.

B.2 Fitness of agents in different environments

The performance of the different agents were also compared in the different environments. In Table 10 to 13 one can find the exact fitness values for the agents in the different environments. The first two tables show the results for the agents in the random environment,

Nr of obstacles	Nr of food	Reactive	Reactive-DN	Control
0	0	125,4982	99,0356	125
	10	749,7928	553,3584	535,6112
	20	750	749,9596	749,098
	30	750	750	750
	40	750	750	750
10	0	95,2172	98,0318	124,6348
	10	497,459	451,8964	419,0428
	20	749,4952	749,9592	705,1114
	30	750	750	747,6192
	40	750	750	748,0944
20	0	76,9814	97,1308	124,471
	10	195,0938	356,7732	294,1572
	20	694,537	749,3564	743,9784
	30	747,8064	750	666,0596
	40	749,9814	750	736,136
30	0	64,4194	96,5028	117,1528
	10	118,4094	285,0464	241,973
	20	335,4088	744,4264	730,7184
	30	668,6916	749,6958	698,2012
	40	737,011	750	576,368
40	0	56,9062	95,8948	123,8178
	10	86,3642	226,6354	195,799
	20	155,5332	709,0156	686,249
	30	352,517	744,7668	732,8226
	40	617,0698	748,5738	738,2538

Table 9: Fitness values for various combinations of obstacles and food for the Reactive, Reactive-DN and Control agent

Nr of obstacles	MTRNN	Perceptron	MLP
0	125	125	125
2	124,942	122,4864	124,5216
4	124,787	156,1016	151,3748
6	132,9438	253,8136	194,3394
8	154,8592	269,5376	329,2668
10	207,3458	705,6486	511,1334

Table 10: Fitness values for MTRNN, Perceptron and MLP agents in the random environment with various amounts of food and no obstacles

Nr of obstacles	Reactive	Reactive-DN	Control
0	125,4982	99,0356	125
2	165,538	116,5470	122,3166
4	249,4672	149,118	146,0398
6	495,4986	203,2598	199,3292
8	737,3174	312,1822	295,681
10	749,7928	553,3584	535,6112

Table 11: Fitness values for Reactive, Reactive-DN and Control agents in the random environment with various amounts of food and no obstacles

the last two tables in the structured environment. It must be noted that the tables for the MTRNN, Perceptron and MLP agent are of main interest, since they are only able to handle sign posts.

C Statistical results

In the Results section it was briefly described how the statistical significance was assessed for a specific example. In this assessment, the following formulas were used:

Deviation within groups: $s_p = \sqrt{\frac{(n_A-1)\cdot s_A^2+(n_B-1)\cdot s_B^2}{N-2}}$

Effect size $= \frac{X_{gem}(A)-X_{gem}(B)}{s_p}$

Effective sample size: $N* = \frac{1}{(\frac{1}{n_A}+\frac{1}{n_B})}$

t =effect size $\cdot\sqrt{N*}$

degrees of freedom: $df = N - 2$

Nr of obstacles	MTRNN	Perceptron	MLP
0	125	124,9908	124,9774
2	125,0352	124,6536	129,3654
4	142,410	185,6794	155,2702
6	138,3224	247,365	244,9642
8	163,8272	540,9444	569,222
10	270,4208	748,9202	747,7726

Table 12: Fitness values for MTRNN, Perceptron and MLP agents in the structured environment with various amounts of food and no obstacles

Nr of obstacles	Reactive	Reactive-DN	Control
0	125,4968	99,036	124,9524
2	165,2782	116,691	123,7764
4	244,7258	146,3906	145,5914
6	479,1526	196,5288	188,1374
8	743,9668	312,6936	299,7846
10	750	719,5816	715,6928

Table 13: Fitness values for Reactive, Reactive-DN and Control agents in the structured environment with various amounts of food and no obstacles