Software Architectures for Agents in Colonies

George A. Bekey & Arvin Agah

Institute for Robotics and Intelligent Systems, Computer Science Department
University of Southern California, Los Angeles, California 90089-0781
bekey@robotics.usc.edu & arvin@robotics.usc.edu

Introduction

We have developed a novel architecture for agents in colonies, in order to investigate certain forms of group interaction. Specifically, we are interested in the extent to which overall goals for a colony can be achieved when each agent is only aware of limited local goals, whether the architecture allows for emergence of unexpected behavior, and whether explicit communication among agents facilitates or hinders task performance. Our architecture supports several forms of learning.

We have studied large colonies of agents (as many as 100) in simulation experiments, where they carried out fetch-and-carry tasks in the presence of predators and with limited energy reserves. In addition, we have fabricated a physical colony of four agents, with the same architecture, to insure that the behaviors we observed in simulation were also present in the hardware implementations.

Tropism System Cognitive Architecture

The architecture used by each robot (agent) in our colony to sense and act upon the world is termed the *Tropism System Cognitive Architecture* [Agah and Bekey, 1994]. This architecture is based on the tropisms of the robot, i.e., its likes and dislikes. Such an architecture transforms the robot's sensing of the world to appropriate actions and enables it to survive and function in an uncertain world. The concept of positive and negative tropisms as principal mechanisms of intelligent creatures was first discussed in [Walter, 1953].

Our simulated world includes robot agents (whose tropisms include finding and gathering objects), predator agents (which have no useful goals, other than to find and immobilize robots), the objects to be gathered, fixed obstacles, and a home base where the robots go to recharge their energy banks. Each agent is capable of sensing other entities in the world and their states. For instance, the entity that is sensed could be a predator, and its state could be "active". Denoting the set of entities by $\{\varepsilon\}$, the set of entity states by $\{\sigma\}$, the set of robot actions by $\{\alpha\}$, and the tropism values by τ_i, with $0 \le \tau_i \le \tau_{max}$, a tropism element can be represented by a set of relations. In each relation, an entity and the state of that entity are associated with an action by the robot, and the associated tropism value.

$$\{ (\varepsilon, \sigma) \rightarrow (\alpha, \tau) \} \tag{1}$$

The larger the tropism value, the more likely is the agent to perform the action. This expression indicates the way in which perceptions are mapped into actions. The entity, its state, the robot's action and the tropism value can then be represented by a tuple of four elements, referred to as the *tropism element.*

$$(\varepsilon, \sigma, \alpha, \tau) \tag{2}$$

The complete tropism system of a robot can be presented by Ξ, the set of tropism elements. As the tropism system can dynamically change, Ξ is a function of time.

$$\Xi(t) = \{ (\varepsilon, \sigma, \alpha, \tau), (\varepsilon', \sigma', \alpha', \tau'), \ldots \} \tag{3}$$

Once a robot performs a sensory sweep of its surroundings (available sensory area), the set of the tropism elements are checked for any matching entity and entity state. For all the matched cases, the selected action and the corresponding tropism value are marked. However, actions of living creatures are

not strictly determined by perceptions. Hence, we have made the assumption that the tropisms are related to the probability of particular actions by the agent. Hence, the selection of one action from the chosen set is done by the use of a biased roulette wheel. Each potential action is allocated a space (slot) on the wheel, proportional to the associated tropism value. Consequently, a random selection is made on the roulette wheel, determining the robot's action.

Without learning, the static tropism architecture is predetermined and the tropism elements would remain constant, i.e.:

$$\Xi(t+1) = \Xi(t) \tag{4}$$

Such an architecture, while interesting, does not allow for improvement of performance with experience. Hence, we incorporated learning in the system.

Learning

The learning tropism architecture allows the robot to dynamically change its tropism elements, based on its experiences in the world. The robot should be able to autonomously add tropism elements, and modify the existing ones. The deletion of tropism elements is implemented by the modification of the actions of the existing elements. With the learning tropism architecture, the robot learns in three types of situations: learning from perception, learning from success, and learning from failure.

Learning From Perception: The learning tropism architecture enables learning from perception, when a novel perceptual situation is encountered. This type of situation occurs when a sensed entity in a given state is encountered for the first time by the robot. In such a case the robot must determine how to deal with the novel circumstance. The learning tropism system must automatically develop a new tropism element. The system selects a random action for the novel situation and assigns the initial tropism value to it. The newly added tropism element which is learned from perception, can prove to be useful, or it could become unusable. Both such cases are handled by learning from failure and learning from success. Denoting the sensed entity by ε, and the entity's state by σ, the new tropism element will have the random action α_{random}, and the preset initial tropism value $\tau_{initial}$:

$$\Xi(t+1) = \Xi(t) \cup \{ (\varepsilon, \sigma, \alpha_{random}, \tau_{initial}) \} \tag{5}$$

The predetermined value of the initial tropism value could be replaced by a random setting of such value.

Since each agent has a finite capacity for tropism elements, this limit will eventually be reached as a result of continuing new perceptual situations. In such a case an existing tropism element must be removed from the system to make space for the newly created tropism element. The selection of an element to be deleted is based on the chronological order of the element formation. Among all elements of the set, the oldest one (least recently created) would be deleted, and the new tropism element would then be added. Denoting the oldest element with $(\hat{\varepsilon}, \hat{\sigma}, \hat{\alpha}, \hat{\tau})$, the new tropism system will be determined:

$$\Xi(t+1) = (\Xi(t) - \{ (\hat{\varepsilon}, \hat{\sigma}, \hat{\alpha}, \hat{\tau}) \}) \cup \{ (\varepsilon, \sigma, \alpha_{random}, \tau_{initial}) \} \tag{6}$$

The deletion of a tropism element in the tropism system architecture is analogous to "robot forgetting". The oldest learned concept is the first one forgotten. Two other types of methodologies could be utilized to select the deletion candidate. In the first method, a count variable is associated with each tropism element, enumerating the number of times an element is used. The deletion procedure would then select the least used element. The second alternative would be to time stamp each element as it is used. The deletion would then select the most dormant element, i.e., the element least recently used.

Learning From Success: Once a tropism element is selected by the robot and the associated action proves to be feasible and useful, the action by the robot will be successful, and the robot can learn from such success. The selected tropism element is then updated by increasing its tropism value. The tropism increment value $\tau_{increment}$ is predetermined, although it is possible to assign different increment

values based on the actions' outcome. Since there is an upper bound to the tropism value in an element, an increase of the tropism value beyond the maximum is not possible:

$$\Xi(t+1) = (\Xi(t) - \{ (\varepsilon, \sigma, \alpha, \tau) \}) \cup \{ (\varepsilon, \sigma, \alpha, \tau + \tau_{increment}) \} \quad (7)$$

Learning from success will not only strengthen the elements that were initially part of the system, but also strengthen the elements that were formed as part of learning from perception.

Learning From Failure: Learning from failure takes place in cases where the action selected by the robot proves infeasible. The reason for such a failure is the randomness that is associated with action selection in learning from perception. The robot does not know how to deal with a novel situation, and hence it makes a guess. Learning from failure enables the robot to recover from a previous wrong guess. The associated action of such a tropism element is replaced by a new random action. The tropism value is not changed, as it is most likely the initial value, since the element has not been used:

$$\Xi(t+1) = (\Xi(t) - \{ (\varepsilon, \sigma, \alpha, \tau) \}) \cup \{ (\varepsilon, \sigma, \alpha_{random}, \tau) \} \quad (8)$$

The ability to learn from failure is also useful in cases where there are changes to the world, so that actions once possible are no longer feasible. Such tropism elements will be eventually modified by the system.

Results

We have used the architecture outlined above in a large number of experiments using agent colonies, both in simulation and in hardware. The results obtained include the following:

1. While each agent operates only on the basis of its own likes and dislikes, the colony can perform global tasks, such as gathering objects distributed throughout the world.
2. The performance of the colony, as measured by the number of objects gathered, the total energy consumed by the agents and the number of predators destroyed, increases with learning.
3. The addition of evolution to the colony (simulated using genetic algorithms) results in further improvements in performance.
4. Addition of explicit communication among the agents may improve colony performance if the communication radius is large enough. However, some smaller radii result in the emergence of unexpected "herding" behaviors among the agents.
5. When tasks require the cooperative behavior of two agents for completion, such behavior emerges in the simulation.

Conclusion

Based on our experience to date, it is evident that much remains to be learned about architectures for agents in colonies. While we have observed the emergence of cooperative behavior, we expect that competitive behaviors will also appear with different choices of architectural parameters.

Bibliography

Agah, A. and Bekey, G. A. (1994). Autonomous mobile robot teams. In *Proceedings of the AIAA/ NASA Conference on Intelligent Robots in Field, Factory, Service, and Space,* Houston, Texas, Vol. 1, 246-251.

Bekey, G. A. (1977). Models and reality: some reflections on the art and science of simulation. *Simulation*, November: 161-164.

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

Walter, W. G. (1953). *The Living Brain.* W. W. Norton & Company, Inc., New York.

Answers to Questions

The answers to the posed questions are addressed strictly from the experience of the authors in using the Tropism-based Cognitive Architecture.

Coordination -- How should the agent arbitrate/ coordinate/cooperate its behaviors and actions? Is there a need for central behavior coordination?

Our agents coordinate their behavior and actions based on the tropism values and the local sensory information. The sensed world and tropism values lead to the likelihood of certain actions. However, it should be noted that the global goals, which exist in the mind of the experimenter, only appear in the choice of tropism values associated with certain perceptions, i.e., the agent will like to pick up objects. The agent does not have an explicit representation of the overall goals and hence cannot use them to coordinate its actions. In our architecture the coordination of the agent's behaviors is implicit, but it must be present, as otherwise there will be conflicts yielding chaotic behavior.

Interfaces -- How can human expertise be easily brought into an agent's decisions? Will the agent need to translate natural language internally before it can interact with the world? How should an agent capture mission intentions or integrate various levels of autonomy or shared control? Can restricted vocabularies be learned and shared by agents operating in the same environment?

We have automated the solution to the interface problem, by designing a system that can transform the tropism cognitive system of an agent to a form readable by humans. Similarly, the mission intention is entered by the human experimenter in a simple language. The use of natural language for communication from or to the human would simply add a translation layer to our architecture, but it would make it more accessible to general use. Also, restricted vocabularies were used in the form of simple messages by the agents to communicate in a world performing certain tasks. This is a simple form of communication and more complex methods need to be investigated.

Representation -- How much internal representation of knowledge and skills is needed? How should the agent organize and represent its internal knowledge and skills? Is more than one representational formalism needed?

The internal representation of the knowledge should be the minimum amount required to allow the agent to perform its tasks in order to achieve the goal. We organize the knowledge into the tropism elements. The initial knowledge could be obtained from the humans, or set randomly as learning combined with simulated evolution allow for obtaining of additional knowledge by the agents. Agents were equipped with a limited set of skills, necessary to deal with hostile entities, survive, and perform the sub-tasks required of them. The only advantage of a comprehensible form of representation is the fact that we can understand it. If a good translation scheme is available from the agents representation of knowledge to a natural language and vice versa, the internal representation need not be easy for us to understand, as long as the agent understands it and functions with it.

Structural -- How should the computational capabilities of an agent be divided, structured, and interconnected? What is the best decomposition/ granularity of architectural components? What is gained by using a monolithic architecture versus a multi-level, distributed, or massively parallel architecture? Are embodied semantics important and how should they be implemented? How much does each level/component of an agent architecture have to know about the other levels/components?

The structural architecture of the robot, be it a monolithic architecture or a multi-level architecture, is more important to the agent designer than to the agent. This is analogous to running your word processing software on a PC running Windows or on a Sun workstation running Unix. Although there will be performance issues, the behavior of the two software systems will be similar. It is more important to take the bugs out of the software than argue the platform choice. The exception will be architectures which are so complex that they cannot operate in anything approaching real time in a single processor architecture. However, our observation is that even simple, single board processors (such as a Motorola 68HC11) allow for amazingly complex behaviors. We need to understand these behaviors before we worry about implementation on parallel processors. If anyone can justify a three level architecture we can come up with a five level architecture, adding one level between each of the existing two levels and then justify them using a different granularity level.

Performance -- What types of performance goals and metrics can realistically be used for agents operating in dynamic, uncertain, and even actively hostile

environments? How can an architecture make guarantees about its performance with respect to the time-critical aspect of the agent's physical environment? What are the performance criteria for deciding what activities take place in each level/ component of the architecture?

We have formally defined and used a systems of metrics to determine the performance of an agent. The metrics are composed of three performance measures: the quality of the tasks done, the energy consumption of the agent, and the total time taken by the agent in completing the task. These measures are referred to as quality, efficiency, and timeliness in the human work-group studies. If only one performance value is desired, the three values can be combined using three Lagrangian multipliers to change the three values into similar units (scalars), so they can be mathematically combined. We dealt with these issues in the tropism system-based agents.

Psychology -- Why should we build agents that mimic anthropomorphic functionalities? How far can/ should we draw metaphoric similarities to human/ animal psychology? How much should memory organization depend on human/animal psychology?

We do not need to build agents that are anthropomorphic unless we need the agent to do very human like tasks. Building agents with inspiration from biology is an excellent idea since nature has provided us with great working examples. It is not clear that at this point we need to go as far as imitating humans, specially since we have trouble mimicking much simpler life forms. Certain concepts from human psychology can be very helpful. The tropism architecture is based on the likes and dislikes of the agents. Such ideas of seeking pleasure and avoiding pain have been around for some time in psychology, and we have shown that they can be utilized in the synthesis of software architectures for agents.

Simulation -- What, if any, role can advanced simulation technology play in developing and verifying modules and/or systems? Can we have standard virtual components/test environments that everybody trusts and can play a role in comparing systems to each other? How far can development of modules profitably proceed before they should be grounded in a working system? How is the architecture affected by its expected environment and its actual embodiment?

Simulation is a very useful tool in studying agents. In our own work we used simulation to study large colonies, of up to 100 agents. After building a physical colony of four real robots to test the tropism architecture, we can only imagine the difficulty in building and experimentation using a colony of 100 robots. Simulation allows the conduct of thought experiments, which might be difficult or impossible to conduct with physical implementations. Further, scaling to larger numbers of agents or more complex architectures is easily accomplished in simulation.

On the other hand, one must always remember that the simulated agent is an abstraction from the real world, so that a translation of behaviors from simulation to reality is not always possible. This is one of the reasons why Brooks argues for agents which are implemented and situated in the world. For a discussion on the problems of relating simulation and reality, see [Bekey, 1977].

Learning -- How can a given architecture support learning? How can knowledge and skills be moved between different layers of an agent architecture?

An architecture can support learning when its structure and /or parameters can be automatically modified as a result of experience. The tropism system supports both phylogenetic and ontogenetic learning. Simulated evolution was used to evolve colonies of robots which were more fit (i.e., which displayed higher values of the performance metrics), while individual robots were able to learn from the world. We implemented learning from perception of novel situations, learning from success, and learning from failure. It was shown that the performance metrics of the agents improved through learning. How the knowledge moves between the layers of the architecture is simply a design choice and is not that significant in the overall survival and performance of the agent, as it is very much implementation-dependent.