Agent Architecture and Cooperation Protocol for an Episodic and Inaccessible Environment

Hamid Haidarian Shahri*, A. Abdolahzadeh Barforush

Intelligent Systems Laboratory
Faculty of Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
*hhaidarian@aut.ac.ir

Abstract: Distributed search in inaccessible environments is an important problem in robotics and multi-agent systems (MASs). Vacuum cleaner agent is a classic example, which is particularly interesting, since it encompasses many aspects of search problems in MAS. In this paper three different architectures that are complimentary to one another have been studied to determine their suitability for distributed searches. Various interaction protocols for coordinating an agent society were simulated and interestingly competition with blackboard produced the better results. The effect of environment accessibility on performance was also analyzed. It was deduced that in complex environments or preset societies, where it is impossible to change the agent interaction protocols, the same performance can be gained by increasing the environment accessibility for individual agents. The effects of utilizing memory to keep track of agent's internal state and introducing complexity into the environment have also been studied.

1 Introduction

Distributed search using a society of autonomous agents without engineering the agent too much is an interesting focus for discussion, because it touches on a number of issues that have received little attention in the classic AI literature. The task requires both symbolic and continuous control and the use of sophisticated perception. Vacuum cleaning agents have been implemented in our experiments as a case study and an example which covers all the features of the distributed search problem.

The problem definition is in section two. In section three, several different architectures, namely reactive, symbolic and intelligent, have been devised for the vacuum cleaner agent and it is shown that these architectures are complementary to one another and a successful implementation must include different aspects of these architectures. The agent architecture has been thoroughly analyzed and its suitability for doing the cleaning task is shown.

In section four, the simulation and implementation of various interaction protocols are described. The black board system was used as an interaction protocol and its effectiveness on the performance of the multi-agent system was measured. We also simulated and evaluated organizational structure as a mechanism for task distribution among agents and to help agents cooperate in the environment and interact more consistently. In this paper we have also studied the effects of exactness of perception, which is closely related to the problem of accessibility of the environment. It is observed that, in complex or static agent based systems where the protocols of the society can not be manipulated, similar performance can be gained by increasing the environment accessibility for individual agents. The results are summarized in the conclusion, in section five.

2 Defining the Problem

Vacuum cleaning is an excellent example of a real-world problem that can be made arbitrarily simple, or arbitrarily complex, depending on exactly how the task is defined. For example, the following questions all seem like legitimate potential elaborations to the basic scenario of vacuuming a room that is empty except for a few, immovable obstacles:

- Will children or other moving agents be present? Will they be cooperative or should they be ignored?
- Should the vacuum be responsive to long term user strategies? For example, “Vacuum the living room every morning but vacuum the bedroom only Saturday when everyone has left the house.”

An appropriate solution to the autonomous vacuum cleaning problem is determined by the details of the task specification. The software architecture, knowledge representation, and programming methodology used for one task may not work well for another.

It is important that we define the problem carefully. We must realize that if we define the problem to be a simple reactive one, then a more complex architecture will be overkill and appear irrelevant. We must allow enough complexity into the problem to let these architectures make their point, if they can.

3 Agent Architecture

A central aspect of the vacuum cleaning problem is that the vacuum cleaner must move. This brings up a host of immediate issues that can alter the problem considerably. First, there is the question of whether objects in the room to be cleaned are fixed or can move. Most real furniture moves around from time to time and a vacuum cleaner should be able to cope.

Similarly, there may be transient objects that are only in the room sometimes and effectively look like objects that have moved. In either case, any a priori knowledge the vacuum may have will necessarily be unreliable.

A second issue is whether or not there will be objects moving in the room while vacuuming is under way. Realistically, a vacuum cleaner will almost always encounter moving objects on occasion because someone is bound to enter the room even when they are forbidden to do so. Thus, a vacuum cleaner must have a strategy for coping with moving objects even if it is as simple as stopping or trying to go around.

Third, is the question of whether the objects in the room to be vacuumed are complex or simple with respect to the vacuum cleaner? Simple objects can be vacuumed around by tracing their perimeter (or some other simple strategy) while complex objects may require intricate steering maneuvers to ensure that nooks and crannies are cleaned effectively. Increasing the complexity of objects can have a significant effect on the kind of software architecture required to steer the robot.

3.1 Using Reactive Strategies

The simple vacuum task is a natural candidate for the use of reactive approaches to robot control. Typically, such solutions would use very little state and simple sensing strategies to differentiate rug from non-rug. The resulting vacuum would embody some strategy for covering the entire room, such as a random walk or a slow spiral outward that relies only on local sensing. It would be able to avoid non-rug areas (perhaps differentiating between non-rug floor and actual obstacles) and it may repeat the vacuuming procedure some number of times to account for moving obstacles that could have caused it to miss parts of the rug earlier. Such strategies might be implemented using subsumption [1], ALFA [2], or any other languages that map the current state into actions through a decision network or collection of concurrent processes.

A reactive approach to vacuuming is attractive because the simple vacuuming task contains a good deal of uncertainty, unpredictability, and lack of knowledge. These things are exactly what reactive approaches to robot control are designed to cope with. The driving force behind the reactive idea is the need to deal quickly and effectively with a changing and uncertain environment. The idea works well because a variety of real world tasks are easily achieved using simple reactive strategies that require only immediate local sensory data. The simple vacuum problem is an example of such a task.

3.2 The Symbolic Vacuum

A more sophisticated autonomous vacuum would be able to apply different vacuuming strategies at different times in different situations. Differentiating situations that require alternative strategies will often depend on sensor information, experience, instructions, and vacuuming knowledge.

For example, consider the need to vacuum around and under complex furniture. One approach is to examine the piece of furniture (or perhaps just the spaces needing cleaning) and retrieve or create a plan to vacuum that area effectively.

A similar situation arises when the floor can be littered with objects that need to be picked up and put away, or moved and vacuumed underneath. A natural solution is to classify each object and choose a plan to move it appropriately. One might also want the vacuum to adopt different vacuuming strategies when different types of object are moving around the room. Perhaps adults can be safely ignored but when a child enters the room the vacuum should stop and wait for the child to leave.

The vacuum cleaner may also need to carry out special instructions from its user. For example, the user may want the vacuum to follow different plans in different situations: vacuuming the living room only when no

one is home, or vacuuming the west side of the room first and then the east. Even if we ignore the problem of how the user specifies that information, the robot still has to be able to tell situations apart and apply different plans.

Let us define the symbolic vacuum problem as the task of cleaning rooms that contain complex furniture and moving and stationary objects that have to be treated in different ways. Furthermore, the vacuum should be capable of following a variety of user instructions in different situations.

3.2.1 Symbols and Representations

Solving the symbolic vacuum problem requires gathering information, classifying the situation, and choosing a plan to apply. The vacuum might adopt a strategy for methodically vacuuming the local region of a carpet and mapping its extent. Equipped with the map, the system could keep track of those regions of carpet it had completed and those areas of the room not yet explored. It might also attempt to classify obstacles and non-rug areas as pieces of furniture that cannot be moved, those that can be moved, items on the floor that should be put somewhere else, or objects (like children and pets) that are likely to move on their own. The vacuum would make and use plans for exploring and cleaning areas of carpet as they were found and for dealing with the objects it encountered along the way. Such a system would have a good idea of when it was done because it could tell by its map and its object classifications. A variety of architectures for managing plans dynamically would be appropriate for such a system: PRS [3], and Universal Plans [4], to name just a few.

Situations and plans are the natural vocabulary for discussing solutions to the symbolic vacuum task because we, as humans, seem to conceptualize the problem in those terms. We can recognize and classify objects and situations with apparent ease and we communicate knowledge of how to deal with different situations in terms of prescribed actions. In effect, we divide the world into symbols and our activities into discrete actions. The symbolic vacuum task fits this mold.

3.2.2 Using Symbols and Reactivity

The primary problem with the use of robot architectures that depend on symbolic classifications and discrete plans of action is that they often have trouble remaining reactive. While it seems natural to classify the world into states and select different plans in different states, none of the simple vacuum problems have gone away. The robot must continue to assume that unpredictability and lack of knowledge exists and it must continue to use reactive techniques for actually taking action in the world.

Strategies for local navigation, obstacle avoidance, and sofa or wall-following are much more easily thought of and coded up as continuous processes (or behaviors, routines, or skills). Symbolic systems have a hard time with such problems. On the other hand, mapping, object classification (and memory) and the selection of different strategies for coping with different objects in different situations are often more productively thought of and coded up as symbolic programs. Once a reactive system starts to build a map, choose among object recognition strategies, and select control actions based on both the external state and its internal map (symbolic), it becomes harder and harder to describe in terms of concurrent continuous processes (reactive).

Thus, the symbolic vacuum problem does not stand on its own; it is an extension of the simple vacuum task. No solution to the symbolic problem can ignore the reactive requirements of the simple task. Software architectures for real vacuums must include and coordinate both symbolic plans and continuous reactive processes.

3.3 The Intelligent Vacuum

The vacuum cleaning problem can be extended further to include the ability to negotiate with its user and understand and respond in a reasonable way to commands like: Stop that, Vacuum here later, Do under the sofa first or Stay away from the baby. The system must be able to understand its own actions at multiple levels of abstraction to understand and respond to such requests properly.

Such commands should also continue to influence the system's behavior for some time in the future. We will define the intelligent vacuum problem as the task of cleaning the carpet in any reasonable situation and responding to user input (or any other stimulus) in a reasonable way.

3.3.1 Using Intelligence

The intelligent vacuum builds on both the symbolic and simple vacuuming tasks. An intelligent vacuum needs to cope with the real world reactively, represent plans of action, perceive and classify objects and situations, and understand its plans and actions well enough to alter them appropriately while conversing with a user. We believe that discussing vacuuming problems that are less complex than the simple task is pointless.

The simple vacuum requires the minimum capabilities demanded by the real world. We also believe that the simple vacuum will find little application in the real world. It just isn’t responsive enough to user requirements. A symbolic vacuum is probably necessary for industrial applications and nothing short of the intelligent vacuum will do in the home.

4 Simulation and Evaluation of Interaction Protocols

For producing a coordinated agent society, an interaction protocol that governs the relationship between different agents is required. In this project three different interaction protocols, namely competition with and without a blackboard and cooperation using organizational structure has been evaluated. The Power Builder (6) programming environment was used for coding and simulating of the multi-agent system.

The agents are designed to be autonomous and act independent of one another in parallel. The interface of the program provides a graphical user interface for setting the number of agents, rubbish and blocks in the environment and also tracks agent movements online, as depicted by figure 1. In the right pane there is a record of settings and actions in the previous runs of the program and the results are recorded for future evaluation.

Different numbers of agents have been used in the society and the results are averaged over 10 runs of the program. Relative number of moves required for collecting the rubbish in the field shows the amount of time needed for the task to finish and is a suitable measure for the effectiveness of the distributed search in the problem space. For interaction protocols, competition with a blackboard and without a blackboard [5] and cooperation using organizational structure [7] were implemented and tested.

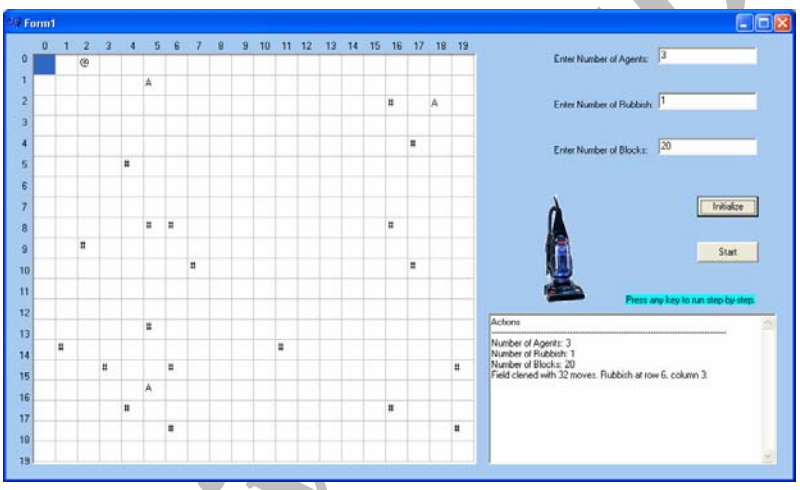

Figure 1: Graphical user interface of the program

At first, by intuition we believed that using organizational structure to allocate tasks to agents should be very effective. Interestingly, figure 2 demonstrates that using competition with blackboard even produces better results. This must be largely due to the internal state tracking and the planning, implicitly hidden in the blackboard system.

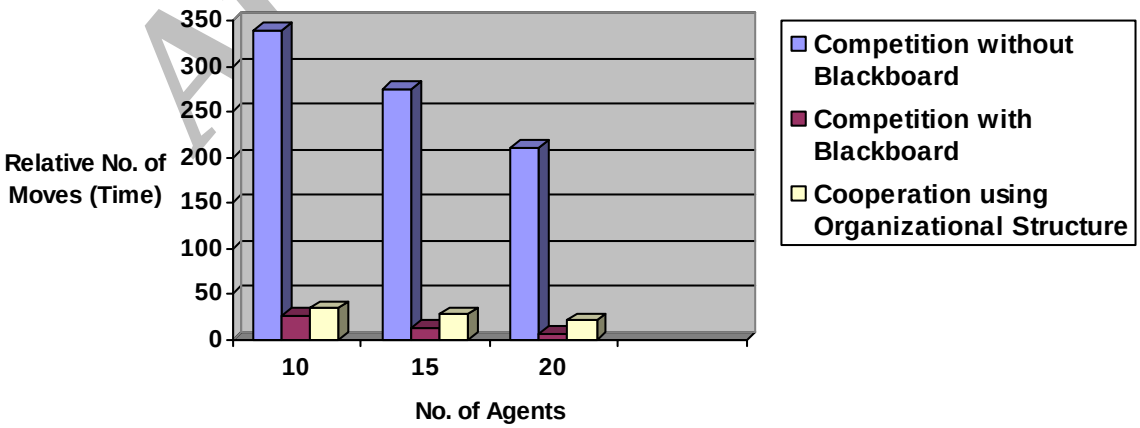

Figure 2: Comparison of different interaction protocols for various societies and following one rubbish, without any blocks in the field

Figure 3 illustrates the architecture of the blackboard, knowledge sources or agents and control components in the system [6]. It shows the workflow in a multi-agent system which uses a blackboard.

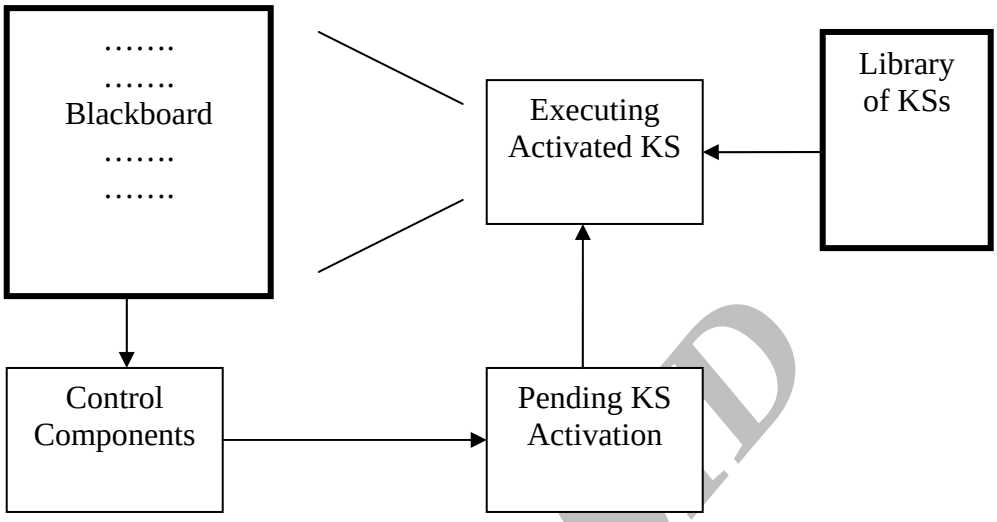

Figure 3: The architecture of the blackboard system

By observing figure 4 carefully, it can be inferred that, especially in physical real-world environments, when the problem is too difficult or when the codes for developing a suitable interaction protocol, like competition with blackboard is complex, performance defects can be compensated by using stronger sensors to make the world more accessible to the agents. For example, in figure 2, when we have 75% accessibility and competition with blackboard can not be implemented, we can implement coordination using organizational structure and employ sensors that make the world 100% accessible, to gain the same performance, when using cooperation with organizational structure.

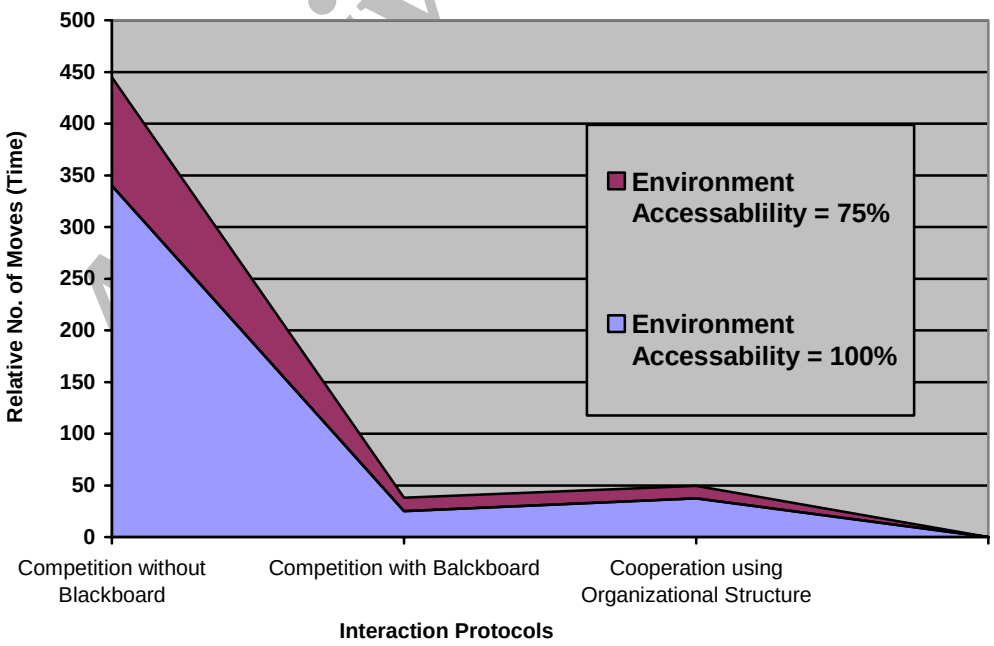

Figure 4: The tradeoff between interaction protocols and environment accessibility; three agents following one rubbish, with no blocks in the field

Table 1 shows that by using memory for keeping track of the internal state of an agent, the number of moves required for cleaning the field is reduced considerably, which in turn entails better agent performance.

Table 1: Internal memory has an effective role in agent performance

	Agent without memory for internal state	**Agent with memory for internal state**
No. of moves (Time)	340	26.6

In figure 5, different environments for the agent society have been tested. The agents always use competition with blackboard in these experiments. It is observed that, since increasing the number of blocks in the environment causes the search space to be reduced, the agents can finish the task more quickly. It should be noted that this fact holds only if this increase in the number of blocks does not introduce new difficulties in the agent movements.

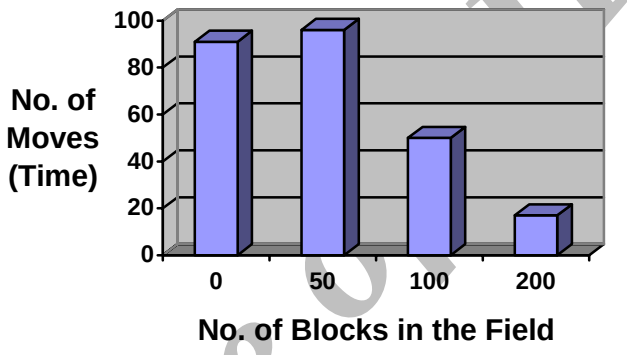

Figure 5: Environment complexity with agents using competition with blackboard; three agents following one rubbish in the field

5 Conclusions

Distributed search is a crucial problem in robotics and multi-agent systems. As a case study for distributed search in inaccessible environments, autonomous vacuum cleaning is very interesting, because it encompasses a variety of different tasks that demand a progressive architecture from reactive, to symbolic, to intelligent. Each task requires a different architecture, yet that architecture must continue to embody key ideas from its simpler versions. At the low end of vacuuming problem complexity, a reactive architecture capable of dealing with misplaced furniture and the occasional moving objects, seems like the simplest solution to consider.

As the need to classify objects and vacuum around them in different ways becomes important, the architecture must change to one that represents explicit object classes and corresponding vacuuming plans. This change is required, because the nature of the solution shifts from one naturally phrased, in term of concurrent processes to one naturally phrased, in terms of goals and plans. However, while much of the knowledge and control structure within this new symbolic architecture will be symbolic, actual actions taken in the world must remain reactive to cope with the same uncertainties and changes that, the simple architecture handles well.

Additionally, as the need to interact with human users becomes important, the architecture must further evolve to include a clear understanding of the greater environment, the robot's own goals and plans, and the apparent goals and plans of the user. Such an intelligent vacuum cleaner must continue to deal with symbols and reactive processes even as it attempts to recognize and account for the needs and desires of other agents.

After the design of the architecture, some coordination protocols are required for the society of autonomous agents. By simulating the vacuum cleaner problem, different interaction protocols were implemented and interestingly enough, the effectiveness of competition with blackboard was discovered. The tradeoff between environment accessibility and interaction protocols was shown and utilized to achieve better MAS performance. Internal memory also improved agent's actions considerably. At last, the effect of introducing different levels of complexity into the environment was analyzed.

References

[1] Brooks, R.A., "A Robust Layered Control System for a Mobile Robot." *IEEE Journal of Robotics and Automation*, RA-2(1), March 1986.

[2] Gat, E., "Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots." *PhD thesis*, Computer Science and Applications, Virginia Polytechnic Institute, 1991.

[3] Georgeff, M.P., Lansky, A.L., and Schoppers, M.J., "Reasoning and Planning in Dynamic Domains: An Experiment with a Mobile Robot." *Tech Note 380*, AI Center, SRI International, 1986.

[4] Schoppers, M.J., "Universal Plans for Reactive Robots in Unpredictable Environments." In *Tenth International Joint Conference on Artificial Intelligence (IJCAI)*, Milan, Italy, August 1987.

[5] Shaw, M., and Garlan, D., "Software Architecture: Perspectives on an Emerging Discipline." *Prentice Hall*, 1996.

[6] Weiss, G., "Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence." *The MIT Press*, 1999.

[7] Wilson, F.A., "Computer Based Systems and Organizational Structures: Designing the Ghost in the Machine." *Proceedings of the IFIP WG8.2 Working Conference on Information Technology and New Emergent Forms of Organizations*, Ann Arbor, Michigan, USA, 11-13 August 1994.