A Quantitative Analysis of Memory Requirement and Generalization Performance for Robotic Tasks

DaeEun Kim
Biological Cybernetics Lab, School of Electrical and Electronics Engineering,
Yonsei University, Shinchon-dong, Seoul, 120-749, Corea (South Korea)
daeeun@yonsei.ac.kr

ABSTRACT

In autonomous agent systems, memory is an important element to handle agent behaviors appropriately. We present the analysis of memory requirements for robotic tasks including wall following and corridor following. The robotic tasks are simulated with sensor modeling and motor actions in noisy environments. In this paper, control structures are based on finite state machines for memory-based controllers, and we use the evolutionary multiobjective optimization approach with two objectives, behavior performance and memory size. For each task, a quantitative approach to estimate internal states with a different number of sensors is applied and the best controllers are evaluated in several test environments to examine their generalization characteristics and efficiency. Finite state machines with a hierarchy of memory are also compared with feedforward neural networks for the behavior performance.

Categories and Subject Descriptors: I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search; I.2 [[Artificial Intelligence]: Miscellaneous; H.4 [Information Systems Applications]: Miscellaneous

General Terms: Algorithms

Keywords: evolutionary robotics, internal states, perceptual aliasing, evolutionary multiobjective optimization, generalization behaviour

1. INTRODUCTION

In mobile robots, the reactive control mapping from perceptions to motor actions has been emphasized. The relevance of hidden states in robotics research or agent behaviours can be observed in research on reactive systems [1, 2, 9]. A hidden state is defined as any world state information not determined by the current immediate perception of a mobile agent [17]. When an agent has only partial information about the surrounding environment through its sensory inputs, and the same perceived situation requires different actions in different contexts, the agent may require internal memory to solve the hidden state problem (it is often called perceptual aliasing problem, and we also say that the agent is in a non-Markovian environment). In the situation, purely reactive control cannot succeed in solving hidden state problems [22, 20, 16]. Perceptual aliasing often appears when sensors have a limited range of view of the surrounding environment or when there are a limited number of sensors.

Many agent problems in a grid world suffer from the perceptual aliasing problem, and encoding internal memory in the control structure has been suggested and tested as an alternative solution. Wilson used a zeroth-level classifier system (ZCS) for his animat experiments [23]. The original formulation of ZCS has no memory mechanisms, because the input-output mappings from ZCS are purely reactive, but Wilson suggested how internal temporary memory registers could be added. Adding an internal memory register consisting of a few binary bits can increase the number of possible actions in the system. Following Wilson's proposal, one-bit and two-bit memory registers were added to ZCS in Woods environments by Cliff and Ross [4]. They insisted ZCS manipulate and exploit internal states appropriately and efficiently in non-Markovian environments. To see the effect of internal memory, finite state machines have been applied to the Woods problems [14]. It was shown that more internal states can handle perceptually aliased situations more effectively. Recently Kim [10, 12] showed state machines can solve several agent problems in a grid world and perceptual configurations and internal states play a significant role on the behaviour performance.

Most of researches relevant to the memory analysis have focused on the grid world environment and it is quite different from real robotic experiments, even though they showed the potential and importance of internal states for robotic behaviors. Especially, grid world problems have restricted environments and their agents have simple value sensors, while robotic environment is involved with noise on continuous sensor readings and more flexible motor actions. Yet it is shown that a simple structure with state machines can solve robotic world problems [13]. Even with primitive behaviors, memory internal to the controller is useful in robotic simulation experiments to overcome limitations of purely reactive systems. It is notable that some "reactive" robots employ internal state to deal with perceptual aliasing and their control systems are not actually purely reactive [3]. Recently some researchers studied recurrent neural networks and plastic mechanism to solve real-world robotic problems [21, 8, 5]. It has been shown that the dynamic property in the neural

GECCO'07, July 7–11, 2007, London, England, United Kingdom

networks can handle perceptually aliased problems without difficulty.

In this paper, control structures are based on finite state machines, with internal memory represented as a set of internal states, in order to quantify easily the complexity of agent-environment interaction in noisy world. With the importance of internal states, an evolutionary multiobjective optimization is used to quantify memory amount needed for robotic tasks in noisy environments (only wall following and corridor following tasks will be shown in this paper). Also state machines with binary sensors are compared with feedforward neural networks in several test environments. Here, we test the generalization characteristics of evolved controllers with respect to variation of environment. We follow Kim and Hallam's evolutionary approach [13, 14] and observe whether an evolved control structure can be applied to different environments for the same task. We will see whether memory encoding structures can be more useful in the generalization of a given behaviour rather than purely reactive systems.

2. METHODS

To determine how many memory elements are required to solve a particular agent-environment interaction problem, evolutionary computation will try to optimize agent performance for each quantity of controller memory. By doing so, the trade-off between performance and quantity of memory can be explored. If sensors are discretized, a controller with internal memory can easily be expressed as a finite state machine (FSM). If a task can be completed with a purely reactive system, the controller can be represented as a 1-state FSM[1] which is equivalent to memoryless strategy. The amount on memory needed for a given task can be determined by counting the number of states in the FSM representing the minimal effective controller.

There has been research of memory analysis in a noise-free grid world environment [14], which uses an evolutionary multiobjective optimization (EMO) with two objectives, memory size and behaviour performance. Similar to their approach, we will use the two objectives, behavior performance and memory size, in the Pareto optimization to try to maximize behavior performance and minimize the quantity of memory (number of controller states) for a given robotic task. The shape of the Pareto surface after a run indicates a desirable number of memory elements for a given performance level. As a result, one can determine a threshold amount of memory needed to achieve a task. We assume that the quantity of memory in the optimal control structure for a given task represents the complexity of the problem faced by the agent.

We test two different control architectures, the Finite State Machines (FSMs) and feedforward neural networks. The FSM we consider is a type of Mealy machine model [15], and the machine is encoded for the evolutionary algorithm as a sequence of pairs (next state number, state output) on each sensor value in canonical order of state number. That is, each sensor configuration specifies the next state transition and the wheel motor outputs. In our multi-objective optimization experiments, the genetic pool should allow variable length chromosomes for variable state machines; the size of FSMs depends on the number of states and thus different members of the pool may have different length genetic representations. FSMs have binary sensors with a threshold on the continuous sensor range. In contrast, feedforward neural networks can be one of the best control architecture in purely reactive control systems, since they can in principle represent any mapping from sensor readings to motor actions. Neural networks used infrared sensors in the front of the robot, each of which produces a continuous integer value ranging from 0 to 1023. In neural networks weight parameters are encoded into integer values from -128 to 127 with 8 bit representation. One hidden layer and four nodes in the hidden layer are used for control structures. Then the chromosome representation will be a series of weight parameters from input layer to hidden layer and from hidden layer to output layer.

In the experiments, tournament selection of size four is used for Pareto optimization. A population is initialized with random length chromosomes. The two best chromosomes are selected using a dominating rank method[6][2] over the two objectives, memory size and behavior performance. They reproduce themselves and the two worst chromosomes are replaced by new offspring produced using one point crossover followed by mutation. In the application of variable state machines, offspring with variable numbers of states should be generated to keep diversity in the genetic pool. Thus, a size modifying genetic operator is introduced to maintain variable length coding. New offspring are thus produced with a size modifying operator, crossover and mutation by turns.

When offspring are produced, the number of memory states is randomly pre-selected for each new offspring. The chromosome size for each offspring will depend on this chosen amount of memory. The size-modifying operators are then used to produce new offspring such that they have characteristics of their parents and have the pre-chosen chromosome length. After applying the size-modifying operator, crossover is applied to the two offspring. The crossover point is selected inside both chromosome strings after aligning the prefixes of two strings. During this crossover process, mutation can be used to change randomly one integer value in the strings. The size-modifying operator is applied to 75% of new offspring, in experiments with variable state machines. When it is not used, the offspring keep the size of their parents.

A realistic simulation method is required to imitate real robot behaviors and we take a Khepera robot model [19] for simulation. Instead of whisker-like sensors, the sensor range for each infrared sensor will be shaped with a cone-bearing, because the real infrared sensors cover almost cone-shaped areas. The simulation will indirectly give us a hint of how a real Khepera robot behaves, since robotic dynamics and sensor readings model real robot tasks very closely. Figure 1(a) shows the detection ranges and angles of eight infrared sensors in a real robot. The developed simulation program will follow such real robot configurations. An example of sensor ranges with cone-bearing angles in a simulation program is shown in Figure 1(b). It models 40 degrees bearing

[1]One state means every action mode has the same internal state and thus it is purely reactive.

[2]The dominating rank method defines the rank of a given vector in a Pareto distribution as the number of elements dominating the vector. Individuals of rank 0 are dominated by no other members of the population; individuals of rank n are only by individuals of rank k for $k < n$.

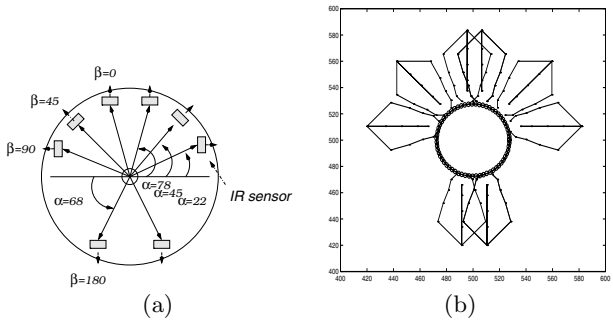

Figure 1: Khepera robot simulation model (a) robot sensor configurations (α: angular position of each sensor on the circular board, β: angular direction of each sensor) (b) cone-bearing area of each infrared sensor

angles with a set of 30 sensor points in a cone area. The sensor values are computed by estimating the detection point and its appropriate sensor value with the 30 sensor points. A uniform random noise $\pm 10\%$ is added to the sensor values. The wheel dynamics are also subjected to random noise. A uniform random noise of $\pm 10\%$ is added to the motor speeds for the two wheels, and the direction of the robot is influenced by $\pm 5\%$ random noise.

In simulation, the world environment has a binary map to help sensor processing detect any object easily. Instead of calculating each sensor detection area directly, objects and walls are masked with one in the environment array. If one of the 30 points which belong to an infrared sensor is overlapped with any masked area, its proper sensor value is recorded; each of the sensor points has the magnitude of the proximity sensor at the corresponding distance[3]. From that point, a more exact sensor value is estimated by measuring the interpolated detection point with any object.

The motor actions for both left and right wheels are restricted to the set $\{-8, -6, -4, -2, 2, 4, 6, 8\}$ and only eight possible actions are allowed for each wheel. They thus have 64 possible combinations for two wheel motors. For neural network controllers, the motor actions will have integer values ranging from -8 to 8; neural network outputs are continuous but separated into integer values, since the real Khepera robot control uses integer values instead of real values.

3. EXPERIMENTS

Some robotic behaviors such as exploration, obstacle avoidance and box pushing behaviour can be achieved by purely reactive systems with binary sensors [13]. In this paper, we provide two robotic task experiments: wall following behavior and corridor following. These tasks especially require internal memory with simple sensor configurations and so we investigate how sensor processing are related with internal states. For evolutionary computation, tournament selection of group size four is used for the test, as explained above. The two best chromosomes in a group will be copied to a new population and the others will be reproduced with genetic operators. Mutation rate is set to 2 over chromosome length, crossover rate is 0.6, and population size 100 is applied. To see the memory effect, FSMs are applied to control structures, and 25 runs are tested to validate the performance. In the experiments, three kinds of infrared sensor configurations are used: two sensors $(45°, 135°)$, four sensors $(45°, 78°, 102°, 135°)$ and six sensors (22°, 45°, 78°, 102°, 135°, 158°). We apply a threshold of 500 to the continuous sensor values to build binary sensors.

[3]This point scanning method was used with 15 sensor vectors in Khepera simulation package 2.0 [18]

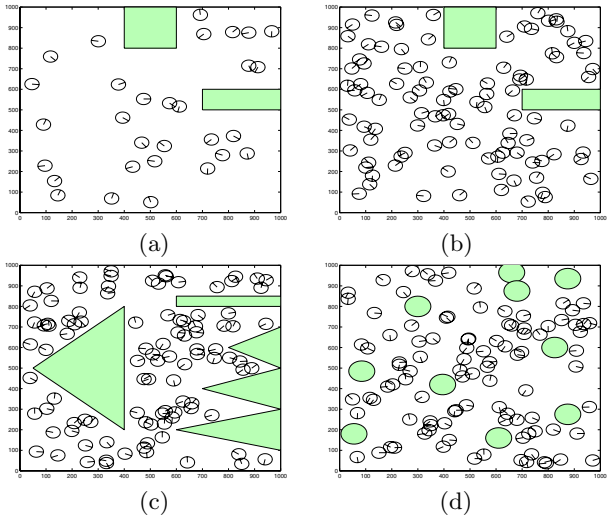

Figure 2: Test environments for wall following (a) `env0` (this is used as an evolved environment) (b) `env1` (c) `env2` (d) `env3`

Each evolutionary run proceeds as follows: Initially N starting positions for the robot are randomly selected in the arena to represent the whole environmental situations. Then every generation evolutionary computation chooses dynamically K samples among N positions depending on the fitness of each position; the higher penalty fitness, the more chance. This dynamic sample selection method reduces the computation time and also has the effect of running over all initial positions [7]. The best chromosomes in the population are evaluated multiple times to support robustness in noisy environments and they are used for the elitism strategy.

3.1 Wall Following Behavior

Wall following behavior requires at least two internal states for whisker-like binary sensor robots without noise [13]. In this paper, we study how noisy environments can influence the memory requirement. Wall following fitness is estimated with the number of energy tanks that a robot agent has visited. The energy tanks are placed on empty spaces (10 cm by 10 cm) around walls and obstacles. If a robot agent can visit all energy tanks within a limited amount of time, we can say that the agent is successful for the wall following behavior. Random noise will increase the possibility of colliding with walls. The collision will have a high penalty and thus the fitness is defined as a penalty function as follows:

$$f_W = P - V + C(T - t_c) \quad (1)$$

where P is the base of the penalty ($P = 100$ is set), V is the number of energy tanks that a robot has visited, and T is the maximum time steps for exploration, which is set to 1200 time steps in the experiments. If collision occurs,

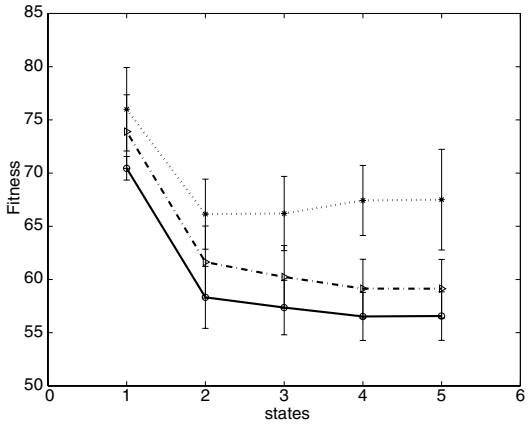

Figure 3: Memory analysis with FSMs for wall following behavior with 4 sensors (dotted: 5 generations, dashdot: 100 generations, solid: 300 generations)

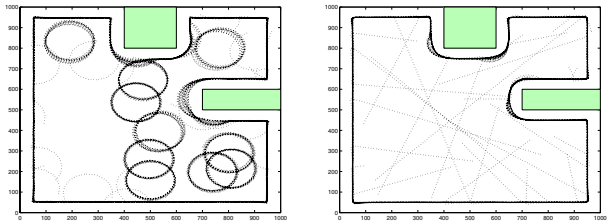

Figure 4: Comparison of the best performance between memoryless and memory-based systems for wall following behavior (with FSMs) (a) purely reactive system (b) 2 state machines

the exploration stops and the fitness is calculated with the collision time t_c. C is the binary collision flag to say if a robot has collided with any obstacle. The environment is shown in Figure 2(a) with 30 random starting positions. N-K sampling with $N = 30, K = 5$ is used. The arena size is 100 cm by 100 cm. The arena has 46 energy tanks around the wall. The optimal fitness will be 54 (=100-46) as the least lower bound. A single run takes 1200 time steps.

3.1.1 Memory Analysis with the EMO Approach

The evolutionary multiobjective optimization (EMO) approach is applied to variable state machines as control architecture in order to analyze the best performance over quantified memory states. Figure 3 shows the memory analysis for wall following behavior with 4 sensors. For significance statistics, 25 runs are tested and their error bars with 95% confidence intervals (t-distribution) are displayed together at the same generations. When the number of generations is increased, the performance becomes close to optimal, visiting all energy tanks (fitness 54). There is a significant difference between memoryless (one state) approach and memory-based (more than one state) approach; when we take independent runs for each number of states using a fixed length of chromosomes instead of the EMO approach, it was hard to see the significant difference. However, two states are not significantly different from more than two states, though more than two states are a little better in the average performance. From the figure, one can just say that more than two states are easily leading to a little faster convergence.

Even if we take only two infrared sensors at 45° and 135° in front, similar results are obtained from the EMO analysis. Figure 4 shows wall following behaviors by the best chromosomes for purely reactive controllers and two state controllers. In Figure 4(a), the memoryless reactive system has circular movements to detect any wall and begins to follow walls when any sensation is recognized. On 10 starting positions among 30 positions, the robot shows circular movements and the performance becomes degraded since the robot cannot scan other areas. Once any wall is detected, the controller shows good performance to follow walls and visit all energy tanks. Wall following behavior with two state machines is demonstrated in Figure 4(b). The behavior is decomposed into two simple behaviors: one is to go straight to reach any wall when the robot sees no obstacle and the other is to follow walls by repeating two actions of turning right and moving forward depending on sensory readings. There is a conflict of perceptions in the two primitive behaviors. The sensation of no obstacle in front should be distinguished between the two behaviors and internal states can solve the conflict by memorizing each distinct situation, but purely reactive controllers have a limitation on the problem because they cannot separate the perceptual aliases.

From the above results, it is inferred that the wall following behavior requires two states, or 1-bit memory even in noisy environments regardless of the number of sensors. The EMO approach showed it was very effective for taking the significance test on fitness distribution as well as for obtaining solutions for each number of states.

For neural network controllers, two, four and six sensors were tested, where each case has no internal state. Their performance is not significantly different, which indirectly implies that two sensors are sufficient to evolve wall following behavior for a given environment. It is because neural networks have continuous ranges on sensors and motor actions.

3.1.2 Testing Controllers

Robot controllers, FSM controllers and neural network controllers, were evolved over 30 random positions and directions in one environment. To prove the usefulness and generalization characteristics of evolved controllers, the controllers were tested in several environments. The purpose of testing controllers is to see what kind of controllers demonstrate better generalization property when applied to various environments. The testing will check whether memory structure and sensor configurations are important to adapt controllers to environments. Also, it will show whether controllers adapted to one environment can be easily applied to other different environments.

Various control structures have been investigated in the experiments as shown in Table 1. The testing environments consist of four environments, `env0, env1, env2, env3` shown in Figure 2. The first environment (`env0`) is the same as the evolved environment and has 30 starting positions. The environment `env1` has the same arena but 100 random positions. The chromosomes with the best performance are first collected by evolving robots in the environment `env0`. Each evolved controller is evaluated 10 times for each of the four

control structure		env0	env1	env2	env3
2 sensors, 1 state	c_0	0.1 ± 0.3	0.7 ± 0.7	3.3 ± 0.6	2.6 ± 1.6
	c_1	0.0 ± 0.0	0.6 ± 0.7	2.9 ± 0.9	0.1 ± 0.3
2 sensors, 2 states	c_2	0.0 ± 0.0	1.2 ± 0.8	4.6 ± 0.7	0.5 ± 0.7
	c_3	0.0 ± 0.0	2.0 ± 0.0	3.8 ± 0.7	$\mathbf{0.0 \pm 0.0}$
2 sensors, 3 states	c_4	0.0 ± 0.0	0.0 ± 0.0	4.4 ± 0.5	0.0 ± 0.0
	c_5	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	4.1 ± 0.9	$\mathbf{0.0 \pm 0.0}$
2 sensors, 4 states	c_6	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	4.7 ± 0.9	0.1 ± 0.3
	c_7	0.0 ± 0.0	0.5 ± 0.5	4.0 ± 0.0	0.6 ± 0.7
4 sensors, 1 state	c_8	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	$\mathbf{0.0 \pm 0.0}$	0.3 ± 0.5
	c_9	0.0 ± 0.0	0.5 ± 0.5	0.0 ± 0.0	1.7 ± 0.9
4 sensors, 2 states	c_{10}	0.0 ± 0.0	0.6 ± 0.5	1.4 ± 0.5	0.6 ± 0.8
	c_{11}	0.0 ± 0.0	1.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	1.0 ± 0.0
4 sensors, 3 states	c_{12}	0.0 ± 0.0	0.6 ± 0.8	0.8 ± 0.9	0.3 ± 0.5
	c_{13}	0.0 ± 0.0	4.0 ± 0.0	2.1 ± 0.9	1.6 ± 0.8
4 sensors, 4 states	c_{14}	0.0 ± 0.0	0.5 ± 0.5	0.7 ± 0.9	0.7 ± 0.6
	c_{15}	0.0 ± 0.0	0.5 ± 0.7	3.0 ± 1.2	0.5 ± 0.7
2 sensors, neural	c_{16}	0.0 ± 0.0	1.9 ± 0.3	34.2 ± 2.8	7.0 ± 0.0
	c_{17}	0.0 ± 0.0	1.3 ± 0.6	54.8 ± 3.8	0.1 ± 0.3
4 sensors, neural	c_{18}	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	2.4 ± 1.5	0.8 ± 0.4
	c_{19}	0.0 ± 0.0	1.0 ± 0.0	6.6 ± 1.0	4.9 ± 0.3
6 sensors, neural	c_{20}	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	2.4 ± 0.7	0.1 ± 0.3
	c_{21}	0.0 ± 0.0	0.0 ± 0.0	1.0 ± 0.0	0.1 ± 0.3

(a) collision test

control structure		env0	env1	env2	env3
2 sensors, 1 state	c_0	68.18 ± 0.72	69.92 ± 0.33	40.00 ± 0.19	82.98 ± 0.53
	c_1	68.21 ± 0.62	69.83 ± 0.29	$\mathbf{37.07 \pm 0.13}$	82.03 ± 0.30
2 sensors, 2 states	c_2	54.00 ± 0.00	54.00 ± 0.00	46.76 ± 0.19	77.98 ± 0.11
	c_3	54.00 ± 0.00	$\mathbf{54.00 \pm 0.00}$	38.96 ± 0.06	76.84 ± 0.24
2 sensors, 3 states	c_4	54.00 ± 0.00	54.00 ± 0.00	44.58 ± 0.10	78.91 ± 0.04
	c_5	54.00 ± 0.00	54.00 ± 0.00	39.27 ± 0.12	76.89 ± 0.33
2 sensors, 4 states	c_6	54.00 ± 0.00	54.00 ± 0.00	39.82 ± 0.22	76.80 ± 0.19
	c_7	54.00 ± 0.00	54.00 ± 0.00	41.83 ± 0.08	78.04 ± 0.07
4 sensors, 1 state	c_8	67.71 ± 0.94	69.67 ± 0.37	38.43 ± 0.14	81.61 ± 0.41
	c_9	68.27 ± 0.91	69.90 ± 0.32	45.74 ± 0.40	83.67 ± 0.36
4 sensors, 2 states	c_{10}	54.36 ± 0.03	54.37 ± 0.02	49.27 ± 0.04	78.92 ± 0.29
	c_{11}	54.00 ± 0.00	54.00 ± 0.00	38.29 ± 0.30	$\mathbf{76.10 \pm 0.31}$
4 sensors, 3 states	c_{12}	54.00 ± 0.00	59.19 ± 0.59	49.46 ± 0.10	81.67 ± 0.28
	c_{13}	54.00 ± 0.09	54.00 ± 0.00	$\mathbf{37.45 \pm 0.24}$	77.88 ± 0.42
4 sensors, 4 states	c_{14}	54.00 ± 0.01	59.15 ± 0.27	49.62 ± 0.12	81.81 ± 0.21
	c_{15}	54.01 ± 0.02	54.16 ± 0.20	48.01 ± 0.38	79.11 ± 0.11
2 sensors, neural	c_{16}	56.46 ± 0.59	60.25 ± 0.64	50.84 ± 0.32	77.92 ± 0.44
	c_{17}	55.66 ± 1.15	59.70 ± 0.67	49.62 ± 0.65	77.78 ± 0.21
4 sensors, neural	c_{18}	55.57 ± 0.94	59.54 ± 0.67	49.57 ± 0.43	77.05 ± 0.55
	c_{19}	55.71 ± 1.13	59.20 ± 0.67	56.81 ± 0.94	77.84 ± 0.31
6 sensors, neural	c_{20}	54.00 ± 0.00	57.66 ± 0.24	41.18 ± 0.09	78.91 ± 0.09
	c_{21}	54.61 ± 0.60	58.86 ± 0.63	46.89 ± 0.12	77.19 ± 0.60

(b) fitness performance

Table 1: Comparison of performance with various control structures for wall following (each data shows the average fitness and standard deviation over 10 evaluations, bold: best performance)

environments and its performance is recorded. The two best controller data with desirable performance in four environments are collected into Table 1. It is assumed that the best solution controllers evolved with each control structure will represent the control structure. The two best controllers for each control structure are collected and they are compared with the two best controllers for other control structures.

From Table 1, the best control architecture is the controller c_{11} with 4 sensors and 2 internal states. It has almost no collision and uniformly good fitness performance in every test environment. The behaviours demonstrate that binary sensor information without the continuous sensor range is sufficient to process perceptions appropriately.

Table 1 shows the results of two components of performance measurement. One is to check how many collisions occur for each environment by testing controllers at all starting positions. The other is to see the fitness which is related with how many energy tanks are visited. The fitness is averaged over starting positions that experience no collision. The optimal fitness (penalty fitness) is 54 for `env0` and `env1`, 37 for `env2`, and 76 for `env3`. The two component measures, collision and fitness, are estimated with 10 runs to see the distribution. Desirable controllers should have the least collisions and the best fitness. In the experiments, there was no perfect controller; no collision and the optimal fitness for every test environment. For the generalization performance of the controllers, we can consider a Pareto distribution of the two objectives. Table 1 shows that two internal states or more with 4 sensors are better than purely reactive systems in that respect. Also two sensors with multiple internal states outperform purely reactive systems in the fitness performance.

Noisy sensors and motor actions often influence the perceptual states on robots with a few sensors, which is a significant factor for the generalization performance [11]. Increasing the number of sensors gives a potential to obtain

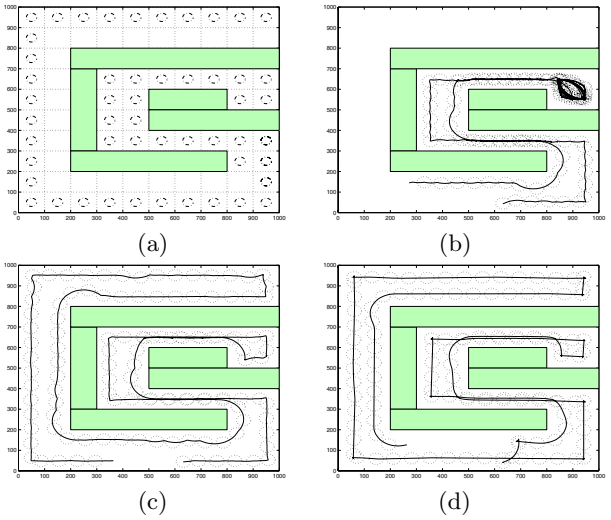

Figure 5: An example of corridor following behavior (a) energy tank distribution (b) purely reactive controller with two sensors (c) two state machine controller with two sensors (d) neural network with two sensors

better generalization performance in different environments. Adding sensors can help robots recognize perceptual situations better even with noisy sensor readings. It will prevent misunderstanding environmental states due to noise and can solve perceptual aliases partly. The collision rate for neural network controllers can be reduced by increasing the number of sensors from two to six as shown in Table 1(a). This also happens for state machine controllers; four sensors have smaller collision rates than two sensors (see test environment `env2`).

Internal states play an important role in achieving desirable generalization performance. Two state machines with two sensors or four sensors are better in fitness performance than even neural networks (see Table 1(b)). Purely reactive systems with continuous sensor range and expanded motor actions still have restrictions in wall following behavior, even though they improve the performance. Thus, one can say that wall following behavior requires at least two internal states in noisy environments.

3.2 Corridor Following

Corridor following is similar to wall following behavior. The task is to follow narrow corridors from one end to the other end. Figure 7(a) shows the environment and 30 random positions selected in the arena. The environment has narrow corridors in the middle of the path and robots need to pass through corridors appropriately to enter a small room at the end of corridor. Narrow corridors are not open fields, so robots tend to hesitate to go through them since there is a higher probability of collision. For exploration time, 1200 time steps are assigned for each genome controller. The energy tanks are distributed around walls as shown in Figure 5(a), which will lead robots to follow corridors.

Evolutionary computation for corridor following uses the same fitness function given in equation (1). The experiments first investigate memory analysis for how many states are required for corridor following. The basic control structure is a

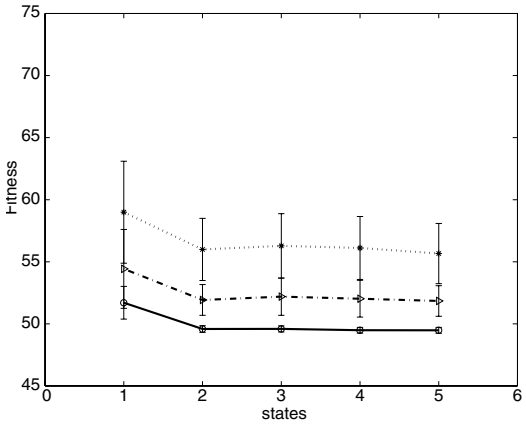

Figure 6: Memory analysis with FSMs for corridor following behavior with 4 sensors (dotted: 60 generations, dashdot: 100 generations, solid: 300 generations)

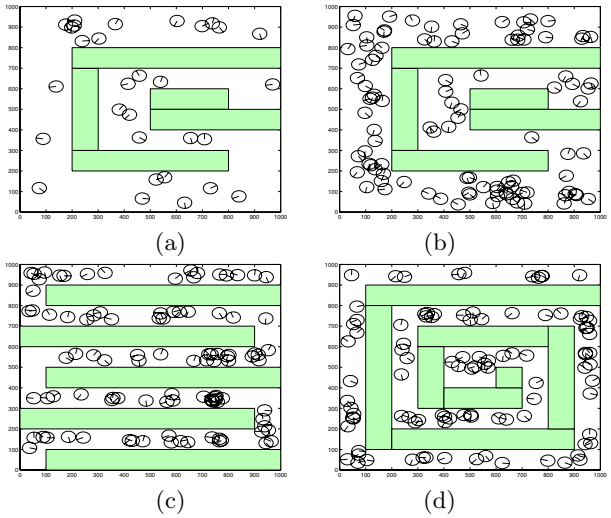

Figure 7: Test environments for corridor following (a) `env0` (b) `env1` (c) `env2` (d) `env3`

finite state machine tested in wall following behavior. Memory and its corresponding performance will be analyzed with the number of sensors. Neural networks are also compared with single-threshold state machines. The sensor range and the number of sensors are relevant factors related to perceptual aliases.

3.2.1 Memory Analysis with the EMO approach

In this environment, robots always move close to corridors and so they will not need the effort of exploration to reach a corridor; for wall following, robots need to go straight or make a circular movement to approach a wall. We still have the question of whether or not corridor following requires internal memory. The EMO approach was quite effective to investigate the memory hierarchy and fitness performance, instead of running each state machine controller independently.

The maze-style arena has a room with a small exit at the end of the corridor in the middle section. Purely reactive

control structure		env0	env1	env2	env3
2 sensors, 1 state	c_0	0.0 ± 0.0	1.2 ± 0.4	2.0 ± 0.4	5.1 ± 1.3
	c_1	0.0 ± 0.0	1.2 ± 0.4	2.0 ± 0.0	1.8 ± 1.0
2 sensors, 2 states	c_2	0.0 ± 0.0	1.8 ± 0.4	4.5 ± 0.5	3.0 ± 0.9
	c_3	0.0 ± 0.0	1.9 ± 0.5	4.6 ± 0.7	2.0 ± 1.0
2 sensors, 3 states	c_4	0.0 ± 0.0	2.2 ± 0.4	2.1 ± 0.3	3.7 ± 1.0
	c_5	0.0 ± 0.0	4.1 ± 0.9	2.4 ± 0.5	4.7 ± 1.1
4 sensors, 1 state	c_6	0.0 ± 0.0	0.0 ± 0.0	1.0 ± 0.0	0.6 ± 0.5
	c_7	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	1.0 ± 0.0	0.2 ± 0.4
4 sensors, 2 states	c_8	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	0.3 ± 0.5	$\mathbf{0.0 \pm 0.0}$
	c_9	0.0 ± 0.0	0.1 ± 0.3	1.1 ± 0.8	1.0 ± 0.0
4 sensors, 3 states	c_{10}	0.0 ± 0.0	0.2 ± 0.4	0.5 ± 0.5	0.2 ± 0.4
	c_{11}	0.0 ± 0.0	0.6 ± 0.8	1.0 ± 0.0	0.3 ± 0.5
2 sensors, neural	c_{12}	0.0 ± 0.0	1.2 ± 0.4	0.7 ± 0.6	0.5 ± 0.5
	c_{13}	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	1.5 ± 0.5	$\mathbf{0.0 \pm 0.0}$
4 sensors, neural	c_{14}	0.0 ± 0.0	0.0 ± 0.0	0.7 ± 0.5	1.0 ± 0.0
	c_{15}	0.0 ± 0.0	$\mathbf{0.0 \pm 0.0}$	0.0 ± 0.0	1.0 ± 0.0

(a) collision test

control structure		env0	env1	env2	env3
2 sensors, 1 state	c_0	59.66 ± 1.22	59.46 ± 0.34	54.68 ± 0.08	55.84 ± 0.54
	c_1	59.05 ± 1.02	59.15 ± 0.37	61.15 ± 0.21	61.37 ± 0.35
2 sensors, 2 states	c_2	50.03 ± 0.09	50.49 ± 0.07	49.03 ± 0.10	47.24 ± 0.09
	c_3	50.11 ± 0.12	50.65 ± 0.22	49.02 ± 0.06	47.33 ± 0.21
2 sensors, 3 states	c_4	50.92 ± 0.27	51.57 ± 0.07	49.78 ± 0.07	47.54 ± 0.12
	c_5	50.16 ± 0.15	51.81 ± 0.07	49.98 ± 0.17	47.49 ± 0.15
4 sensors, 1 state	c_6	50.19 ± 0.15	50.73 ± 0.09	48.96 ± 0.05	46.97 ± 0.16
	c_7	50.19 ± 0.13	50.79 ± 0.09	49.33 ± 0.15	46.66 ± 0.22
4 sensors, 2 states	c_8	49.14 ± 0.16	49.19 ± 0.09	48.02 ± 0.16	46.03 ± 0.14
	c_9	49.09 ± 0.08	49.09 ± 0.10	47.79 ± 0.15	44.98 ± 0.06
4 sensors, 3 states	c_{10}	49.10 ± 0.09	49.51 ± 0.06	48.80 ± 0.19	46.67 ± 0.29
	c_{11}	49.02 ± 0.07	49.11 ± 0.05	47.62 ± 0.11	45.30 ± 0.22
2 sensors, neural	c_{12}	49.54 ± 0.05	49.55 ± 0.06	47.96 ± 0.07	45.37 ± 0.18
	c_{13}	49.76 ± 0.08	49.82 ± 0.06	48.36 ± 0.16	45.85 ± 0.15
4 sensors, neural	c_{14}	49.00 ± 0.00	49.47 ± 0.00	$\mathbf{46.61 \pm 0.07}$	$\mathbf{43.70 \pm 0.04}$
	c_{15}	49.00 ± 0.00	$\mathbf{49.00 \pm 0.01}$	47.50 ± 0.04	44.44 ± 0.39

(b) fitness performance

Table 2: Comparison of performance with various control structures for corridor following (each data shows the average fitness and standard deviation over 10 evaluations, bold: best performance)

control systems have difficulty in finding the exit directly and repeat moving around inside the room as shown in Figure 5(b). This behavior tends to be sensitive to noise. In some cases they easily find the exit by accident and in others they stagger around the room for a long time. They normally get a low score for visiting energy tanks within a limited time amount, because they spend much time in the room and cannot cover other areas. Memory-based systems with at least two states follow walls closely and so efficiently find the exit. Figure 5(c) shows their different behaviors. In two state controllers, for instance, state information is used to sequentialize two kinds of motor actions: move forward quickly, and move forward in the right direction when there is no sensation. This helps speedy movement in a narrow corridor. Internal memory plays different roles in wall following and corridor following behaviours. The memory analysis in Figure 6 shows the difference in performance efficiency rather than the outcome of whether robots can succeed in following corridors without collision; for wall following behavior, internal memory is used to achieve the task in open areas.

Figure 5(d) shows the corridor following behaviour with neural network control with two sensors. When the neural network recognizes the sensation that the robot is very close to a wall, it produces motor actions for moving backward in the left direction until the right sensor is free of sensation. Then it continues to follow corridors. This kind of behavior has often been observed with two binary sensors and memory states. It seems that neural networks can process a continuous range of sensor readings and effectively use a quantitative information over sensors.

3.2.2 *Testing Controllers*

For the generalization performance of corridor following behaviour, various control structures were tested in several environments. The best controllers were obtained via the evolutionary process with N-K sample method and then they were evaluated over four environments given in Figure 7. Each controller chromosome was evaluated 10 times. Two sensors and four sensors were tested with states ranging from one to three, respectively. Also, feedforward neural networks were evaluated with two and four sensors. The best evolved behaviours with minimal collisions and the best fitness are shown in Table 2.

The arena in Figure 7(a) was used for evolving controllers. The environment, called `env0`, has 30 random starting positions. Various controller structures were evolved to find the best fitness in the environment. The same arena, but with 100 new random positions, which is named `env1`, will be tested to see how correctly evolved controllers work in the given environment. Another two environments in addition to `env0` and `env1` are used for test environment with 100 random positions. For the environments `env0` and `env1`, 51 energy tanks are distributed and for the environments `env2`

and `env3`, 55 and 60 energy tanks are placed at blank spaces (one space is assumed as a 10 cm by 10 cm square space). The optimal fitness for `env0, env1, env2` and `env3` is 49, 49, 45 and 40, respectively.

We can consider a Pareto distribution of the two objectives, fitness performance and collision rate in Table 2. It will help find the best control architectures for the corridor following. For state machines, multiple states with four sensors have better generalization performance than purely reactive systems or FSMs with two sensors. Specifically, adding two more sensors in front remarkably reduced the number of collisions for every state machine, while it did not improve the fitness performance to a large degree. More sensors means more information about the surrounding environment and thus the number of sensors can significantly influence the design of more adaptable controllers for the corridor following behaviour (we note that memoryless systems with four sensors are better in test environments than memory-based systems with two sensors in terms of collision and fitness performance). In addition, multiple states much improved the fitness performance of purely reactive systems for every environment. Internal states link two different motor actions to speed up robot movements and thus increase the fitness performance.

Neural networks with only two sensors have good fitness performance and rare collisions. They have a finer scale of sensor readings than binary sensor modeling and it is presumed that they easily adapt themselves to find local features for sensors. Even in neural networks, four sensors improved the performance of collision and fitness.

From Table 2, the best control structure for corridor following is the neural network controller with four sensors (c_{15}) or two state machines with four sensors (c_8). Although the state machines with binary sensors cannot be directly compared with neural network controllers with continuous sensor ranges, it is notable that controllers with a few states and binary sensors do almost as well as feedforward neural network controllers.

4. CONCLUSION

In this paper we presented the effect of internal memory on robotic behaviours in noisy environment and how to quantify internal states required for desirable robot behaviors. The evolutionary multiobjective optimization (EMO) approach was applied for memory analysis. The EMO approach can produce significance statistics on the performance difference of variable state machines in noisy environments. According to the memory analysis, wall following behavior and corridor following behavior require internal states to improve performance. Memory can be served as connectives of decomposable tasks or a series of actions, or plays a role of remembering the past perceptions.

In the experiments, evolved controllers were tested in new environments to see how the behaviour can be generalized with variation of environments. Two criteria, the collision factor and fitness factor, have been considered to choose the best controllers among the various control architectures. From the experiments, finite state machines with varying dynamic elements were sufficient to achieve desirable robotic behaviors. With neural network controllers, it is inferred that the performance of purely reactive systems can be influenced by their sensor range and variety of motor actions. It is notable that controllers with a few states and binary sensors do almost as well as or better than feedforward neural network controllers with continuous sensor ranges. More complex type of state control structures with variable thresholds [11] or multi-thresholds can be studied over a variety of robotic tasks for the future work.

5. REFERENCES

[1] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Autonomation*, Vol. RA-2, No.1:14–23, 1986.

[2] R. Brooks. Intelligence without representation. In *Workshop on th Foundations of Artificial Intelligence*, Endicott House, Dedham Mass., 1987.

[3] R. Brooks. Intelligence without reason. AI memo 1293, Artificial Intelligence Laboratory, MIT, 1991.

[4] D. Cliff and S. Ross. Adding temporary memory to ZCS. *Adaptive Behavior*, 3(2):101–150, 1995.

[5] D. Floreano and J. Urzelai. Evolution of plastic control networks. *Autonomous Robots*, 11(3):311–317, 2001.

[6] C. M. Foncea and P. J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 416–423, San Mateo, California, 1993. Morgan Kaufmann.

[7] C. Gathercole. *An Investigation of Supervised Learning in Genetic Programming*. Ph. D. dissertation, University of Edinburgh, 1998.

[8] I. Harvey, E. di Paolo, R. Wood, M. Quinn, and E. Tuci. Evolutionary robotics: A new scientific tool for studying cognition. *Artificial Life*, 11:79–98, 2005.

[9] L. P. Kaelbling. An architecture for intelligent reactive systems. In *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop*, Endicott House, Dedham Mass., 1986.

[10] D. Kim. Analyzing sensor states and internal states in the Tartarus problem with tree state machines. In *Parellel Problem Solving From Nature 8, Lecture Notes on Computer Science vol. 3242*, pages 551–560, 2004.

[11] D. Kim. Evolving internal memory for T-maze tasks in noisy environments. *Connection Science*, 16(3):183–210, 2004.

[12] D. Kim. Memory analysis and significance test for agent behaviours. In *Proceedings of the 8th conf. on Genetic and Evolutionary computation*, pages 151–158. ACM, 2006.

[13] D. Kim and J. Hallam. Mobile robot control based on Boolean logic with internal memory. In *Advances in Artificial Life, Lecture Notes in Computer Science vol. 2159*, pages 529–538, 2001.

[14] D. Kim and J. Hallam. An evolutionary approach to quantify internal states needed for the woods problem. In *From Animals to Animats 7, Proc. of Int. Conf. on the Simulation of Adaptive Behavior*, pages 312–322. MIT Press, 2002.

[15] Z. Kohavi. *Switching and Finite Automata Theory*. McGraw-Hill, New York, London, 1970.

[16] A. McCallum. Overcoming incomplete perception with utile ditinction memory. In *Proceedings of the 10th International Machine Learning Conference*. Morgan Kauffman, 1993.

[17] A. McCallum. *Reinforcemnet Learning with Selective Perception and Hidden State*. Ph. D. dissertation, University of Rochester, 1996.

[18] O. Michel. An artificial life approach for the synthesis of autonomous agents. In *Artificial Evolution*. Springer, 1996.

[19] F. Mondada, E. Franzi, and P. Ienne. Mobile robot miniaturisation: A tool for invesitigation in control algorithms. In *Proceedings of the Third International Symposium on Experimental Robotics*. Kyoto, Japan, 1993.

[20] A. P. Singh, T. Jaakkola, and M. I. Jordan. Model-free reinforcement learning for non-Markovian decision problems. In *Proceedings of the 11th International Machine Learning Conference*. Morgan Kauffman, 1994.

[21] E. Tuci, V. Trianni, and M. Dorigo. Feeling of time through sensory-motor coordination. *Connection Science*, 16(4):301–324, 2004.

[22] S. D. Whitehead. *Reinforcemnet Learning for the Adaptive Control of Perception and Action*. Ph. D. dissertation, University of Rochester, 1992.

[23] S. W. Wilson. ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1):1–18, 1994.