Evolving minimalistic control for complex behavior

Zhou Fang
Department of Artificial Intelligence
Faculty of Social Sciences, Radboud University Nijmegen
Supervised by dr. W.F.G. Haselager and dr. I.G. Sprinkhuizen-Kuyper

May 27, 2010

Abstract

Traditionally, intelligence was thought to be associated with thinking, reasoning, planning etcetera. More recently there has been an increasing interest in 'low-level' behavioral responses. It is likely that people make use of a combination of a fast, 'lower', automatic system, and an adaptive, 'higher', deliberative system. How these systems are combined to produce behavior is uncertain however. One theory suggests the existence of a minimalistic control system that enables complex behavior by biasing the automatic system. Here this theory about how automatic and deliberative structures can be integrated is investigated. This is done by simulating the evolution of simple neural networks in an environment with two different states, during which the appropriateness of an action may also be different. It is expected that purely reactive agents are not able to perform optimally in such an environment. The theory predicts that a control structure/deliberative system will evolve that is inhibitory/modulatory in relation to the automatic system. The control system does not need to be active all the time, only in situations for which the automatic system alone is not adequate. In contrast to expectations, many evolved networks did not evolve hidden units, indicating that the task may not be difficult enough. However, the contextual input units evolved to regulate behavior in a manner similar to the hypothesized control structure, supporting the theory that natural control systems are minimalistic in nature. Further research to clarify the results is suggested.

Contents

1 Introduction

In many ways people and other animals are much alike. We have the same cell structures, the same basic needs and some of the same innate behaviors. It is the extensive ability to plan, reason, etcetera that seems distinctive for humankind. Because these cognitive capacities appear to be so important for intelligence, AI programs initially focused on modeling these kind of abilities (Russell and Norvig, 2003). After some decades it became apparent however, that programs relying heavily on explicit planning and reasoning are too slow to successfully interact in the real world. People, capable of making split-second decisions, must be doing something more clever.

In response to these issues, Brooks argued in his classic paper *Elephants don't play chess* (1990) to shift focus from representation and planning to basic action and perception. He developed robots with sensing-action association layers that involved no planning at all. These robots, in contrast to their predecessors, were able to produce a variety of behaviors similar to that of biological organisms. But obviously this could never be a sufficient model for human cognition, since people are able to do things like plan, philosophize, play chess and so on.

Research in Artificial Intelligence and Cognitive Science has shown that people probably implement a combination of automatic and deliberative behavior (Evans, 2008). Such a structure seems to combine the best of both worlds; one mode is flexible and adaptive but slow and resource limited, while the other is quick and effortless but lacks control (Schneider and Chein, 2003).

If the strategy of the brain is to do some things automatically with one system and deliberately with another, the next question logically is: how do these systems interact with each other and the world?

One possibility is that the deliberative system establishes goals and makes action plans, while the 'lower' action-perception systems monitors the surroundings and action execution. This idea underlies most AI robots to date and is called the hybrid paradigm (Murphy, 2000).

Another possibility is that the deliberative system modifies the output of the 'lower' system and only plays a supportive role (Haselager et al., 2008; Miller and Cohen, 2001). It is suggested that natural cognitive systems consist of reactive layers which account for automatic behavior and control layers which inhibit or facilitate the reactive ones. Actions follow each other naturally, given the input from the environment and the control system. The main distinction with the previous suggestion is that the control system does not actually generate behavior and is not needed most of the time.

Haselager et al. (2008) suggest such a 'minimalistic' control structure is more likely to evolve. Simple organisms have a small behavioral repertoire and may survive with a reactive system only. As organisms become more complex, the coordination of their behavior becomes more difficult, requiring 'higher' systems to support good behavior. It seems natural that an additional, higher system builds on existing structures and uses these as much as possible, rather than replace it. Thus it is hypothesized that a minimalistic control structure evolved on top of reactive systems because of selective pressure and an increase in environmental complexity.

The validity of this theory is investigated here. The specific question addressed in this paper is:

Do minimalistic, modulatory control structures evolve in agents having to deal with environments where the appropriateness of actions radically changes?

The necessary factors for the evolution of minimalistic control structures have been identified as 1) the availability of several different actions to the agent and 2) variability of the appropriateness of an action depending on context. It is reasoned that given these conditions, a purely reactive agent will not be able to perform optimally and some kind of control is needed.

To simulate these conditions, neural network controllers are evolved in an environment with different states, using an evolutionary algorithm. The hypothesis is that starting from a random initial population, individuals with a minimalistic, modulatory control structure will evolve because of their advantage over other individuals under these conditions.

Specifically, the control structure is hypothesized to have the following characteristics:

1. The most frequent course of action can be performed when the control system is not active.

2. The control system modifies automatic behavior but does not generate behavior itself.

In the following section, theoretical background and simulation methods are discussed further. I will explain why minimalistic control structures are thought to exist (Section 2.1), what evolutionary algorithms and neural networks are and why they are used here (Section 2.2 and 2.3). In Section 3 all aspects of the experimental setup, such as environment, agents and conditions, are explained. In Section 4 the results of the simulations are presented and conclusions are drawn. In Section 5 the experiment, the method and the results are discussed and ideas for further research are suggested.

2 Background

2.1 Minimalistic, modifying control

Support for the existence of modifying control structures in humans can be found widely in neurological data. Several case studies and many imaging studies (Garavan et al., 2002; Menon et al., 2001) show that the prefrontal cortex (PFC) plays a key role in control/inhibition of actions and damage there can result in automatic activation of behaviors by the environment. Based on these findings Miller and Cohen (2001) describe the functioning of the PFC as follows.

> [...] the PFC units themselves are not responsible for carrying out input-output mappings needed for performance. Rather, they influence the activity of other units whose responsibility is making the needed mappings.

It has also been concluded by Sakagami et al. (2006) that the PFC influences automatic behavior by resolving competition between behaviors, supporting behavior that might be weaker given the current stimuli, but more appropriate given the context and the task at hand. The suggestion that the role of the PFC is modulatory is further supported by the observation that damage in this area does not lead to motor execution deficits (Fuster, 1997).

The existence of automatic and deliberate components in humans becomes especially apparent in patients with PFC lesions suffering from environmental dependency syndrome (Lhermitte et al., 1986; Lhermitte, 1986). Patients with this syndrome are compelled by the environment to perform certain actions and are unable to control this tendency. This results in what one might call reactive behavior. As Lhermitte (1986) puts it:

> The patients' behavior was striking, as though implicit in the environment was an order to respond to the situation in which they found themselves.

For example, patients may use any object in their vicinity, or imitate behavior of the researcher, even after being explicitly asked not to (Tanaka et al., 2000). The induced behaviors can be quite complex; upon seeing a bed in his doctors' apartment, one patient undressed and got into bed, ready to sleep.

That such complex behavior can be triggered and performed automatically is in line with Haselager et al. (2008)'s proposition that most daily actions can occur without much control. Only minimal control is required because the environment provides us with enough cues about which action to perform and how to do it. For example, many people are familiar with the phenomenon of suddenly standing in front of one's house without remembering driving there, though driving is far from being a trivial task.

Apart from being efficient, such a system is also thought to be evolutionary plausible; it is unlikely an entirely new system comes to replace the old one, given the small steps evolution is thought to take one at a time. If complex cognitive systems evolved from simpler ones, it makes sense that they would use the 'old' system as much as possible.

2.2 Evolutionary Algorithms

To simulate the evolutionary process under the proposed conditions, Evolutionary Algorithms (EA) are used (Eiben and Smith, 2008). Evolutionary algorithms are programs for finding solutions to problems using principles taken from biological evolution. The idea behind EAs is that the same mechanisms used by evolution for solving the 'problem' of survival can also be applied to many other 'problems'. EAs are often employed for solving complex problems for which no obvious or simple solutions exist. However, they can also be used to answer questions about evolution and selection, and there has been an increasing interest in EAs for answering other questions about biological organisms (Ruppin, 2002).

In general, EA's have the following form. The program starts with a number of random solutions, called individuals, which form the initial population. The individuals are coded in 'genes', which should represent all relevant characteristics of the individual. Each individual is assigned a fitness by the fitness-function, which indicates how good an individual is as a solution. The higher ones fitness, the higher the chances of surviving and reproducing. Selected individuals generate offspring by recombining their genes with others and random mutation in those genes. On average, successive populations will consist of better individuals, because good individuals have a better chance of reproducing and thus contributing their good genes to the next generation.

Evolutionary Algorithms are useful in this study, because they use the most fundamental principles from natural evolution and find individuals that are suitable for the environment. If minimalistic, modulatory control structures indeed have an (evolutionary) advantage over other architectures, the populations in evolutionary algorithms should converge to this solution.

The simulations are greatly simplified compared to reality. The goal is to test the principles of minimal control structures rather than to simulate the evolution of a specific biological organism. It can also be noted that it is difficult to say something about the factors that resulted in present biological structures, partly because of exaptation. Given the flexibility of the brain, this must be especially true for cognitive functions; structures that evolved for one purpose can come to serve another over time. Still, if the proposed control structure develops naturally under the given conditions, it provides support for the theory in addition to neurological evidence. This study differs from hand-made computational models in that it tests not only whether the model would work, but whether it is likely to come to existence.

2.3 Artificial Neural Networks

Artificial neural networks (ANN) are sets of interconnected artificial units with characteristics that are inspired by networks of biological neurons. Each unit in the network has a certain activation that may change over time as the units influence each others activity through the connections. A typical neural network has input and output units, through which it is connected to the outside world, and hidden units, which only have connections to and from other units. It is analogous to the brain of an organism receiving input from the world through its sensors, which activates certain neurons. This activation is then used to activate other neurons in the network, i.e. information being processed in the brain. Finally, those neurons activate certain motor neurons to produce output. The network's behavior is determined by the (weights of) connections between units and how the units combine incoming activation to produce output activation.

There are several reasons for selecting neural networks as controllers for evolving agents. One reason is that given the 'right' network, the agent can show virtually any behavior. This is desirable because in setting restrictions to the behavior of the agent, the results could be strongly biased towards what is expected.

Another reason to choose ANN is that they are relatively easy to represent in such a way that evolutionary operators can manipulate their components meaningfully. Biases, connection weights and other features can be changed simply by adding or subtracting values. In other words, they are fit to be evolved by evolutionary algorithms (Husbands et al., 1995). Both their topology and connections can be evolved, setting little restrictions to the inner workings of the control system as well.

Finally, in Evolutionary Robotics, a field that deals with developing robot controllers by using (forms of) EA, ANN are often used as controllers (Nolfi and Floreano, 2001). For simulated as well as real agents, neural network controllers have been successfully evolved in the past for environments similar to the one proposed. Therefore, we can be fairly confident that with the right settings, the agents will be able to carry out the task to some extent, and analyzing them will tell us something about how they are able to do this. After all, the proof of the pudding is the eating.

The evolutionary algorithm will be used to evolve both network size and connection weights. In contrast to the most common neural networks, the networks here will not learn throughout their lifetime. With other words, the connection weights of the networks do not change while the agent is running in the environment. Evolution and learning are similar in the sense that they adapt individuals to the environment, though their time span is different. Including both would make the results more difficult to interpret however and this aspect was left out in this initial study. Some more attention will be given to this issue in Section 5.

Research in neural network control structures often use fixed network sizes, comparing different types of networks. For the current study this is not adequate, because we are looking for the best possible control structure. If the network size were fixed, the connections would adapt to that particular network and all units would be likely to have some functions, even if they are not strictly necessary. Another possibility would be to run multiple EAs, with different fixed network sizes and compare these to each other. But this approach neglects the competition between networks of different sizes as presumably occured in nature.

3 Experimental setup

As explained above, the hypothesis that in an environment where control is necessary, minimalistic control structures evolve, is tested by simulation of evolution of neural networks under certain conditions. In this section the experimental setup is described in more detail. First the environment (Section 3.1) and agents (Section 3.2) are described. The evolutionary algorithm is covered in Section 3.3, and in Section 3.4 the experimental conditions are presented.

3.1 The environment

The task of the agent is to 'survive' in a computer simulated environment of 200 by 200 pixels, shown in Figure 1. The environment contains 5 areas of interest ('targets') with a radius of 10 pixels, two of which are 'bad' and three which are 'good'.

The environment is infinite in the sense that there are no walls and the agent cannot get out. If the agent moves off the right border, it will turn up near the left border and likewise for up and down. Its vision works in a similar manner; if the agent is standing in the lower-left corner and looking south, it will be able to perceive T1 (Figure 1).

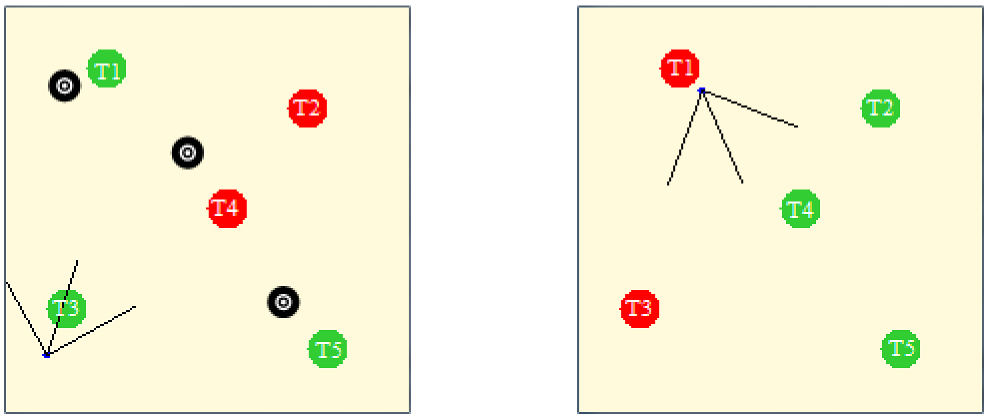

Figure 1: The environment of the agent. Light circles indicate good targets, dark circles indicate bad targets. The agent is pictured as a dot and its field of view is indicated by black lines. Everything within the quarter circle bordered by the two outer lines is visible to the agent. The small black circles indicate the three starting points of the agent. Left: environment in state 1. Right: environment in state 2.

The score of an agent for one run equals the total number of 'good' visits x reward minus total 'bad' visits x penalty. Five points are given for each time an agent visits a 'good' target, given it had not already received points for that target directly before. In other words, no points are rewarded for visiting the same target over and over again. An exception to this rule is made if the status of the target has changed since the agent's last visit. Five points are subtracted in a similar fashion for 'bad' targets.

At the end of the run, one point is subtracted for every two additional units in the network. This penalty is introduced because a small network which achieves the same score as a big network is more efficient, thus preferable. The penalty was kept small so it would only make an actual difference if the agents of different sizes scored exactly the same. Also, by only setting a penalty for every two units, there is room to explore the use of extra units without immediately being penalized.

The fitness of agents is defined as their average score in the environment over 3 runs of 500 time-steps each:

$$Fitness = \frac{1}{3} \cdot \sum_{i=1}^{3} (Score_i - \lfloor \frac{N_{units}}{2} \rfloor)$$

The agent's starting positions were predetermined at (150,150), (90,70), and (20,30) (Figure 1; left). These positions were chosen randomly with the only constraint they should be at least 20 pixels away from targets. Predetermined positions were chosen because on one hand determining the fitness on basis of a single starting position would result in agents that carry out the task 'blindly' instead of learning the general task. On the other hand, entirely random positions leave too much room for '(un)lucky' agents unless the agents are tested on a large number of starting positions. Conducting so many runs for each individual would slow down the EA considerably however. Thus, a compromise was made by testing each agent on the same set of multiple starting points.

3.2 The neural network agent

All agents are controlled by neural networks. The type of neural networks and updating rule used here, as well as the parameter ranges, are derived from Beer and Gallagher (1992) and Beer (2009). An example of the kind of networks used is shown in Figure 2. At each time-step, the network updates the activation of the input units corresponding to the agents' current position and direction. The remaining units compute their activation using the incoming activation from the previous time-step using the updating rule described later. At each time-step the output of the network is used to determine its next position. The networks always have 7 input and 2 output units. The number of hidden units can vary from 0 to 14.

The networks are fully connected, meaning every unit has incoming connections from every unit, including self-connections. Only the input units have no incoming connections since they are set externally.

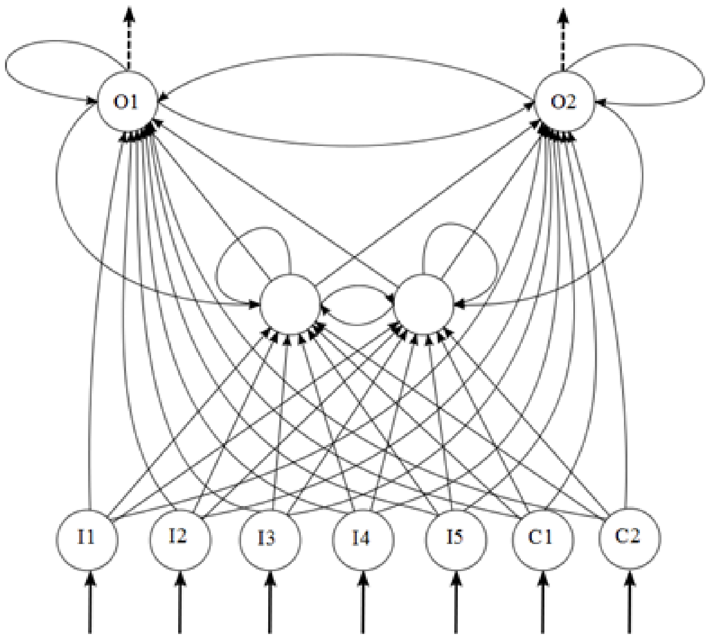

Figure 2: Example network with the required 7 input units and 2 output units. This particular network has 2 additional 'hidden' units.

3.2.1 Input

The agents receive information about the environment through 7 input units. Five of these correspond to the 5 targets in the environment and are named I1 - I5 for future reference. The activation of these units is in the range [0,1] and depends on the direction the agent is facing and the distance between agent and target. Each agent starts with a direction of 0°, which corresponds to facing southwards. The agent has a perceptual field of 90°, 45° left and right from its direction, and a sight range (*SightRange*) of 50 pixels, as depicted in Figure 1.

If a target is outside the agent's perceptual field, the activation of the corresponding unit is always 0. If a target is within the field, the activation A_{I_i} of input $unit_i$ corresponding with $target_i$ linearly decreases with the perceived distance between agent and $target_i$:

$$A_{I_i} = 1 - \frac{Distance(agent, target_i)}{SightRange}$$

The two remaining input units, C_1 and C_2, indicate the state ('Context') of the environment. If the environment is in state 1 $C_1 = 1$ and $C_2 = 0$, if it's in state 2 $C_1 = 0$ and $C_2 = 1$. The activation of all input units is entirely determined by the environment; they do not receive activation from other units or each other. Note that the agent is always able to recognize the targets because each target is assigned a different input unit, but that the input itself does not indicate whether a target is good or not.

3.2.2 Updating the network

The activity of the hidden and output units is defined as follows:

$$A_{unit_i,t+1} = A_{unit_i,t} + C * (-A_{unit_i,t} + In(unit_i))$$

$$In(unit_i) = \sum_{j=0}^{n} \frac{1}{1 + e^{-((A(unit_j,t) - BIAS) * w_{j,i})}}$$

$A(unit_i,t)$ is the activation of $unit_i$ on time-step t. $In(unit_i)$ is the total activation $unit_i$ receives through its connections from others (and itself) and $w_{j,i}$ is the weight of the connection from j to i. The weight can take all real values between -5 and 5.

BIAS and *C* are constants that are the same for all units in the network and have ranges [-1,1] and [0,1] respectively. The *BIAS* enables the units in the network to either have a higher or lower starting activation, depending on the sign of the constant. This gives the network the freedom to for example be active without any external activation to begin with. The constant *C* modifies the amount of influence the current input has on the activation of the unit. With other words, it weighs how important past activation is compared to the current percept.

3.2.3 Output

The agent's movement is controlled by the two output units, O1 and O2. At each time-step the activity of the output units, as determined according to the rule in Section 3.2.2, is taken to compute the next position of the agent.

O1 determines the speed of the agent on time t according to:

$$Speed_t = \begin{cases} 0 & \text{if } A_{O1,t} \leq 0 \\ A_{O1,t} * MaxS & \text{if } 0 < A_{O1,t} < 1 \\ MaxS & \text{if } A_{O1,t} \geq 1 \end{cases}$$

Here $A_{O1,t}$ is the activation of unit O1 on time t and *MaxS* is the maximum distance the agent can travel in one time-step, which is 10 pixels.

Similarly, O2 determines the turning angle relative to the agent's current direction $Direction_t$ according to

$$Turn_t = \begin{cases} -MaxT & \text{if } A_{O2,t} \leq -1 \\ A_{O2,t} * MaxT & \text{if } -1 < A_{O2,t} < 1 \\ MaxT & \text{if } A_{O2,t} \geq 1 \end{cases}$$

Here $A_{O2,t}$ is the activation of unit O2 on time t and $MaxT$ is the maximum turning angle of 90°, where $-MaxT$ corresponds to a 90° left turn and $MaxT$ to a 90° right turn. The absolute direction the agent is traveling in is computed as $Direction_{t+1} = Turn_t + Direction_t$. $Direction_{t+1}$ is then rescaled between 0° and 360°. For example, if $Direction_t$ is 350° and $Turn_t$ is 20°, $Direction_{t+1}$ would be 10°. The agent's next position is a shift from its current position in direction $Direction_{t+1}$ with $Speed_{t+1}$ pixels distance.

3.3 Evolving agents

For the evolutionary algorithm, the neural network agents were represented as vectors of doubles. Each double constitutes a 'gene' of the individual. The length of the genome is variable, since the number of hidden units can vary from 0 to 14. The network size thus can vary from 9 to 25 units. The first three genes always represent the following network attributes: N total number of units in the network, C and $BIAS$. The other genes represent the weights of the incoming connections for each unit.

Figure 3 shows an example genome for a network with 2 hidden units. The first three variables (11.0; 0.3; 0.4) are the values for N, C and $BIAS$. The values thereafter are weights of connections between units. First the weights of the connections to output unit O1 are represented. They come in the following order: connections from I1 - I5, C1 and C2, O1 (self-connection) and O2, and finally connections from H1 an H2. After all connections to O1 have been represented, the next N values represent connections to O2. Following O1 and O2 come the connections to H1 and H2. Because the network is fully connected, each unit always has N incoming connections. Input units need not be represented, since they have no incoming connections. The hidden units have been placed at the end of the genome so they are easily added or removed.

Network size, like the other values, is also represented as a double though it obviously needs to be discrete. The value is floored to obtain a whole number. For example, N = 11.8 would constitute a network with 11 units. If the network size decreases due to mutation, the last added unit and its incoming and outgoing connections are removed. If the network increases in size, a unit with random incoming and outgoing connections is added.

Because of an error in the program, the evolutionary algorithm rounded the network sizes instead of flooring them; this resulted in a discrepancy between the module creating the individuals and the module evaluating them. So for example if N = 11.8, the EA would create a 12-unit network, making 63 genes. Meanwhile, the evaluator would assume the individual has 11 units, thus would only read the first 47 genes. This means that a change in N has a different effect on the network than previously planned. A few evolutionary runs were repeated after correcting the error, but no noteworthy difference was found in either network size, fitness or behavior of the evolved agents.

The fitness of each agent is determined by running them in the environment 3 times

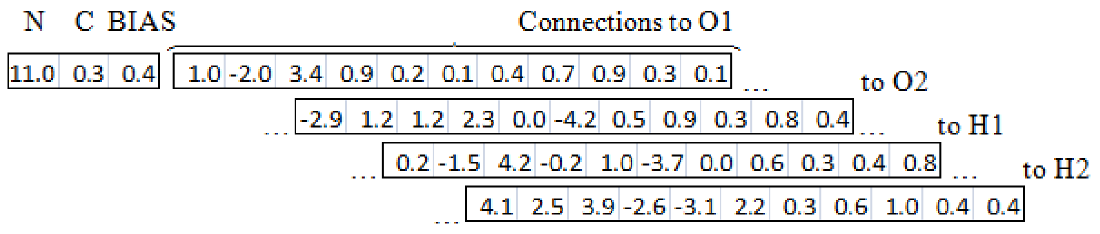

Figure 3: Example genome of a network with 11 units: 7 input, 2 hidden and 2 output units. Each row represents the incoming connections for one unit. For each unit the connections come from I1, I2, I3, I4, I5, C1, C2, O1, O2, H1 and H2, in that order. Values are rounded to one decimal here for clarity purposes.

for 500 time-steps and averaging the scores, as described in Section 3.1. In order to produce the next generation, individuals from the current population are selected to become parents using Tournament selection. This means that six individuals are randomly selected from the population and the one with the highest fitness among those becomes a parent. Using random selection means that relatively bad individuals also have a realistic chance to reproduce; they do not need to compete with the best of their population, only with the five random others. Six is a relatively large for such a tournament and was chosen because of the relatively large population size.

Individuals for the next generation are created by crossing over two parents and mutating their genes. Two parents will result in two children. Which and whether genes are crossed over and/or mutated is based on chance. For crossover the weights of the incoming connections for one neuron are taken as one block. So for example, all incoming connections of O1 are switched between parent 1 and 2. This was done because one neuron is one functional unit and switching units is expected to deliver better results than switching connections (Yamauchi and Beer, 1994). If the parents have a different network size, the smallest network is taken as guideline for how many crossovers are possible.

Uniform crossover with chance 0.33 was used. This means that for each (block of) genes, there is a 1/3 chance that the parents will swap these genes. On average, the new individual will have 1/3 of the genes of one parent and 2/3 of the genes of the other parent. After crossover, every gene is mutated with a chance of 0.1. If a gene is mutated, a random number from a Gaussian distribution with standard deviation 1 and average 0 is taken and added to the gene.

An overview of the most important settings is given in Table 1. Pilot runs were conducted to assess the effect of different parameter values. Tests were done varying mutation rate, crossover rate, population size, tournament size, network size penalty, target rewards/penalties and number of generations. In general making these parameters slightly smaller or larger seemed to have little positive effect. Only population size, number of generations and tournament size were increased compared to the initial settings, because the fitness of the agents under these conditions was found to be significantly higher.

For the implementation of the evolutionary algorithm, code was used from the freely available ECJ package [1].

Parameter	**Value**	
Population size	200	
Generation#	300	
Genome size	[21, 453]	
Bias	[-1, 1]	
C(onstant)	[0, 1]	
Network weights	[-5, 5]	
Operator	**Type**	**Value**
Selection	Tournament	6
Mutation	Gaussian	Chance = 0.1
		Average = 0
		Stdev. = 1
Crossover	Uniform	Chance = 0.33

Table 1: Overview of the settings of the evolutionary algorithm.

3.4 Experimental conditions

The evolutionary algorithm was run under three different conditions. In the experimental condition A, the environment is in state 1 (Figure 1; left) 80% of the time and in state 2 (Figure 1; right) 20% of the time. Specifically, between time-steps 200 and 300, some targets that were good become bad and vice versa. Thus, the agents need to adapt their behavior to the changing environment. Logically, there is no need to develop an entirely new behavioral repertoire. The agent can make use of old behaviors such as ‘approaching’ or ‘avoiding’ a target; it only needs to modify which target to approach and which to avoid.

As control conditions, the simulations were also run for environments that were in either state 1 (condition B) or state 2 (condition C) during the entire lifespan of the agent.

All agents ‘know’ in which state the environment is through the activation of the C-units. If the environment is in state 1, $C1 = 1$ and $C2 = 0$. In state 2 the activation pattern is reversed.

[1] Available on http://cs.gmu.edu/ eclab/projects/ecj/

The experiment was repeated for 10 different randomly initialized populations to obtain reliable results. The resulting networks from the three conditions were compared in fitness, behavior and network dynamics.

4 Simulation results

In this section the results of the simulations are presented. First the results are evaluated by looking at the development of fitness over generations and the behavior of the evolved agents from condition A, B and C (Section 4.1 and 4.2). Then the networks underlying the behavior are studied in more detail in Section 4.3 and 4.4. A summary of the results is given in Section 4.5.

4.1 Fitness

The development of the best and average fitness of the populations over generations is shown in Figure 4. Because there is a difference in the highest obtainable scores between conditions, the fitness has been normalized using the maximum score. It is improbable any agent would actually achieve this score, since the agent would have to act perfectly efficient in every time-step. The maximum score was 356 for condition A, 356 for condition B and 413 for condition C (see Appendix A how these scores are obtained). Table 2 shows the average score of the agent per time-step.

	Average in state 1	**Average in state 2**
Condition A	0.1529	0.07333
Condition B	0.2022	
Condition C		0.1994

Table 2: Average points per time-step obtained by the best agents from each condition, averaged over 10 evolutionary runs

Figure 4 suggests that the evolved agents from all conditions have learned to cope with the environment to some degree. Comparing the results from condition A with B and C, it is clear that the fitness in condition A is lowest. The task is more difficult in condition A, but the agents should have achieved scores similar to other conditions in the end, since the fitness is shown in percentage of what can be obtained. The highest fitness in condition A rises steadily until approximately generation 170, but then it evens out. The average fitness of the population keeps growing until the 300^{th} generation and presumably thereafter, which suggests that the population is converging to one solution.

There is also a considerable difference between condition B and C in fitness. However, this gap seems to be closing, as the best agents from C continue to improve even

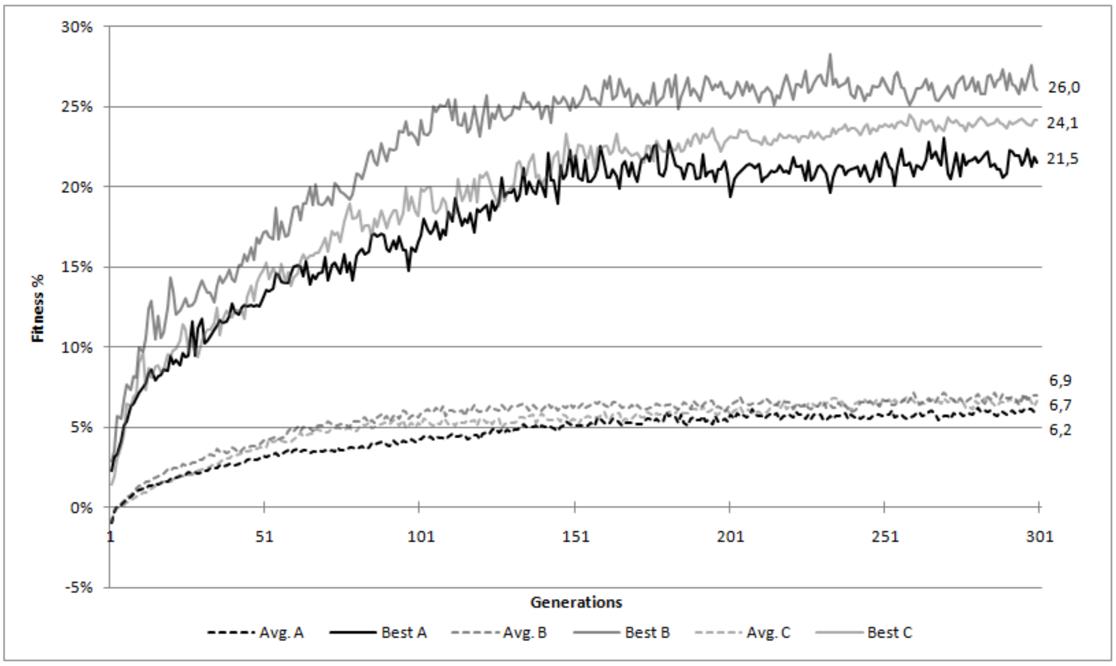

Figure 4: Mean and highest fitness for conditions A, B and C over generations, normalized to percentage of the maximum obtainable score in the condition. Results were averaged over 10 runs of the EA.

close to the 300^{th} generation. Thus, there may be no difference if the EA is run for more generations.

Running the EA for more generations is unlikely to deliver fitter individuals for conditions A and B, given that in those conditions the best fitness is no longer rising in the later generations.

4.2 Behavior

To determine how and to what extent the EA evolved agents that can handle the task, the behavior of the agents was studied. For this analysis, the best agent of the last generation of each evolutionary run was used. It is reasonable to assume that these agents represent the solution each evolutionary run has found for the environment and will converge to.

First the agents from condition B and C are described, then the agents from condition A.

4.2.1 State 1 or 2

In condition B, the environment is always in state 1. Figure 5 B1 and B2 show the behavior of a typical agent from this condition in the first 200 time-steps. The agent starts at position (20,30) and starts to travel in a circle. It quickly encounters target T5 and slowly approaches the good targets by using the corners. It takes some more time-steps for the agent to fine-tune its behavior (Figure 5 B2). The path of the agent varies very little once it has found a route. Between which targets the agent travels often depends on the agents' starting position. Most agents tend to travel between T3 and T5, which is also the optimal solution for state 1: the minimal distance between them is 53 pixels (6 steps), whereas this is 62 pixels for T1 - T5 (7 steps) and 88 pixels for T1 and T3 (9 steps).

In condition C, the environment is always in state 2. The strategy of these agents is quite similar to those from condition B; they have one preferred route and hardly deviate from it once it has been found (Figure 5; C1 and C2). The scoring mainly depends on how fast the agent is able to find the right targets and route. Many agents travel the most efficient route (T2 - T4), but other routes are also observed.

Overall, agents from both condition B and C learn the task reasonably. However, their path is not optimal, because they tend to travel in circles rather than in straight lines. Some also take unnecessary detours while traveling between targets.

Most agents scored less when put in novel starting positions, and some scored much worse. This seems mostly to be caused by the fact that many agents search for targets by circling around and do not actually cover a lot of distance. If the targets are not where they are expected, it can take a long time to find them (Figure 5 B3).

Though the agents are not able to score efficiently from every starting position, they are able to avoid bad targets from any position. It is harder to visit a target than to avoid it; avoiding can always be accomplished by turning away sharply, but to visit a target the agent needs to know whether it needs to go left, right or straight ahead. This is information is not directly available to the agent, making it harder to take the appropriate course of action.

4.2.2 State 1 and 2

In condition A, the world is in state 1 during time-steps 0 - 200 and 300 - 500 and in state 2 during time-steps 200 - 300. Figure 6 (A1 and A2) shows the behavior of a typical agent from condition A in state 1. The behavior of the agent in state 1 is similar to agents from condition B: it travels in circles and slowly approaches good targets, taking some additional steps to find an efficient path. Note that though most evolved agents have similar strategies, the actual behavior can vary considerably over different evolutionary runs (Figure 6; A4 and A5).

During time-steps 200 - 300, the world changes to state 2. As can be seen in Figure 6; A3, the agent's behavior in state 2 is not similar to those of the agents from condition C. There is visible difficulty in approaching the good targets and most agents only score a few times. They often miss targets and tend to circle around them instead of over them. This is curious since no points are rewarded for being close to a good target. But

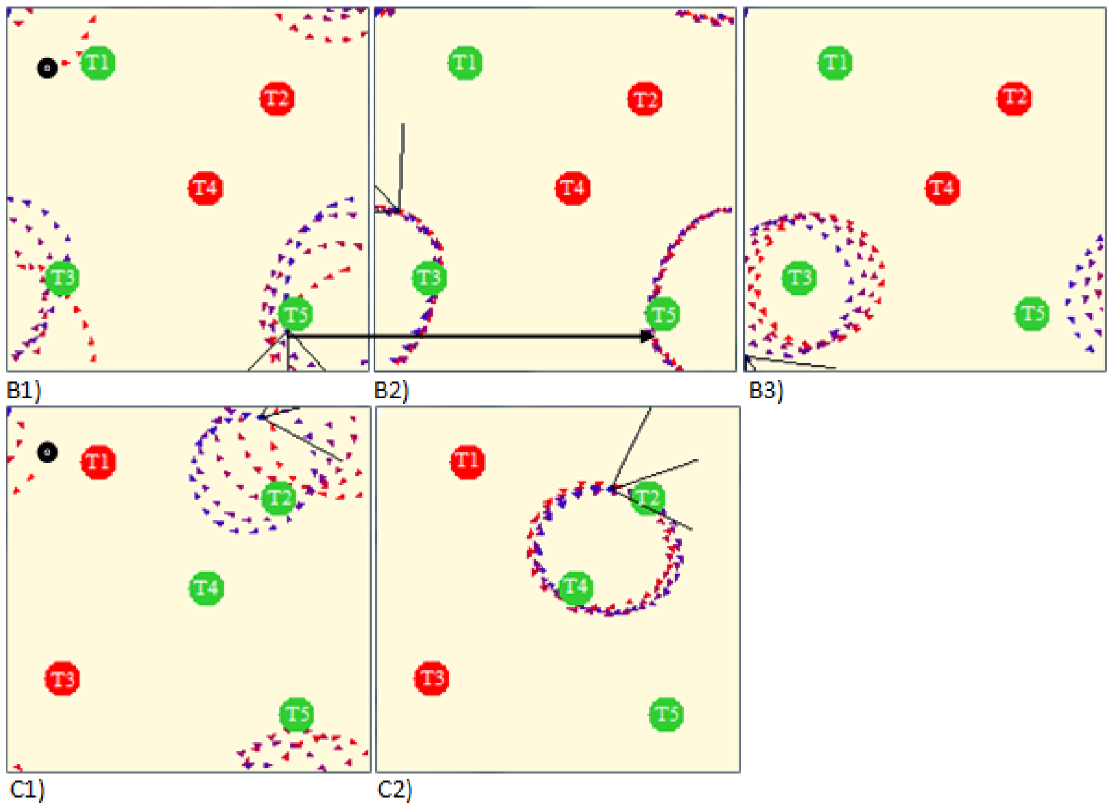

Figure 5: B: Path of best scoring agent evolved in condition B. The small black circle indicates the starting point. Triangles indicate steps of the agent. B1: agent steps from time 0 to 100. B2: path continued from B1 for time 100 to 200. In the first 100 steps the agent does not score much, trying to find the best route by circling around. Once it has found its route, its behavior hardly changes and it scores very quickly. B3: Path of the agent for time 100 - 200 when starting from a novel position. In contrast to B2, the agent is not scoring yet on time-steps 100 to 200. Its search strategy appears to be inefficient for novel positions. C: Typical agent from condition C. Starting point (20,30) as indicated by the small black circle. C1: path of agent in time 0 - 100. C2: path of agent in time 200 - 300.

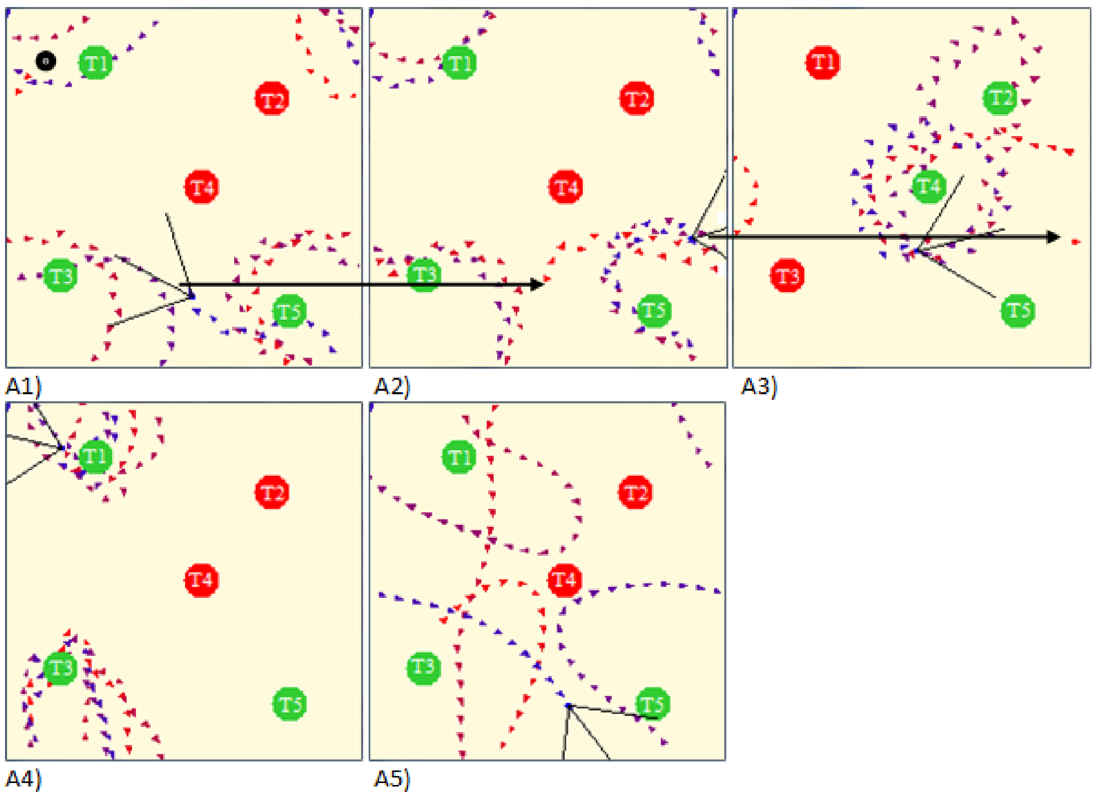

Figure 6: Example behaviors of agents from condition A. The small black circle indicates the starting point. Triangles mark steps of the agent. A1: path of a typical agent from time 0 to 100. A2: path of agent continued from A1 for time 100 to 200. A3: path of agent continued from A2 for time 200 - 300. A state change occurs on time 200. A4: path of best scoring agent from condition A for time-steps 100 - 200. A5: path of worst scoring agent from condition A for time-steps 100 - 200.

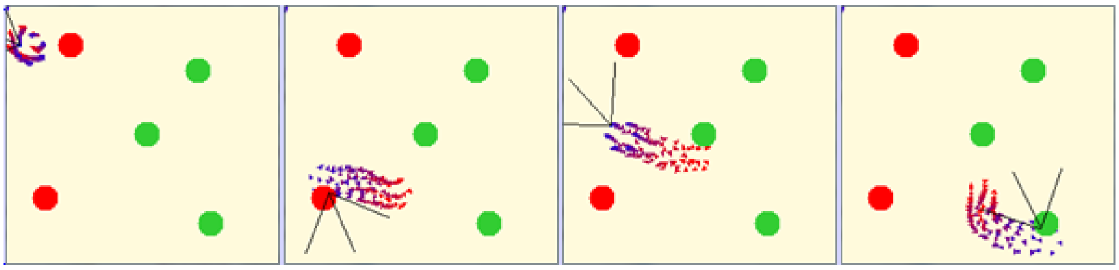

Figure 7: An ambiguous agent from condition A in state 2, shown from 4 different starting positions. There appears to be no clear strategy.

if they are close to the right targets, there is a higher chance they score if they drift off from their usual path. Notably, unlike condition B or C, agents from condition A tend to deviate from paths they are following. This enables them to deal better with novel starting positions than either agents from B or C, but they also score less efficiently.

A striking observation is that some agents have much higher scores in state 2 than others, while others have much higher scores in state 1 compared to the former (Table 3). Behavior analysis shows that this is due to different survival strategies: in 5 of the 10 evolutionary runs, agents evolved that had a clear reversed attraction pattern in the state 2 compared to state 1. In state 2 they start to circle T2 and T4, which had been bad in state 1. For the best agents from the remaining evolutionary runs no such behavior was observed. Three of them dealt with the change by circling in a small, restricted area until the alternative state had passed. Effectively, they were 'waiting' for the infrequent situation to go away. The two remaining runs resulted in ambiguous agents who did move around, but with unclear underlying strategy. An example of this is shown in Figure 7.

	Average in state 1	**Average in state 2**
Overall	0.1529	0.0733
Strategy for both state 1 and 2	0.1183	0.1267
Specialized in state 1	0.1913	0.02361

Table 3: Average points per time-step obtained by the best agents from condition A. Scores are averaged over 10 evolutionary runs and grouped according to the agents' strategy.

Though the task was designed to involve agents that show 'appropriate' behavior for both state 1 and 2, the agents specialized in state 1 also have a valid survival strategy. Their behavior may not be as expected, but their behavior in state 2 differs from state 1 and is good enough to get by. This is shown through the fact that agents that could score in both states did not have a higher fitness than the other agents. There appears to be a trade-off between doing okay in both states and doing well only in state 1; the former score much less in state 1 and much more in state 2 on average than the latter (Table 3). Which strategy the evolutionary algorithm converges to depends on the random initial population.

In summary, the agents from condition A learned to deal with the changing environment one way or another. Two different strategies are observed: one strategy is to focus attention on state 1, another is to try to cope with both. But even agents with the second strategy were better in state 1 than state 2; in state 1 their behavior is similar to agents from condition B (state 1 only), but their behavior in state 2 is not similar to those from condition C (state 2 only). Logically, state 1 is more frequent than state 2, so more can be gained by scoring well there. Still, agents from the non-changing conditions were on average more efficient in scoring in their respective states than agents from the changing evolutionary runs (Table 2). The latter deviated more from their path between the targets compared to the former.

4.3 Network size

After looking at the behavior of the agent, we continue to look at the networks underlying the behavior. First the development of network size over generations is examined. The mean network sizes of the best individuals do not show differences depending on condition (Figure 8). This is unexpected since condition A is more difficult than B or C.

The best networks from the last generation of the evolutionary runs have an average size of approximately 9.5[2]. There appears to be a large preference for small networks, despite the fact there was only a 1 point penalty per 2 additional units (for comparison: visiting a good target once is worth 5 points). This indicates that little additional gain was found in acquiring additional units.

Figure 9, presenting the fitness of agents according to network size, also shows no clear indication that hidden units have a positive effect on fitness. However, no conclusive argument can be made since the number of instances for networks with 1, 2, 3 or more hidden units is very small compared to those with 0 hidden units.

Lastly, agent strategy, as discussed in the previous section, does not appear to influence network size. Only 1 of the 5 agents that specialized in state 1 had hidden units. The same ratio holds for the other group of agents.

4.4 Revising the hypothesis and further analysis

To gain a better understanding of the inner workings of the agents from condition A, units were disabled to make their effects on behavior visible. If the disabled units form

[2]The majority of agents have 9 units (5 input, 2 context and 2 output), but the average is relatively high because there are also 12-unit networks in the population

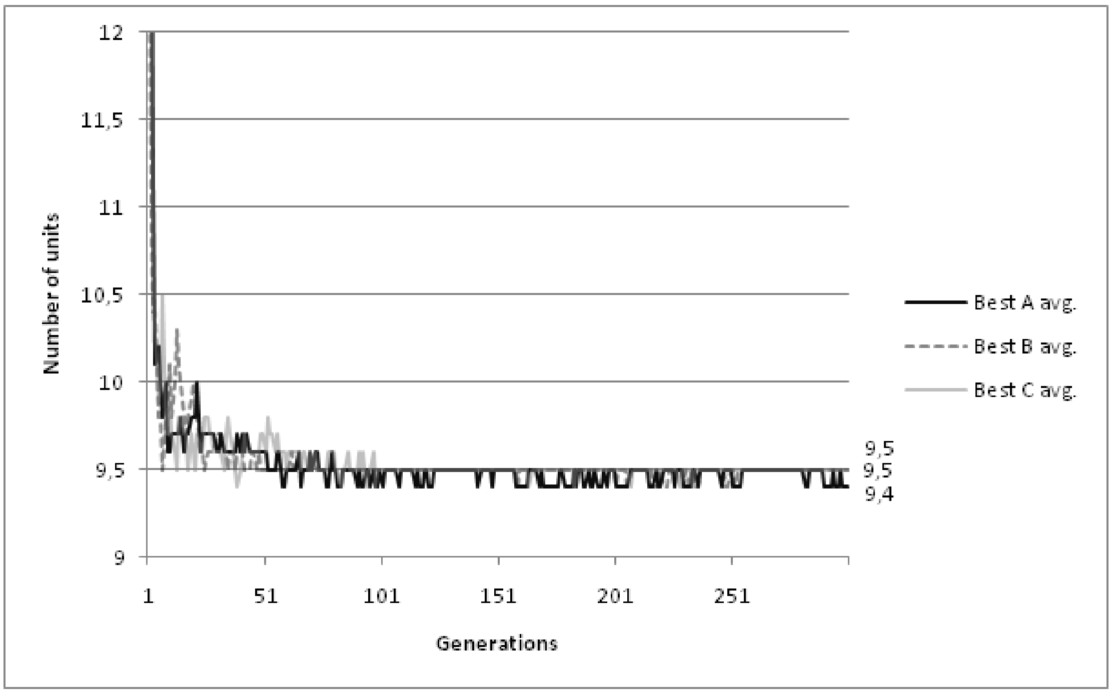

Figure 8: The network size of the fittest individuals for each condition, averaged over the 10 evolutionary runs.

the hypothesized control structure, the most frequent behavior should occur with little problem, since automatic behavior can occur without control. However, only 2 out of 10 best agents from each evolutionary run possess hidden units. Tests show that deleting any of these units has a destructive effect on behavior. Disabling these units (i.e. set their activation to 0)[3] usually also has a destructive effect, though disablement of some hidden units did not change behavior considerably. Given that none of the hidden units fulfill the requirements of a minimalistic control structure, one possibility is that the hypothesis is incorrect. However, it is also possible that the hidden units are part of the automatic system and the ability to show alternate behavior lies somewhere else. The latter is supported by the observation that many evolved agents show variable behavior dependent on the state of the environment without any hidden units. Thus it is hypothesized that perhaps in this particular task the hypothesized control structure is not to be found in additional structures, but in the C-units instead.

The C-units are different from other input units because they are not directly part of the agents' sensory system and are not under the agents' control. They are the output of more elaborate processing, through which the agent is able to recognize that the envi-

[3]The effect of activities of units being zero is not the same as deleting them; if the network has a non-zero bias, it will propagate 0 - BIAS activity through its outgoing connections if activity is 0.

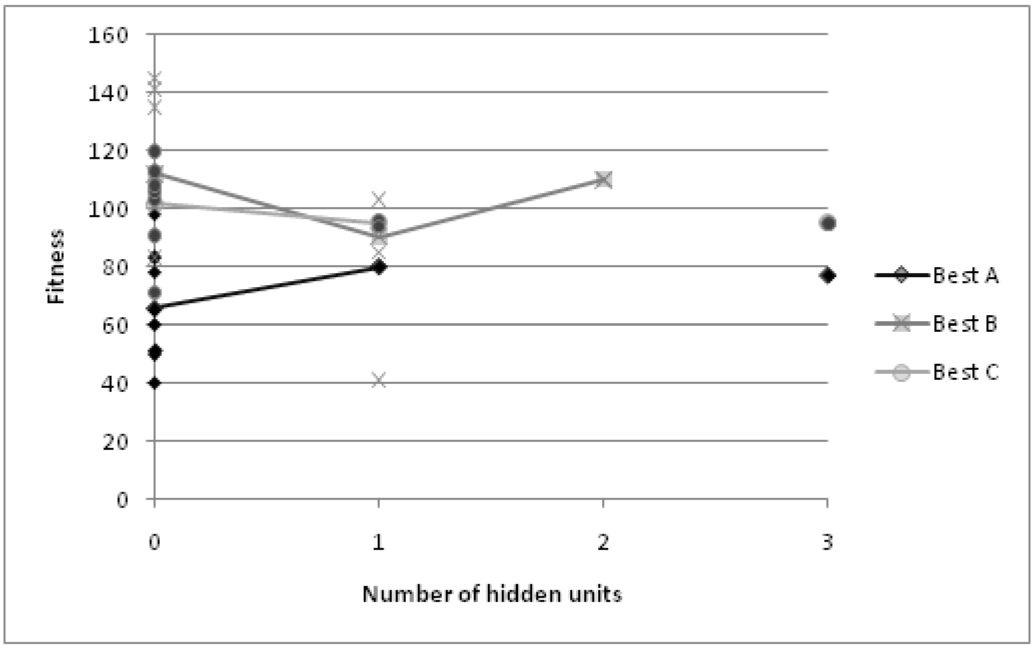

Figure 9: Fitness of the best agents from the evolutionary runs split in groups according to network size.

ronment has entered another state or symbolize a different internal state. For example, they could represent the difference between day and night, rain and dry season, hunger and thirst, etcetera. Koechlin et al. (2003) make distinctions between three kinds of input in their proposed functional architecture of the prefrontal cortex. The model is depicted in Figure 10. Through a series of imaging experiments they find supporting evidence for the theory that the PFC is organized as a cascade of processes that control behavior according to stimuli, perceptual context, and temporal episode in which the stimuli occur. In terms of the current study, the I-units form the current stimuli, whereas the C-units form the contextual signals.

With other words, there is a system of I- and O-units that always exhibit certain behavior given certain stimuli, whose output is modified by the context units using other (contextual) knowledge. In this case, the task may be simple enough for the context units to directly influence output, rather than go through intermediate units.

To test this theory, the agents were put in the environment while the context units were both disabled by setting them to zero[4]. As in the experimental condition, the environment was in state 1 for 400 time-steps and in state 2 for 100 time-steps.

As can be seen in Table 4 row 1 and 2, the agents' fitness drop drastically for both

[4]Tests were also done deleting one or both C-units, but it had a much larger decremental effect on the agent's behavior.

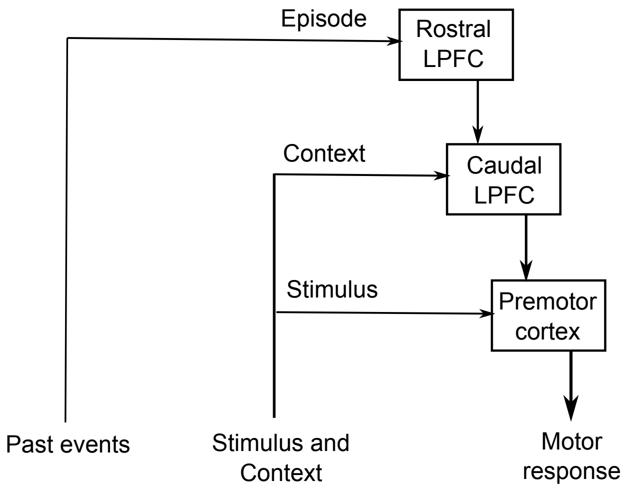

Figure 10: Functional model the prefrontal cortex of Koechlin et al. (2003), adapted from Koechlin et al. (2003)

state 1 and 2 when the context units are not functioning. Given that rewards and penalties are worth 5 points, the values appear to indicate that the agents' behavior is back to random level without the C units. From the agents' behavior it can be concluded however, that most agents still show fairly intact strategies that can be recognized as similar to their typical behavior in either state 1 or state 2 (Figure 11). The agents were categorized systematically by testing them for a number of starting positions and observing their search behavior and attraction/ repulsion towards each target. From the 10 'best' agents, 6 have been classifiedas showing state 1 behavior (group 1), and 4 as showing state 2 behavior (group 2). The average scores of these two separate groups is shown in Table 4 row 3 and 4.

	state 1	**state 2**
Normal	61.167	7.333
C1 and C2 always 0	4.167	1.167
C1/2 = 0, Group 1	18.611	-1.389
C1/2 = 0, Group 2	-17.5	5

Table 4: Average total score of best agents from the 10 evolutionary runs when C1 and C2 are always 0. The scores are also split out for agents that are obviously attracted to state 1 targets versus agents that are attracted to state 2 targets. Note that state 2 occurs less frequently than state 1, so the absolute scores are necessarily lower. Of importance is not the difference between the states however, but the difference between the normal and disabled group, and group 1 and group 2.

As one can see, there is a considerable difference between the two groups in scoring

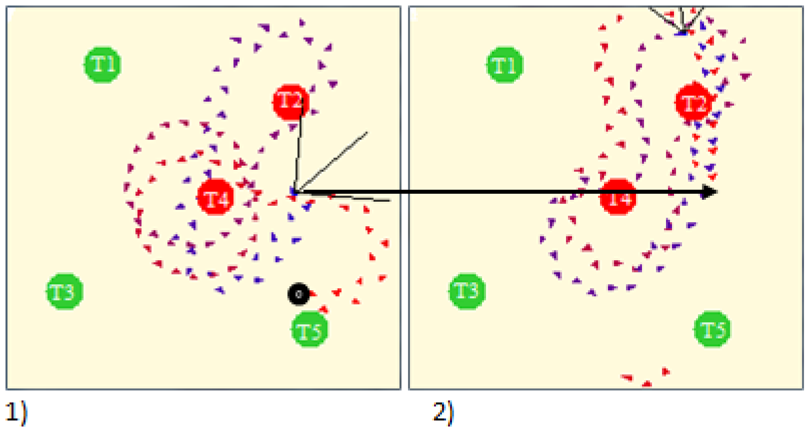

Figure 11: Behavior of a typical evolved agent from condition A after both C-units have been set to 0. It is observed the agent shows behavior appropriate for state 2. The small black circle indicates the starting point, which is at position (150,150). Left: path of the agent for time-steps 0 - 100. Right: path of the agent for time-steps 100 - 200.

in the two states. In other words, from disabling the C-units it can be observed that some agents show automatic behavior that is appropriate for state 1, while others show appropriate state 2 behavior by default. An agent that shows state 2 appropriate behavior is shown in Figure 11.

The fact that the agents' default behavior could either be state 1 or 2, can be explained by the fact that there was no penalty for the control system being active and no noise or malfunctions were present. Thus, there was no reason to match the default behavior to the most frequent situation. In real environments, where these factors do play an important role, the agents' default behavior is more likely to match the most frequent state.

4.5 Summary

In all conditions agents evolved with specific strategies that enabled them to score in the environment. Agents from condition B scored better than those from C and much better than those from A. Expected was that the task in condition A would require networks with additional structures in to show optimal behavior. Most agents did not evolve hidden units however, irrespective of condition. The hidden units of the remaining few networks do not show the characteristics of a minimalistic control system. This means that either the theory is incorrect or no hidden units are needed to complete the task. Since networks with hidden units did not score better than those without, the latter is more likely to be true.

Given these results, it is hypothesized that for this task, the C-units might be sufficient as ‘control system’. Since the C-units are set externally, this would mean that the environment is providing enough cues for the agent and no internal control system is needed.

By manipulating the activation of the C-units it is shown that the agents have a ‘default’ preference for either state 1 or state 2 behavior. This shows that automatic behavior is coded within the input-output mappings alone. Since the weights of the connections between input and output units are non-zero and the activation cannot be blocked, the C-units can only have a modifying effect on the behavior of the system. With other words, the observed behavior is realized by modifying the existing perception-action system.

Though the overall behavior remains intact, fitness does decrease significantly for both states if C-units are not functioning. With other words, action execution is not unaffected. The main cause of the decrease in fitness appears to be the loss of accuracy. Searching for targets also is less effective. No preference was found for matching the automatic behavior with the most frequent state. The most obvious reason for this is that there were no disadvantages for performing the infrequent action automatically and the frequent one by using control, as might be the case in reality.

In summary, the results show that networks evolved that made use of external cues by using the C-units as control system, instead of evolving additional units. For the current level of performance no additional system appeared to be necessary. No minimalistic control structure such as hypothesized has evolved, but the C-units do show similar characteristics. The C-units only modify current mapping instead of generating new behavior and they are not needed for one of the two diffferent behaviors. Actions are found to be less effective however if executed when C-units are inactive, which indicates they were also necessary for fine-tuning behavior. The results are in line with minimalistic control theory, showing that appropriate behavior for one state of the environment tends to become coded automatically, while the other can be realized by modifying the automatic system.

It may be unfortunate that the C-units functioned as control system, because the setup was such that they could not be influenced by the network itself. The networks do not tell us how the activation of context units develops over time. Their activation is set at a constant value and is always correct. Moreover, disabling the C-units automatically means that the network is no longer capable of judging the difference between state 1 and 2. This means that the found effects may be artifacts of the minimal size of the networks. In the next section this issue, among others, is discussed further.

5 Discussion

The results of this study indicate that under certain circumstances, agent controllers evolve that make the existence of minimalistic control structures plausible. It was found that the pregiven C-units evolved to take the function of and characteristics similar to a minimalistic control structure. In this section I will first discuss the implications of these results. Then the most important discrepancies between hypothesis and results are discussed (Section 5.2). The validity of the experiment will be discussed in Section 5.3). In Section 5.4 further research is suggested that could shed more light on the evolution and nature of (biological) control systems in addition to the current study.

5.1 Implications

Minimalistic control theory can explain the neurological data found in patients and healthy subjects on the issue of control. Though the hypothesis was not fully confirmed, several observations from the results support the theory. First of all, the network appears to function in a minimalistic manner. Apparently opposite reactions to the same stimuli were realized by modification of the automatic response depending on context, rather than coding for all behaviors directly. It shows that 'control' does not per se mean to be in full control of actions and the same perception-action system can react very differently but in a meaningful manner when biased a little. This strategy is not only fruitful, but the fact that many networks evolved with these characteristics also shows that it may likely evolve under certain conditions.

The results underline the power of making use of available information, rather than 'complex' reasoning. This is in line with other research in reactive/non-reactive agents. van Dartel et al. (2005) for example, found that even reactive agents could cope with perceptual ambiguity in an active categorical perception task. They used the environment as external memory to compensate for the lack of internal memory. Here similar function of the environment was found in the light of control. The activation of the C-units is set externally, so one interpretation may be that the agent is using a kind of external control: decisions are made 'by' the environment instead of an internal control system.

Finally, the results indicate how we may proceed further in testing the hypothesis that minimalistic control structures have an evolutionary advantage. It was found that I-, C- and O-units alone were sufficient for the current task. This system matches the lower two modules in Koechlin et al. (2003)'s model (Figure 10). In order to make allow more elaborate structures to evolve, the task must become more demanding. Given the model, it makes sense to include information about past events as a factor of importance for determining the appropriateness of actions.

5.2 Discrepancies between hypothesis and results

Some differences exist between expectations and results. Here these differences and possible underlying factors are discussed. Suggestions for testing these explanations and determining the cause of discrepancies are given in Section 5.4.

The most obvious discrepancy between the hypothesis and the results is that most evolved networks do not possess hidden units. If no additional units evolve, there can be no additional control system. There are two main explanations for the absence of additional units: 1) the task could be performed satisfactorily with the pregiven units alone or 2) the evolutionary algorithm was unable to find better (bigger) networks though they exist. On one hand, there appears to be no positive effect of increased network size in the evolved networks (Figure 9). On the other hand, fitness and behavioral analysis shows that the evolved agents do not perform optimally.

Another discrepancy is that no system could perform effectively in either state without the C-units. If the C-units would function exactly like the hypothesized control system they would not be needed at all in one of the states. As mentioned before, the use of the C-units as control system complicates findings because their activity is set externally and constantly. This fact may explain their constant use: their activity always needs to be integrated into the network somehow. Thus results are expected to conform more to the hypothesis if control is performed by units whose activities can be influenced by the network. Another possible explanation is that the theory about minimalistic control needs revision or is incorrect. The theory was partially supported by findings, but the similarities found could also be an artifact of the simplicity of the evolved networks.

Thirdly, it was found that some networks choose state 2 for their automatic behavior though state 1 is the most frequent. The reason for this is probably that there is no gain associated with performing the most frequent action automatically. Whether this explanation is sufficient can be tested by associating a cost with use of the control system. This is only possible if the activation of the system is under control of the network, however. In reality there is also likely to be a cost associated with occupying the control system (unnecessarily).

Lastly, 50% of the evolutionary runs (in condition A) resulted in networks that do not have a ‘good’ strategy for state 2. They compensate waiting in state 2 by performing more efficiently in state 1. This is not actually in clash with the theory, but the existence of two different strategies may be thought to be problematic. However, in this study this strategy is not found to make a qualitative difference for the results. In both strategies behaviors change when the state changes and both are valid for survival. Doing nothing is also observed in nature. Think for example of animals going to sleep at night rather than dealing with a changed environment they are not suited for. Still, a suggestion to make sure the agent performs the task as intended in the future is to change the fitness function such that only agents with high scores in both states can have high fitness. This eliminates the possibility of waiting as a valid strategy.

5.3 Validity of results

A number of questions may be raised concerning the generalizability of current results to more realistic environments and agents. One such issue is that there necessarily is a gap between the studied system and the real system when using simulations. The advantage of simulations is that all factors are under control and it is possible to study systems we otherwise could only reason about. The kind of evolutionary research conducted here for example, would not be possible without it. The downside is that simplifications are necessary and wrong assumptions about which properties are important may deliver unrepresentative results. The environment is much simplified compared to real environments. And though evolutionary algorithms have some obvious similarities with natural evolution, there are also many dissimilarities. The same is true for artificial and biological neural networks. Therefore, conclusions extending to biological organisms must be drawn with care.

Consider for example the differences between simulated and physical agents. From the field of Robotics, it is well-known that controllers from simulated environments tend to break down if directly transferred to real environments (Floreano and Mondada, 1994). It is not our purpose to design a controller for a functional physical agent, but it cannot be guaranteed that simplifications in simulated input and output do not qualitatively influence the results. Past experiences have shown that evolved agents, both physical and simulated, tend to use characteristics of the environment and body that researchers had not expected. In physical agents these characteristics are real, but in simulated agents they may be unintended artifacts of simplications in the environment. Because it is so difficult to predict which elements are of importance, deciding what may be simplified and what not may be beyond our knowledge. This problem can be avoided partially by using physical agents that act in the real world. The disadvantage is obviously that not as many controllers can be evaluated compared to simulated agents.

However, it would remain problematic to ascribe certain characteristics to biological systems based on the results of the evolutionary simulation alone. The fact that the evolutionary algorithm has found a certain solution does not mean that it is the same solution natural evolution has found. Logically, there may be many control systems that can account for the same behavior. To say that something *can* evolve is very different from saying it *did* evolve. It is impossible to simulate biological evolution as it occurred, because even very critical things depend on chance. This is also observed in EAs: a few details can have a considerable effect on what evolves and what is lost in the process (Bullinaria, 2001). So in that sense even a very detailed simulation of the world and organisms may not give the absolute answer. Optimizing an EA to find the best solutions does not give us more certainty in this matter either, since natural evolution is not guaranteed to have taken the optimal path. But it is also not necessary to simulate evolution as it occurred in order to answer questions about biological organisms. Here a theory about the conditions in which organisms could have evolved is tested. While on its own perhaps not conclusive because programming details may have biased the results unintentionally, more experiments in different environments and with different agents may provide more insight and certainty in the conditions in which certain control systems evolve.

Finally, one important simplification in the current study is the exclusion of learning. Most biological organisms are capable of some form of learning, even if they are very simple. It may not be per se that individuals that are born very fit have the greatest advantage, but individuals that can learn to be very fit. This has also been termed the Baldwin effect. Being able to learn throughout one's lifetime has the obvious advantage of being able to adapt to unexpected changes in the environment. The current experiment did not involve learning because it would have complicated findings unnecessarily for this initial study. The effect of learning on evolved control systems may be great however, since the advantage of a control structure is precisely increased flexibility in behavior. By letting out learning, an important advantage and factor in the evolution of control structures might be ignored.

5.4 Future research

The results are not conclusive in answering evolutionary questions about control systems, and further research is necessary to establish the presence and need for minimalistic control structures. Here I will discuss some possibilities for further research.

It has been suggested that no additional structures evolved because the current task does not require other units and the C-units are sufficient to change behavior appropriately. Internal control structures are expected to evolve if the environment is more complex. Conducting the current study on multiple levels of increasing complexity will give more insight in general principles involved in evolved control structures. The complexity can be raised by enabling the agent to perform more actions, other than navigational, and add more constraints to the required actions, for example that certain actions are performed in certain order. Whether and how results scale up is an important issue: generalizations from simulations to biological organisms can be argued for only if the qualitative trend in the results remains the same when the complexity of the task is changed.

Moreover, the results of more complex tasks would also be of interest to the field of Robotics. Up to this point, most research in evolution of artificial agents has focussed on small 'problems' that may no longer be of interest from an engineering point of view. However, there may be problems in making the environment and agent's body highly complex while the controller has to start from scratch. According to the Embodied Embedded Cognition point of view, the body is an important part of the cognitive system (van Dijk et al., 2008). Evolutionary studies also show that body and brain co-evolve; it is unrealistic to put a 'stupid' brain in a high-capacity body and complex environment (Floreano and Mondada, 1994). An option would be to evolve body and mind at the same time (Hornby et al., 2001).

To exclude the possibility that the evolutionary algorithm may converge to small networks too quickly, some changes to the EA are suggested. The present network size penalty is too small to explain the fast convergence to small networks alone. Large networks are not only disadvantaged by the penalty, but also because they have more weights that need to be set sensibly. This costs more time, during which small networks can gain the upperhand. This is amplified by the fact that if a network increases in size due to mutation, the weights of the additional unit are initialized randomly. In general, Floreano et al. (2008) observed that when evolving network topology and weights, changes in the topology usually result in lower fitness even if they could increase fitness later on.

Suggested is to generate the maximum number of genes for all networks and add a gene for every unit that indicates whether the unit is active or not. The number of units can be evolved freely and any hidden unit can be deactivated. But units and their connections are not lost when the network decreases in size. Individuals in later generations can utilize units abandoned by their ancestors, thus also giving these units more chance to evolve connection weights that make them useful.

Given the purpose of this study it is not recommended to explicitly favor larger networks in the beginning or to run multiple EAs with fixed sizes and then compare the networks with each other. After all, it can be argued that some disadvantages larger networks experience in competition with smaller networks are natural, and reason for biological evolution to perfer a certain simplicity in organisms.

An additional possibbility is to adapt the genome of the network such that it codes for not only which units are active, but also which connections are active. Though in the current study it was possible to neglect connections by setting their weight (close) to 0, this almost never happened. It is not likely all connections are needed and they are likely to complicate finding suitable weight settings. Moreover, fully connected networks are biologically implausible: there are only so many connections to other neurons one neuron can handle (imagine receiving input from 10000 of your peers) and spatial constraints also play a role in bigger systems.

Furthermore, the simulations can be made more realistic by introducing noise to the C- and other input units. Sensors are not always reliable in the real world and agents should not fully rely on the exact input always being correct. This could result in more robust behavioral pathways that compensate for experienced noise in input.

Finally, as discussed previously, it would be interesting to see what the effect is of combining evolution and learning. To make this possible, a different kind of network is necessary. A learning rule needs to be selected that determines how the weights of the connections are updated after feedback. Both the initial weight setting and the change of the weights over time are of importance to the behavior of the agent. It is expected the agent will be able to perform tasks better, and that there will be a preference for network configurations that are more flexible.

The suggestions made here should provide more conclusive answers to the research question at hand. These are of course only a few of the many possible studies that can be conducted in evolving control systems in the future.

6 Conclusion

In this study the possible evolutionary advantage of minimalistic control systems was tested by simulating the evolution of neural controllors in a changing environment. A minimalistic control system is defined as a control system that supports non-reactive behavior by modifying automatic behaviors. Distinguishing characteristics are that this system is not needed for performing frequent actions and that it only modifies behavior instead of generating it. The idea behind this is that in general, the environment provides us with enough cues to complete even complex tasks automatically.

It has been found that the C-units evolved to function as simple control systems, enabling agents to show different behaviors in the same positions. This was unexpected, since the C-units are considered a form of input and it was thought that the task would require networks with additional units. The C-units appear to function in a manner similar to the hypothesized, minimalistic control system. Important is that the C-units are not 'picking' certain behaviors, such as 'approach T2 and T4 now' and 'avoid T1 and T3'. The exact same perception-action system is active in both states of the environment, but their outcome is simply influenced by the activity of the C-units. This observation supports the theory that only minimal control is required in many situations. Given that the C-units are pregiven and their activation is set externally, the question remains whether and when these kind of control units evolve by themselves. Further research is necessary to establish how the results scale up to more complex environments and agents, and whether internal control structures will have characteristics similar to the C-units and in line with minimalistic control theory.

References

Beer, R. (2009). Beyond control: the dynamics of brain-body-environment interaction in motor systems. *Adv Exp Med Biol*, 629.

Beer, R. D. and Gallagher, J. C. (1992). Evolving dynamical neural networks for adaptive behavior. *Adaptive Behavior*, 1(1):91–122.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15.

Bullinaria, J. A. (2001). Simulating the evolution of modular neural systems. In *Proceedings of the Twenty-Third Annual Conference of the Cognitive Science Society, Mahwah, NJ: Lawrence Erlbaum Associates*, pages 146–151.

Eiben, A. E. and Smith, J. E. (2008). *Introduction to Evolutionary Computing (Natural Computing Series)*. Springer.

Evans, J. S. (2008). Dual-processing accounts of reasoning, judgment, and social cognition. *The Annual Review of Psychology*, 1:255–278.

Floreano, D., Dürr, P., and Mattiussi, C. (2008). Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 1(1):47–62.

Floreano, D. and Mondada, F. (1994). Automatic creation of an autonomous agent: genetic evolution of a neural-network driven robot. In *SAB94: Proceedings of the third international conference on Simulation of adaptive behavior : from animals to animats 3*, pages 421–430, Cambridge, MA, USA. MIT Press.

Fuster, J. (1997). *The Prefrontal Cortex: Anatomy, Physiology, and Neuropsychology of the Frontal Lobe*. Lippincott Williams & Wilkins.

Garavan, H., Ross, T. J., Murphy, K., Roche, R. A., and Stein, E. A. (2002). Dissociable executive functions in the dynamic control of behavior: inhibition, error detection, and correction. *Neuroimage*, 17(4):1820–1829.

Haselager, P., van Dijk, J., and van Rooij, I. (2008). A lazy brain? embodied embedded cognition and cognitive neuroscience. In Calvo, P. and Gornila, A., editors, *Handbook of Cognitive Science an Embodied Approach*. Elsevier Inc.

Hornby, G. S., Lipson, H., and Pollack, J. B. (2001). Evolution of generative design systems for modular physical robots. In *IEEE International Conference on Robotics and Automation*, pages 4146–4151.

Husbands, P., Harvey, I., and Cliff, D. (1995). Circle in the round: State space attractors for evolved sighted robots. *Robotics and Autonomous Systems*, 15(1-2):83–106.

Koechlin, E., Ody, C., and Kouneiher, F. (2003). The architecture of cognitive control in the human prefrontal cortex. *Science*, 302:1181–1185.

Lhermitte, F. (1986). Human autonomy and the frontal lobes. part ii: Patient behavior in complex and social situations: the "environmental dependency syndrome". *Annals of neurology*, 19(4):335–343.

Lhermitte, F., Pillon, B., and Serdaru, M. (1986). Human autonomy and the frontal lobes. part i: Imitation and utilization behavior: a neuropsychological study of 75 patients. *Annals of neurology*, 19(4):326–334.

Menon, V., Adleman, N., White, C., Glover, G., and Reiss, A. (2001). Error-related brain activation during a go/nogo response inhibition task. *Human Brain Mapping*, 12(3):131–143.

Miller, E. K. and Cohen, J. D. (2001). An integrative theory of prefrontal cortex function. *Annual Review of Neuroscience*, 24:167–202.

Murphy, R. R. (2000). *Introduction to AI Robotics*. MIT Press, Cambridge, MA, USA.

Nolfi, S. and Floreano, D. (2001). *Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-organizing Machines*. MIT Press, Cambridge, MA. 2001 (2nd print), 2000 (1st print).

Ruppin, E. (2002). Evolutionary autonomous agents: A neuroscience perspective. *Nature Reviews Neuroscience*, 3(2):132–141.

Russell, S. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition.

Sakagami, M., Pan, X., and Uttl, B. (2006). Behavioral inhibition and prefrontal cortex in decision-making. *Neural Networks*, 19:1255–1465.

Schneider, W. and Chein, J. M. (2003). Controlled & automatic processing: behavior, theory, and biological mechanisms. *Cognitive Science*, 27:525–559.

Tanaka, Y., Albert, M. L., Hara, H., Miyashita, T., and Kotani, N. (2000). Forced hyperphasia and environmental dependency syndrome. *J Neurol Neurosurg Psychiatry*, 68(2):224–6.

van Dartel, M., Sprinkhuizen-Kuyper, I., Postma, E., and van den Herik, J. (2005). Reactive Agents and Perceptual Ambiguity. *Adaptive Behavior*, 13(3):227–242.

van Dijk, J., Kerkhofs, R., van Rooij, I., and Haselager, W. (2008). Can there be such as thing as embodied embedded cognitive neuroscience? *Theory & Psychology*, 13:297–316.

Yamauchi, B. M. and Beer, R. D. (1994). Sequential behavior and learning in evolved dynamical neural networks. *Adapt. Behav.*, 2(3):219–246.

Appendices

A Maximum score

In order to compare the fitness of agents from different conditions, the maximum score in each condition was determined. First the shortest routes are chosen by determining the distance between the good targets of each state (Table 5 and 6). Two additional steps were taken into account if it was necessary for the agent to turn around. The extra steps it takes the agent to get from their starting position to the target most efficiently is determined. The total maximum number of visits to good targets is determined for each starting position by counting the number of hits if the minimal number of steps is used for each distance. These scores are averaged and floored. For condition A an extra difficulty is the state change at time 200 and 300. For this condition, the scores were determined separately for the time segments $0-200$, $200-300$ and $300-500$. The ending position of the agent in one time segment is taken as starting position in the next. Then the scores for the segments were summed. An overview of the results is given in Table 7.

Targets	**Distance**
T1 - T3	62px (7 time-steps)
T3 - T5	53px (6 time-steps)
T1 - T5	88px (9 time-steps)

Table 5: Distance between good targets in state 1

Targets	**Distance**
T2 - T4	44px (5 time-steps)
T4 - T5	66px (7 time-steps)
T2 - T5	61px (7 time-steps)

Table 6: Distance between good targets in state 2

	Condition A	**Condition B**	**Condition C**
(150, 150)	360	415	355
(90, 70)	355	415	360
(20, 30)	355	410	355
Average	**356**	**413**	**356**

Table 7: Maximum scores for each condition from each starting position