An Evolutionary Approach to Quantify Internal States Needed for the Woods Problem

DaeEun Kim*  John C.T. Hallam
IPAB, Division of Informatics
University of Edinburgh
5 Forrest Hill
Edinburgh, EH1 2QL
Scotland, United Kingdom
{daeum, john}@dai.ed.ac.uk

Abstract
The Woods Problem is a difficult problem for purely reactive systems to handle. The difficulties are related to the perceptual aliasing problem, and the use of internal memory has been suggested to solve the problem. In this paper a novel approach in evolutionary computation is introduced to quantify the amount of memory required for a given task. The approach has been applied to Woods Problems such as wood101, wood102, Sutton’s gridworld and woodsl4.

Finite state machine controllers are used, as these permit easy measurement of the amount of memory in the controller. A concurrent evolutionary search for the minimal but optimal control structure in memory-based systems, using an evolutionary Pareto-optimal search mechanism, determines the best behavior fitness for each level of controller memory. This memory analysis demonstrates the effect of internal memory in evolved controllers for Woods Problems and is also used to investigate the relationship between the number of sensors available to an agent and the amount of memory necessary for effective behavior.

1 Introduction
Woods problems are goal-search problems for an agent to try to find a goal position, starting at random initial positions. The agent has eight sensors and it needs to find the shortest path to the goal. A variety of such problems have been defined in the literature (see below) and they are often cited as difficult problems for a memoryless strategy to solve.

McCallum defined a hidden state as any world state information not determined by the current immediate perception of a mobile agent (McCallum, 1996). The relevance of hidden states in robotics research can be observed in research on reactive systems (Brooks, 1986, Brooks, 1987, Kaelbling, 1986). Purely reactive agents choose their current motor action using only their current perception, and a significant amount of robotics research has concentrated on them (Maes and Brooks, 1990, Lee et al., 1997, Sutton, 1991, Chapman and Kaelbling, 1991, Lee, 1998, Mahadevan and Connell, 1991). However, in some non-Markovian environments, purely reactive control cannot succeed in solving hidden state problems (Whitehead, 1992, Singh et al., 1994, McCallum, 1993). Hidden states often appear when sensors have a limited range of view of the surrounding environment or when there are a limited number of sensors. Such hidden state problems are often called perceptual aliasing problems (Whitehead and Ballard, 1991).

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 suffers from a perceptual aliasing problem. A solution to this problem is to find actions leading to a situation where the agent has an unambiguous sensory pattern (Nolfi and Floreano, 2000). However, this strategy is not a fundamental solution to perceptual aliasing problem and it is effective only when the agent can find at least one unambiguous sensory pattern. Some “reactive” robots employ internal state to deal with perceptual aliasing and their control systems are not actually purely reactive (Brooks, 1991). Even with primitive behaviors, memory internal to the controller is useful in some robotic experiments to overcome limitations of purely reactive systems (Kim and Hallam, 2001).

McCallum (McCallum, 1996) argued that internal memory should be added to solve perceptual aliasing

*Now with Max Planck Institute for Psychological Research, Amalienstr. 33, Munich, 80799, Germany

1The environmental features are not immediately observable, but they have dependencies with past states.
problems in non-Markovian environments. He developed a reinforcement learning algorithm incorporating memory to prevent hidden states. When an agent suffers from hidden state problems, the perceptions cannot directly define the next motor action. Its decision will depend on the current perception and its internal memory about past perceptions and actions. Memory thus plays a role to disambiguate aliased perceptions.

Wilson used a zeroth-level classifier system (ZCS) for his animat experiments (Wilson, 1994). 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 (Cliff and Ross, 1995). They insisted ZCS manipulate and exploit internal states appropriately and efficiently in non-Markovian environments. Bakker (Bakker and de Jong, 2000) proposed a means of counting the number of states required to perform a particular task in an environment by extracting state counts from finite state machine controllers to measure the complexity of agents and environments. They applied their methodology to the Woods7 environment (Wilson, 1994) to estimate internal states. Colombetti and Dorigo developed ALECSYS, a classifier system to learn proper sequences of subtasks by maintaining internal state and transition signals which prompt an agent to switch from one subtask to another (Colombetti and Dorigo, 1994). Similar to the ZCS experiments (Wilson, 1994), Lanzi has shown that internal memory is effective with adaptive agents and reinforcement learning, when perceptions are aliased (Lanzi, 2000). Also there has been research using a finite-size window of current and past observations and actions (McCallum, 1996, Lin and Mitchell, 1992).

Woods Problems have often been mentioned in association with the perceptual aliasing problem. The environments include many ambiguous situations that require different actions. There have been many variations of the Woods environment: Sutton's gridworld (Sutton, 1990), McCallum's maze (McCallum, 1996), maze10 (Lanzi, 1998), woods102 and woods14 (Cliff and Ross, 1995). They range from simple to complex environments. They have been tackled with classifier systems or reinforcement algorithms (Lanzi, 1998, Lanzi, 2000). The reinforcement systems are mainly based on purely reactive systems (Kaelbling, 1986). Memory-encoding methods with reinforcement learning have been successful on complex Woods Problems (Cliff and Ross, 1995, Lanzi, 2000).

The incorporation of state information permits an evolved control structure to behave better, using past information, than a pure reaction to the current sensor inputs. Finite state machines have been used previously in evolutionary computation to represent state information (Fogel et al., 1966, Stanley et al., 1995, Miller, 1996). In this paper, control structures are based on finite state machines, with internal memory represented as a set of internal states. With the importance of internal states, a novel approach is introduced to quantify memory amount needed to solve Woods Problems. We discuss the potential of evolving memory-based control structures and how they work to improve the performance. Pareto optimization will be tools to recognize the relevance and importance of memory for given tasks.

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 which is equivalent to memoryless strategy. Conversely, the amount on memory needed for a non-reactive task can be determined by counting the number of states in the FSM representing the minimal effective controller.

We have developed an evolutionary algorithm with Pareto optimization for variable sized finite state machines. Two objectives, behavior performance and memory size, are used in the Pareto optimization to try to maximize behavior performance and minimize the quantity of memory (number of controller states). 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 often 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.

The finite state machine we consider is a type of Mealy machine model, defined as $M = (Q, \Sigma, \Delta, \delta, \lambda, q_0)$ where $q_0$ is an initial state, $Q$ is a finite set of states, $\Sigma$ is a (sensor) input space $\{0, 1\}^*$, $\Delta$ is the multi-valued output, $\delta$ is a state transition function from $Q \times \Sigma$ to $Q$, and $\lambda$ is an output mapping from $Q \times \Sigma$ to $\Delta$, i.e. $\lambda(q, a) \in \Delta$. $\delta(q, a)$ is defined as the next state for each state $q$ and input value $a$, and the output action of machine $M$ for the input sequence $a_1, a_2, a_3, \ldots, a_n$ is $\lambda(q_0, a_1), \lambda(q_1, a_2), \lambda(q_2, a_3), \ldots, \lambda(q_{n-1}, a_n)$, where $q_0, q_1, q_2, \ldots, q_n$ is the sequence of states such that $\delta(q_k, a_{k+1}) = q_{k+1}$ for $k = 0, \ldots, n - 1$. 
The machine is encoded for the evolutionary algorithm as a sequence of pairs (state number, state output) on each sensor value in canonical order of state number. In our multi-objective optimization experiments, the genetic pool should allow variable length chromosomes for variable state machines; the size of finite state machines depends on the number of states and thus different members of the pool may have different length genetic representations.

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\(^2\) 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 new 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.

Three different methods for a size-modifying operator are shown in Figure 1. 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.

\(^2\)The dominating rank method defines the rank of a given vector in a Pareto distribution as the number of elements dominating the vector. The highest rank is zero, for an element which has no dominators.

The following 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.

The first operator concatenates several copies of one of the parents, cutting the result at the pre-chosen length of the offspring’s chromosome. This is repeated for the second offspring using the other parent – see Figure 1 (a).

The second operator concatenates alternately a copy of one parent and a copy of the other parent, again cutting at the pre-chosen chromosome length. A second offspring is generated similarly starting with the other parent – see Figure 1 (b).

The third operator takes a copy of one of the parents and if its size is larger than the pre-chosen chromosome length, it is cut; if not, random strings are added to build it up to the pre-chosen length. A second offspring is generated in a similar way, but instead it starts with a copy of the other parent – see Figure 1 (c).

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, bit flipping mutation can be used to change 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 as shown in Figure 1 (d).

3 Experiments with Woods Problems

Wilson suggested that temporary memory can be added to a zeroth-level classifier system (ZCS) to solve problems in non-Markovian environments (Wilson, 1994). It is implemented as a few bits in a memory register to be set or reset by actions. A Woods environment is a grid world where agents have discrete sensors and a limited number of motor actions. An agent explores the environment to find its food but there are many trees as obstacles to block moving forward. The agent can recognize the status of each of its eight neighboring cells through eight sensors and so processes only local information about its surrounding environment.

The performance of a strategy or solution to a Woods Problem is measured by the average number of time steps needed to reach a goal from all possible initial cells. An optimal solution has the minimum average steps to the goal. We suggest that an evolutionary approach as outlined above, with state transitions and memory, should be able to handle complex problems requiring long chains of actions and will find optimal solutions using minimal memory for Woods Problems. The complexity of the problems can then be defined in terms of necessary memory size. The proposed method for memory analysis, evolutionary multi-objective optimization
(EMO), was applied to Woods Problems. The penalty fitness function is defined as follows:

\[ F_g = \frac{1}{|S|} \sum_{p \in S} \min(40, d(p, G)) \]  

(1)

where \( S \) is a set of initial cells, \( G \) is the goal position and \( d(u, v) \) is the path length from position \( u \) to position \( v \).

If the goal is not reachable from a cell, then the path length becomes infinity and a high penalty value (40) is taken: 40 time steps are assigned for each initial cell and if agents can reach the goal from an initial cell, the number of time steps spent for exploration will be the path length from the initial cell to goal. The fitness \( F_g \) is the average number of time step taken to reach the goal from any empty cell. There may be one or more cells from which the agent cannot reach the goal when a strategy is applied. In this case we say the strategy cannot solve the problem.

To prove that a given Woods Problem cannot be solved by specialized control systems, for example, purely reactive systems, we need to test all possible strategies which belong to that class. This may be a very large exhaustive search. In the experiments here, a control structure is given to solve the problem and evolutionary search will be applied with the control structure to try to find the best performance. If no trial succeeds in finding a solution, then we assume that the control structure is not suitable for the problem. The proof of solvability of problems using particular control structures cannot be validated theoretically in general, but it is based on experimental sampling data.

In this paper, FSM structure was used for memory analysis. Crossover rate was 0.6 with population size 100 and mutation rate was set to 2 divided by the chromosome length. For significance statistics, 25 trials were repeated with the EMO approach over behavior performance and memory size. For each number of internal states, 95% confidence intervals of fitness are estimated by assuming \( t \)-distributed results.

### 3.1 Woods101 Problem

Figure 2(a) is a simple maze example of a Woods Problem. It was tested to show hidden states by McCallum (McCallum, 1996) and it is called McCallum’s maze or woods101. In the woods101 environment, an agent is placed at any empty cell and it can take an action towards one of four directions (up, down, left and right) each time step. Normally Woods Problems allow eight directional moves by considering diagonal movements,
but here we restrict the agent’s motor actions, making the problem more difficult. The task is to find one food cell at the middle bottom by the shortest path. If a food cell is seen as the goal, this navigation problem becomes a goal search problem.

This problem cannot be solved by purely reactive agents. At some cells with * marks in Figure 2(a), agents have the same sensory patterns but need different actions. In the picture, an agent should move left at one cell and move right at the other. It has been shown that temporary memory can help solve such perceptual aliases (Cliff and Ross, 1995, McCallum, 1996).

The Evolutionary multi-objective optimization (EMO) approach was applied to analyze how many internal states are required to solve the perceptual aliasing problem in this Woods environments. The result says two internal states are sufficient to solve woods101 problem. Even if only four sensors are used for an agent instead of eight\textsuperscript{3}, the result remains that two states are sufficient for this problem and more than two states are luxurious. The environment has six aliased positions for four-sensor agents. Sensor limitation increases the apparent complexity of the problem by generating more sensor aliases but doesn’t change the required memory for solution.

Figure 2(b) is the result of the EMO approach with four sensors. It shows for each quantity of internal memory, from one to four internal states, the distribution of behavior fitness obtained with 95% confidence intervals. Performance with two internal states differs significantly from that with one internal state but not from that with 3 or 4 states; hence we conclude that the problem requires two internal states for solution. Purely reactive agents cannot solve the problem and thus the average fitness for one internal state (purely reactive) has a high penalty value. The agent with four sensors needs more time steps to find the goal than eight sensors as it wanders around the environment, looking for the situations which can trigger proper actions.

Using the EMO approach, the best solutions are easily obtained within a small number of generations. An example of solutions is given in Table 1. It is noteworthy to see how the controller handles sensor aliases with internal memory. Cell 6 and cell 3 have the same sensory pattern. At cell 3, an agent moves right, down and down. At cell 6, it first moves right and sees a corner at right. Then it changes its state and turns left. The agent is now at cell 6. It experiences the same sensory pattern as at cell 6 but it has a different internal state and thus moves left, down and down. Cell 0, 5 and 9 also have the same sensations\textsuperscript{4}. By utilizing internal memory states, the agent can reach the goal from any initial cell. Increasing the number of aliases does not necessarily require increasing the number of internal memory states. Agents can efficiently use their internal states by distributing sensor states over memory states. The EMO approach can find compact arrangements of internal states to achieve desirable solutions. In the woods101 problem, one bit of memory is sufficient even for 4-sensor agents. Its performance has total 43 steps; this is the total length of the path from all initial positions to the goal. Its average score is 4.3 time steps, which is the fitness value shown in Figure 2(b).

\textsuperscript{3}Four sensors among eight are selected with front, back, left and right directions

\textsuperscript{4}Since agents have only binary sensors they cannot distinguish trees and food
3.2 Woods102 Problem

A new environment called woods102 was designed to have more difficult situations than the woods101 environment (Cliff and Ross, 1995). This environment is more complex than woods101: there are several cells requiring different actions with sensor aliases marked `*' in Figure 3(a). Cliff and Ross mentioned that two bits of memory are sufficient to disambiguate the sensor aliases (Cliff and Ross, 1995). We restrict the agent to four sensors as in our woods101 example above. An agent will experience more ambiguous situations with this sensor limitation.

When the EMO method with memory states ranging from one to four is applied, the result shows that this problem still requires two states as in Figure 3(b). Thus, a one-bit register can solve the problem. However, the memoryless policy — the single state machine — cannot handle perceptual aliasing and the agent cannot reach the goal starting from many of the empty cells. It has a high penalty in its fitness and Figure 3(b) shows that it is easily distinguishable from a memory-based policy. Woods102 has more sensor aliases than woods101. In this problem, increasing the number of internal states improves performance. EMO method was again applied to this problem with memory states from two to five, because one state performance is quite distant from the performance of machines with more than one state, to enable the method to concentrate its search on memory-based controllers with best performance. After 1000 generations, results indicate that two states produce the best solution of 155 time steps in total or an average of 5.96 steps to goal. The three-state strategy has 146 time steps or an average of 5.62 steps and the four-state strategy has 137 time steps or an average of 5.27 steps. More than four states have the same performance as four states. Thus, the best performance can be achieved with 2-bit registers.

3.3 Sutton's Gridworld

In Sutton's gridworld (Sutton, 1990), shown in Figure 4(a), an agent can sense eight neighboring cells and takes one of four directional moves. The environment has 46 empty cells, 30 distinct sensor states and one goal position. Littman used branch-and-bound method to
find the best memoryless policy (Littman, 1994), which takes a total of 416 steps, an average of 9.04 steps (= 416/46) to reach the goal. Especially at cells with three trees on the right side (the three cells marked with "*"), the best memoryless agent experiences ambiguous perceptions and makes inefficient movements, even though it can reach the goal from any initial cell. The agent chooses a roundabout way to escape perceptual aliasing situations.

When internal memory is used, two internal states yield an optimal strategy with performance of 410 total steps to goal, an average of 8.91 steps. Its trace is displayed in Figure 4(b). Two arrows in one cell mean that internal states are involved to give agents appropriate actions. Internal states can be seen as milestones to say which direction is appropriate and they are determined by past history that agents have experienced.

When four out of eight sensors are taken as in the woods101 experiments, its performance is worse than eight sensor experiments. Agents have more sensor aliases for a tree on the right. The memoryless strategy cannot solve this problem, unlike the eight sensor version: there are several cells from which the goal cannot be reached. The best policy with two states for this problem has a performance of total 418 time steps, an average of 9.08 steps to goal. The best strategy as shown in Figure 4(c) is different from the eight sensor strategy in Figure 4(b). An agent chooses to go down when it first sees a tree on the right side. If there are trees on the right and below, it is bounced back to go up with an internal state marker. Otherwise, the agent moves right. The EMO method shows that more than two states give similar performance as displayed in Figure 4(d). One can say that two states are sufficient for this problem with four sensors.

However, 418 time steps is not an optimal solution with four sensors. An optimal strategy with four states was found with a total of 416 time steps. In this strategy, two additional states record the perceptual situation (marked "*" in Figure 4(c)) where a tree is detected on the left side and after moving right two steps, another tree is seen on the right side. This state blocks going down in the rightmost marked cell and the agent can go straight up, while two or three state machines waste time going down and returning to the cell from below.

If an agent can recognize every feature of the entire environment, it will obtain an optimal performance of 404 steps from all empty cells, average of 8.78 steps to goal. However, the agent has only local information about its neighboring cells. Internal memory states help agents overcome the shortcomings of this local information. The performance of the memory-based policy is quite close to the optimal performance of 404 steps. The number of sensors is also an important factor to obtain efficient solutions. In this problem, an optimal memoryless strategy with eight sensors has the same performance as an optimal four-state strategy with four sensors. Reducing sensors or restricting sensor range often requires increasing internal states. In other words, agents with more sensors will have a higher chance of succeeding with purely reactive controllers.

### 3.4 Woods14 Problem

Woods14 is a Markovian environment designed by Cliff and Ross (Cliff and Ross, 1995) with eight sensors available on the agents. It has a simple linear path of empty cells and agents need to go through a field of trees. At each cell an agent experiences a different sensation and only one appropriate action among eight directional moves should be taken to reach the goal. Cliff and Ross showed that ZCS even with internal memory could not
succeed in developing desirable control systems for this problem (Cliff and Ross, 1995). Woods14 shown in Figure 5(a) is a very difficult problem for ZCS to solve because it requires long chains of actions before the reward is given. Also ZCS tends to create a conflicting overgeneral classifier producing inappropriate actions, that is, a classifier that matches multiple sensations and does not cover appropriate actions for such sensation. This happens when the classifier’s sensor pattern contains too many don’t care (‘#’) terms.

An FSM with eight sensors, allowing 256 distinct sensor states, was evolved to solve the woods14 problem with a perfect score, average 10 time steps. Without difficulty, evolutionary computation succeeded in finding an appropriate strategy. When only four sensors are used as in the woods101 problem, this woods14 problem generates complex situations. Most empty cells have sensor aliases because of the sensor limitation. The problem becomes a non-Markovian environment.

An EMO analysis for four-sensor agents shows that the memoryless policy, as expected, fails to find solutions to the woods14 problem. Even two or three states were insufficient; there are starting cells from which the agent cannot reach the goal. As shown in Figure 5(b), there is a hierarchy of complexity with memory size. For two or three internal states, performance intermediate between the memoryless strategy and the four state strategy can be achieved. It depends on how many empty cells are successfully built into the path to the goal. An optimal memoryless policy has 5 empty cells that can be connected to the goal. The other 14 cells fail to reach the goal. With two states, the goal is reachable from 11 empty cells. Three states gives 17 successful cells among the total of 19 cells. The diagram in Figure 5(b) shows how closely a given number of states can achieve solutions.

One of the best solutions with four state machines, see Table 2, shows how to handle perceptual aliasing. Cell 13, 14, 15 and 16 have the same sensory pattern. They are surrounded by trees in all four directions. Cells 13 and 14 need the same action of moving to the lower left, while both cell 15 and cell 16 need an action of moving to the upper left. The strategy first tries to move lower-left and if the situation is unchanged, it then tries to move upper-left. The two actions are sequentialized with internal states, depending on sensations. Cell 0, 5 and 17 have sensor aliases. Cells 0 and 17 prefer to take an action of moving down, but cell 5 needs an action of moving to the upper-left. These kinds of sensor aliases require internal states to escape cul-de-sacs. Internal states place markers on each sensor alias state and then block taking the same action. For instance, an agent first moves down at cell 5, changes its internal state, returns to cell 5 and then moves to the upper-left. The agent does not memorize sensation values and put milestones at particular sensations. In this way internal memory can be easily represented by Boolean logical values or finite states.

If the number of sensors is decreased, more sensor aliases may occur, but not necessarily. Instead of four or eight sensors, two sensors are selected among eight sensors to see the influence of sensor limitation. First, FSMs with two sensors, front and right sensors, are evolved and the EMO analysis in Figure 6(a) shows that three internal states are insufficient to solve the problem. There are 9 initial cells which cannot reach the goal for the best three-state machine. (In experiments with one or two sensors, penalty 80 is used instead of 40 in equation 1 to see a clear distinction of solvability for each state machine.) In this case four state machines can solve the problem with a total 301 time steps to reach the goal from all empty cells while four-state agents with four

---

Table 2: A solutions of woods14 problem (each pair of numbers in the trajectory represents motor action and state number in FSM. 0: down, 1: right, 2: up, 3: left, 4: lower right, 5: upper right, 6: upper left, 7: lower left.)
sensors have a score of total 198 time steps\(^6\). When two sensors in front and left are selected, it still requires four states to solve the problem and the performance is improved with total 245 time steps to the goal as shown in Figure 6(a)-(b). For three state machines, there are two initial cells which cannot reach the goal.

The selection of two sensors on the left and right makes the woods14 problem more difficult. Even with five internal states, there were still three cells unable to reach the goal. Six or more states could solve the problem. Figure 6(c) shows a significant change of performance between five states and six states. If the number of sensors is limited, what sensors are selected will be an important issue to solve problems and produce good performance.

When only one sensor, the front sensor, is used, four states are sufficient to solve the problem with a total 371 time steps to the goal. This agent unexpectedly outperforms the one with two sensors on the left and right. Three state machines had 14 cells which failed to reach the goal. Restricting sensors generally degrades the performance, but does not imply definitely that more internal states are required, as is evident from the above results of two sensors on the left and right versus only one sensor in front. Clearly, sensor morphology is important for a gent problems.

It is also notable that the two sensor results including the front sensor are better than the result with only the front sensor, even though all these require four states to solve the woods14 problem. In Figure 6(d), the best performance of the one sensor experiment — even with many states — is significantly worse than the performance of those two sensor experiments including the front sensor.

Thus, we can build a partial ordering relation on the performance for a various number of sensors. For \( S_A \subseteq S_B \), where \( S_A, S_B \) are a set of sensors, the performance

\(^6\)These score solutions may not be optimal, and they are obtained with 5,000 generations.
function $G$ keeps the partial order for the same amount of memory, $M$.

\[ G(S_A, M) \leq G(S_B, M) \]

More sensors can see the environment better. Also for $M_1 \leq M_2$, the partial order relation holds with a given set of sensors.

\[ G(S_A, M_1) \leq G(S_A, M_2) \]

Using the above property, the following partial order relation may be inferred for the performance of robotic tasks or agent problems.

\[ G(S_A, M_1) \leq G(S_B, M_2) \]

where $S_A \subseteq S_B$, and $M_1 \leq M_2$.

When a pair of sensor sets and memory is not ordered with each other — for example, where there is no subset relation between $S_A$ and $S_B$ or $S_A \subseteq S_B$ and $M_1 > M_2$ — the performance order cannot be predictable. It will depend on tasks and environmental situations. The partial order relation in the woods14 environment is displayed in Figure 7. If the front sensor is included, at most four memory states are required to solve the problem. When the number of sensors is increased, the performance becomes better. Thus, it is presumed that agents with front, left and right sensors will solve the problem with four states and have performance in the range ($198 \leq G \leq 245$).

4 Conclusion

Woods Problems are goal search problems with perceptual aliasing. Four Woods Problems, woods101, woods102, Sutton's gridworld and woods14, were solved with a novel Evolutionary Multi-objective Optimization (EMO) analysis. The woods101 environment requires two internal states to solve the problem, and woods102 also needs two states to reach the goal from any initial cell but presumably four states for an optimal solution. In Sutton's gridworld problem, two internal states yield an optimal strategy for eight sensors. Reducing the number of sensors to four sensors requires four states for an optimal solution. Woods14 requires four states for an optimal solution with four sensors, while memoryless agents with eight sensors can solve the problem. The number of sensors influences the necessary memory size, since reducing the number of sensors or sensor range has the potential of creating more sensor aliases. The memory size required to solve a problem is not necessarily proportional to the number of aliases. Instead it is more concerned with what kind of sensor aliases are experienced in the path to goal. Placing the same sensory patterns serially in the path may increase the memory requirement.

Memory plays an important role to solve sensor aliases and can improve an agent's performance. The EMO approach was effective to find the minimum number of memory elements for a variety of tasks and to visualize a hierarchy of performance depending on available memory. All the problems described show that purely reactive systems are inferior to memory-based systems. However, the results also show that a large number of memory elements are not necessarily required. A small number of bits are sufficient to evolve desirable behaviors in several Woods Problems, although many complex control designs such as recurrent neural networks have been suggested for agent problems. The problem set provided in this paper may be simple, but the results indirectly imply that reactive systems with small-sized memory are effective in agent problems. Also the number of sensors is correlated with memory size for desirable performance.

The Woods Problem may be seen as a model of agent navigation tasks. In robotic tasks, robotic agents ex-
perience perceptual aliasing in various situations which may cause similar decision problems in which their actions cannot be completely determined by perceptions. The EMO approach to the Woods Problem can be easily applied to robotic tasks requiring a small amount of memory or no memory, in order to measure the necessary memory amount for the tasks as well as to see the memory effect, assuming suitable task simulation environments are available.

References


Brooks, R. (1991). Intelligence without reason. AI memo 1293, Artificial Intelligence Laboratory, MIT.


