CS 664 – Graduate Seminar

Term Paper

Winter'04

John Leung

jkleung@csupomona.edu

Autonomous Agents and Complex Adaptive Systems

Abstract:

Autonomous agents are programming constructs governed by rules to perform tasks within an environment. Some agents are complex while some are simple. In this paper, I will describe both types in two different models. These agents may seem "unintelligent" and can not accomplish much alone. However, when interact with one another they can accomplish tasks that one may not think possible. This paper will emphasize on different theories and ideas on these agents and their environments, and the different types of tasks and applications they can accomplish.

1. Introduction

Today most computer programs and applications are written to solve specific problems. They are created with specific goals in mind and requirements to follow. They serve well under most circumstances, but are not suitable solutions for problems that require changes and the ability to adapt to them. Some examples of these problems are robotics control systems, forecasters such as weather and economic forecasting, and others. Solving these problems with static pre-programmed solutions requires much labor to update them once new changes to their environment occur, not to mention the time, money, and danger (if human lives are involved) associated with these changes as well. What we need are ways in which these systems can adjust to these changes by adapting to them automatically.

Some of the well known methods that address to environment changes are Genetic Algorithms, Evolutionary Strategies, and Neural Networks. These methods have

their advantages and disadvantages, but they are beyond the scope of this paper. What this paper does is to introduce two new methods called Complex Adaptive Systems (CAS) and Swarm Intelligence. CAS, Genetic Algorithms, and Evolutionary Strategies fall under the same category known as Evolutionary Computation (or Approaches), but they are a bit different; and it is uncertain exactly what category Swarm Intelligence fall under for now. However, this paper will only focus on CAS and Swarm Intelligence. In fact, I will introduce them without requiring the reader to have any prior knowledge of any other evolutionary computation methods.

2. CAS History and Background Information

Genetic algorithm was invented by John Holland in the 1960s. In contrast to Evolutionary Strategies also introduced in the same time period by other people, genetic algorithms is designed to study the phenomenon of adaptation as it occurs in genetics and to port over those ideas into computer systems. Holland then looked at adaptation from a totally different perspective in the early 1990s. Instead of looking at how one particular population interacts within an environment such as neurons within a central nervous system or genes within bodies; he looked at all such entities and see if there is a common model in which they all interact within their environment. In reducing these entities to their basic forms, he sees that they all behave in a similar fashion: antibodies interact in immune system, buyers and sellers commerce in the stock market, companies that play a role in the economy, etc. These entire phenomenons can be broken down to members (agents) within their environment working together to achieve their goals, which forms what he calls it Complex Adaptive Systems (CAS).

2.1 Agents

Since agents are the main role players in these systems, we should first take a look at how they are modeled. The word *autonomous* is often added in front of *agent* because they respond automatically to stimulus or messages given. As mentioned before, CAS composes of agents and their interactions within the environment. The key word here is interaction. Interaction is the main driving force behind adaptation and especially so in Swarm Intelligence (which I will introduce later). The best way to model these interactions are post production rules. That is:

IF receive *stimulus (condition)* message THEN post *response* message

Agents are composed of these post production rules. A chain of rules may be activated to produce the final response for the agent. The general form of the agent's characteristics is as follows:

IF receive *initial stimulus (condition)* THEN post *final response* message

For example, the following could be a set of rules for a frog agent:

Rule1: IF (there is a moving object) THEN (flee)

Rule2: IF (flee) THEN (move legs)

Rule3: IF (flee) THEN (urgent)

Rule4: IF (move legs, urgent) THEN (move legs fast)

Agent: IF (there is a moving object) THEN (move legs fast)

Since the final response of the agent is the posting of a new message, this construction easily facilitates interactions between agents within the environment because that new message can in turn be the stimulus for another agent.

2.2 Messages and Rules

Messages are representation of the stimulus that the agent sees, and is presented by a string of length L, composed of 0's & 1's. The conditions of the agents' rules consist of the same length L of 0's, 1's, & #'s. Where #'s are don't cares. For example, the following could be messages and their encodings in a frog agent environment:

There is a moving object -> 1000000000

There is a smaller moving object -> 1100000000

And these could be the rules for the frog agent:

IF (there is a moving object -> 1#########)

THEN (flee -> 0000000010)

IF (there is a smaller moving object -> 11########)

THEN (approach -> 0000000001)

The less #'s in the conditions, the more specific the rules are. In the above example, the second rule is more specific than the first because it has less #'s. Thus, if there is a small moving object, only the second rule will activate. The rules with more #'s can be thought of as the "default" rules; ones that get activated if nothing else can be. If the default rules can be activated for both cases, then why do we need the more specific one (and thus save ourselves some storage space too)? Well, it is best to let itself decide if the default rules are satisfactory enough. For a frog agent, if it always flees when it sees a moving object, it will eventually starve to death. In effect, the "default" rules can sometimes do more harm than good. But what happens if there is a moving object, will both rules be activated?

The solution is strength association. Each rule for the agent is associated with a strength value also. The activation of the rules is actually determined by the product of the specificity and the strength value. In the above example, the specificity of the first rule is 0.1 since only 10% of the condition is specified; likewise, 0.2 for the second rule. If there are two or more equally specific rules (e.g. having both "a smaller" and "a weaker" moving object, both having the specificity of 0.2) that satisfy the condition then the strength value will decide which rule to activate. Another reason for the need of strength value is that sometimes it may more beneficial to have the "default" rule be activated. In our example, the frog may live in a very hostile environment where some of the smaller moving object is poisonous. After adapting to this environment, the rule for the smaller moving object may get a much smaller strength value than the default rule. And thus, the frog may want to flee even if it sees a smaller moving object and only approaches if it sees a "weaker" object instead.

One thought I was pondering over is what if two or more of the rules with the same specificity and strength product value are satisfied, which rule will be activated? Since they all have the same product value, they should all theoretically have the same chance to benefit the agent. A simple solution is if the same product value is a result from different specificities and strengths (e.g. 0.1 * 20 and 0.2 * 10, both = 2.0), then give more power to either specificity or strength, depending on the type of application the CAS is intend for. However, if the same product value is a result of the same specificity and strength (e.g. both are .1 * 20), then this method will not work. The solution I came up with is to associate the rules with a time stamp and a storage space for the latest rewarded value (to be explained later, think of it as the most recent positive or negative

value added to the strength). The one that was previously rewarded most amongst them will get to be activated. If two of more of rules was previously rewarded with the same highest value, then the one with the latest time stamp will be activated. This latest time stamp will ensure that the agent will adapt to the latest changes in the environment.

2.3 Adaptation

Strength association is not only a means to solve ambiguity of rule activation, but also a key idea behind adaptation. If a set of rules produces a beneficial result for the agent, then that set of rules must be remembered for it. Otherwise, the next time the situation arises; other worst performing rules may be activated instead. The mechanism for "remembering" this is simply to reward the set of rules by increasing their strength values (and likewise, decreasing their strength values for harmful results). This way, the set of rules that gets to activate are guaranteed to be the most beneficial for the agent. Notice that the word *set* is always used along side with the word rules above. The reason why we need to award the entire set of rules is that sometimes, some of the rules within the chain of reasoning will do temporary harm to the agent. One example of this if CAS is used to implement a poker game player, then the agents may be the different strategies involved. Some of the rules for a particular strategy can be a bluff move which can cause a temporary setback but will be beneficial to the agent at the end. Thus, all rules involved for the final outcome must be rewarded or punished accordingly.

Thus far, we have only dealt with adaptation in a very light way. Strength rewarding by itself only deals with adapting to a constant environment. What happens when the environment changes? This is the question that must be answered in order to

have a true adaptive system. To do this, we must have new rules to handle the new environment. The generation of new rules is done in CAS using genetic algorithms. Since genetic algorithm is such a vast topic by itself, I will only briefly describe it.

In genetics, chromosomes are represented by a string of 1's and 0's; and a fitness value is associated with each chromosome. This fitness value basically determines the chromosome's chance and ability to produce surviving offspring. Because of such close resemblance in characteristics and representation between chromosomes & fitness in genetics and messages/conditions & strength in CAS, CAS is able to use genetic algorithms for its mechanism for adaptation. The following is a set of my simplified understanding of genetic algorithm terms that may be helpful in understanding Holland's method of applying genetic algorithms to CAS (further below):

Pairing – Selection of parents for reproduction

Mutation – Random change of one or more symbols to other legal symbols in the alphabet within one chosen string

Crossover – Reproduction based on different breeds (in CAS, it would be rules that may not have a directly connection)

Holland's method of new rule generation in CAS:

1. Reproduction according to fitness. Select strings from the current population (this might be the set of rules for the agent) to act as parents. The more fit the string (the stronger the rule), the more likely it is to be chosen as a parent. A given string of high fitness may be a parent several times over.
2. Recombination. The parent strings are paired, crossed, and mutated to produced offspring strings.
3. Replacement. The offspring strings replace randomly chosen strings in the current population. This cycle is repeated over and over to produce a succession of generations. [Holland, p.70]

This selective rule generation based on strength value is a very important concept. If we are not careful in the selection of the rule generation process, then not only will most of the computation time be wasted for generating useless rules, but also the number of rules will get so huge that by the time it finds the best rule to activate, it will be too late already. This is especially important in a real time system where every microsecond counts. For example, if CAS is implemented in our Mars Land Rover, and if the on board cameras are not positioned well, then it may only have a second or so before reacting to a crack on the ground. If too many rules exist, then the set of rules to stop or detour may not be activated in time before it falls off the crack, thereby making a unforgettable multi-million dollar mistake.

2.4 Thoughts

If we look back at the IF THEN construction of the rules for agents, one may find this to be similar to rules in expert systems. In a way, rules in an expert system can be viewed as agents, and the dependencies in which one rule causes another to fire can be thought of as interactions between them. However, the difference is that expert systems are very static in the way they are constructed today. In fact, ever since expert systems gained popularity in the 1980s, they have only been shown to be useful in very few applications. One of the main reasons contribute to this is that in order for the expert systems to be up to date and be useful, it must adapt to new changes. Thus, they have only shown usefulness in applications where minor changes are required. Well, if both expert systems and CAS use the same basic underlying structure, then why couldn't we

just apply the adaptation methods described previously from CAS into expert systems? I came up with two plausible reasons for this.

First, most of time in spent in developing a typical expert system is knowledge engineering. Knowledge engineering usually involves knowledge engineers meeting with experts in their fields to gather expertise. This process is so manual labor intensive that manual updating of the code is nothing compare to it. Second, most of the expert systems are really made of knowledge from human expertise and opinions, it may not be realistically possible for agents to explore and learn such expertise and opinions.

A second thought I had was that the length of the messages and conditions may be "predefined" from my understanding of Holland's explanation. If this is the case, it will be hard for the agents to truly adapt to new environments which the developers did not anticipate. Since this sounds pretty absurd, the predefined length could be just my misunderstanding or it was just a simplified explanation of how the rules and agents work. It is very possible that the actual implementation of the length is dynamic, and agents learn to adapt to the newly found messages by reproducing rules to handle them using genetic algorithms. If the newly reproduced rules failed to produce good results, it will get punished. Soon, the agents will reproduce new rules which will truly adapt to the new environment.

3. Swarm Intelligence

Swarm Intelligence is a new field of study in complex systems. It was introduced in the mid 1990s. It uses some key ideas from CAS and expands further in them such as agent interaction. In fact, agent interaction and collaboration is so important here that

these agents are implemented with normal programming constructs; the basic and simple form of agent construction in CAS is not sufficient. Because of this, swarm intelligence is really not considered as CAS but a different set of complex systems. In fact, because of its complex form of agent representation, it is difficult to use genetic algorithms or other learning systems for it adapt. For this reason and the fact that swarm agents are difficult to program, the only field in which swarm intelligence has shown real promise is robotics.

Swarm intelligence is really a modeling system of interactions of organisms in a few particular colonies of (mainly) insects, such as ants, bees, wasps, and termites. The interesting thing here from observations of these colonies is that "every single insect in a social insect colony seems to have its own agenda, and yet an insect colony looks so organized." [Bonabeau et. al., p.1] This phenomenon is known as self-organization. In a sense, this self-organization makes up for its lack of true adaptation.

To elaborate more on the interaction and collaboration of such agents and self-organization, let us look at the ant colony first. Ants communicate with one another either directly or indirectly. Direct communication is simply direct contact, visual contact, chemical contact, or other means. Indirect communication usually takes form by "one modifying the environment and the other responding to the new environment." [Bonabeau et. al., p.14] For example, when one ant deposits an item such as dead bodies, larvae, etc. randomly (thereby changing the environment), other ants will start piling those items in the same location (responding to the new environment).

Another example is when an ant finds food; it leaves a trail for others by leaving a special scent on the trail. When another ant picks up this scent, it will follow the trail and

leaves more trail for others if there is more food. But what if the route the original ant took is an ineffective long route, will all the other ants follow? One interesting observation of this trail following is that not all ants are created equal. They each actually have their own agenda. Eventually, one of the ants will randomly select a different path. If that new path is shorter, the scents will become stronger than the longer path since the ants that have traveled that new path can traverse it much more often (because the path is shorter); and thereby leaving more scent on the new trail than the other. The stronger the scent, the more chance new ants will follow it.

3.1 Applications

The modeling of the trail following observation gave researchers insights to use swarm intelligence for finding solutions to problems such as the Traveling Salesman Problems and others. The item piling observation previously described can be used for clustering and sorting problems. The division of labor in ants and other colonies phenomenon can guide us in finding solutions to optimization and resource allocation problems. However, as mentioned before, the only real visible field in which swarm intelligence has shown real promise is robotics. The following diagrams best describe this:

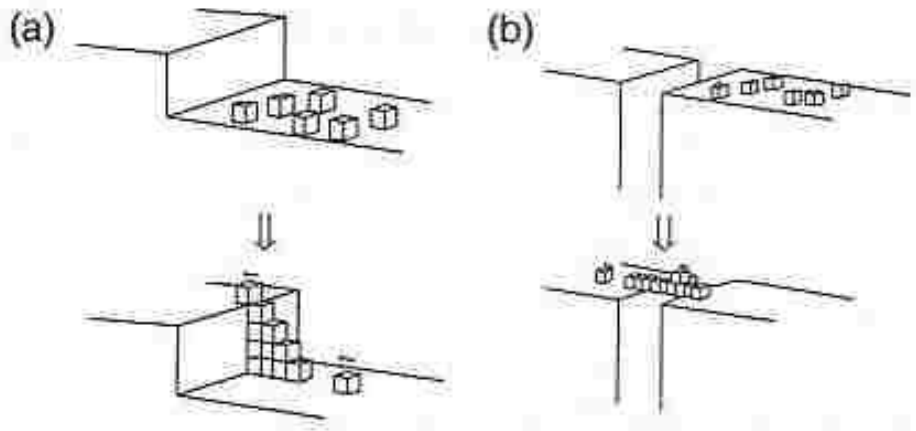

Figure 1. Theoretic robot collaboration to climb stairs and cross gap [Bonabeau et. al., p.244]

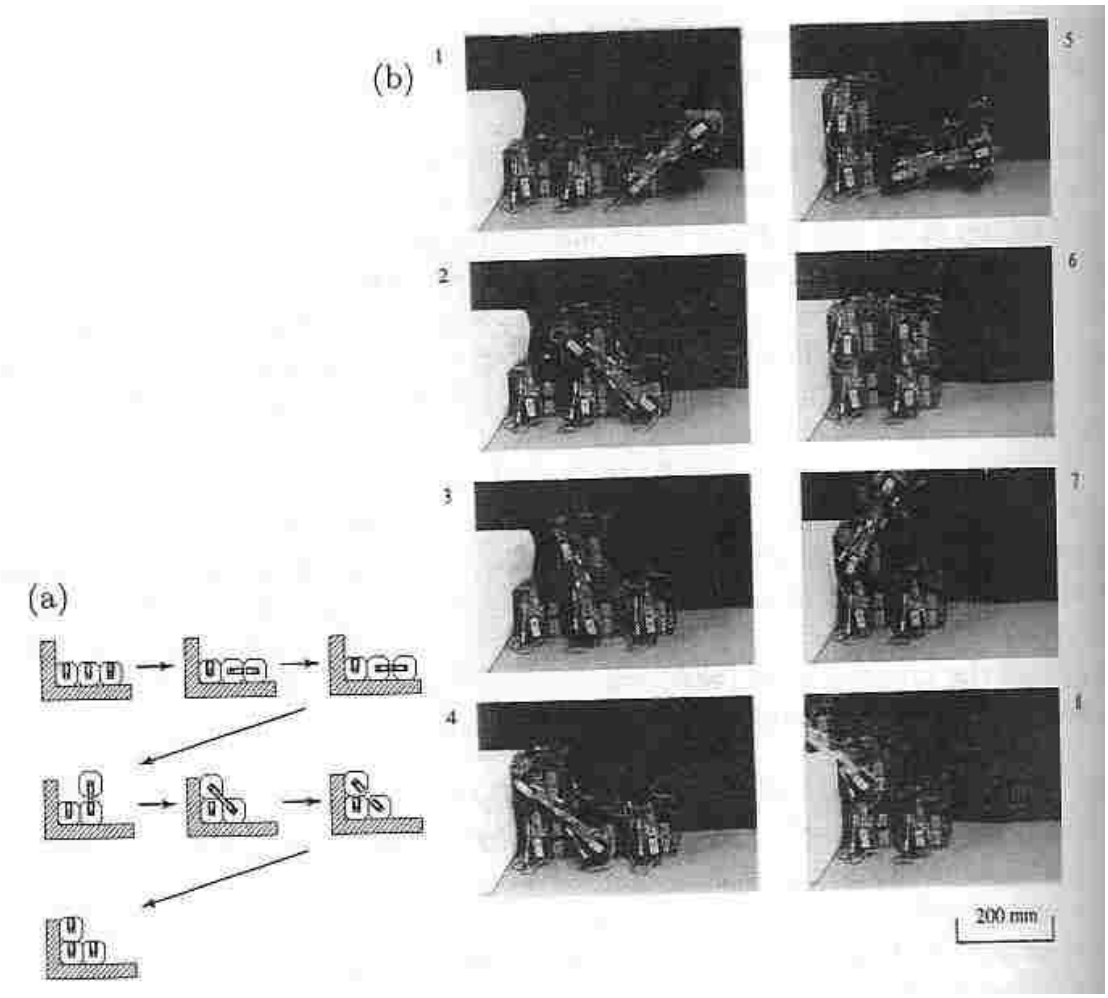

Figure 2. Theoretic and applied robot collaboration to climb stairs [Bonabeau et. al., p.246]

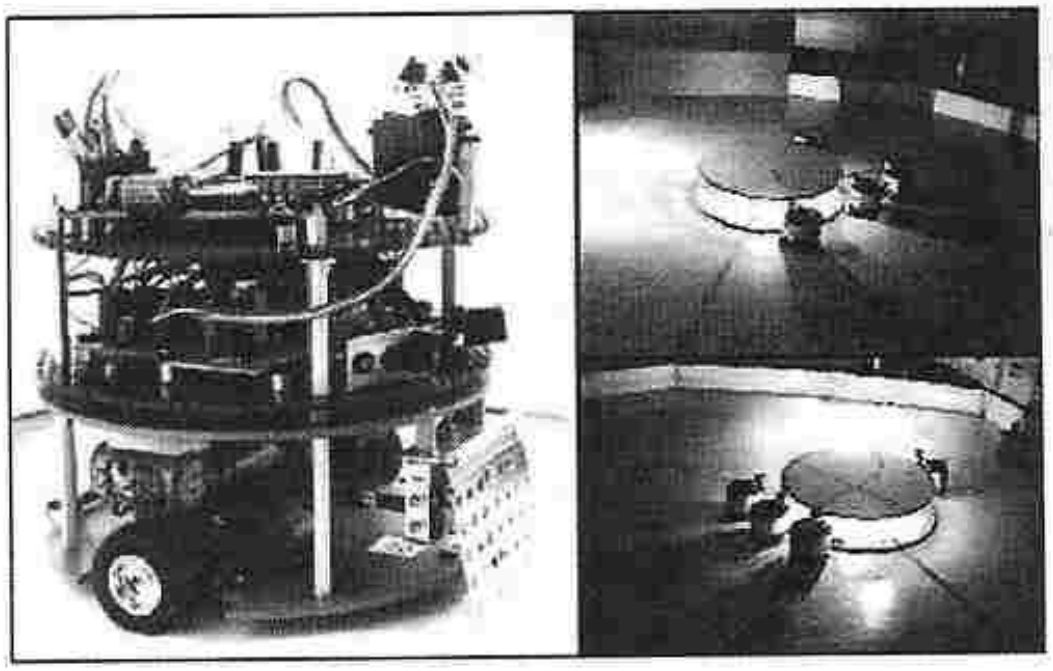

Figure 3. Applied robot collaboration to move object [Bonabeau et. al., p.269]

Collaboration in robotics is useful since in most cases, it is better to develop a set of robots to work together than a single robot to do so. The reason is that first, once an initial simple robot is developed, it can be cheap, fast, and efficient to produce the others based on it. Second, it is often difficult or impossible to make a single robot developed to handle certain task to handle other new tasks. This is not the case for a group of smaller and simpler robots (figure 1 and 2 is useful in seeing why). Third, it is safer and more reliable since if one of the robots fails, it usually does not jeopardize the mission; whereas if a single robot fails, the whole mission has to be abandoned. Finally, smaller robots can get access to areas of small entrance such as collapsed buildings after an earthquake. The robots can later join together to push away obstacles (example shown in figure 3).

4. Conclusion

Swarm intelligence has actually gained more popularity even though it is both newer and its applicable fields are narrower than CAS. I believe this is mainly due to the fact that people can actually "see" it in action. But as mentioned previously, it can also be used to solve theoretically problems. CAS on the other hand has actually made a bit more progress in terms of applicable fields. Holland has built a system called *Echo* for general purpose simulation and modeling. It has been shown to work as the backend system for a flight simulator; although simulation, forecasting, and modeling of ecology, weather, economy, others should also be possible. In terms of theory, it has been shown to give solutions to the Iterative Prisoner's Dilemma. In sum, both CAS and swarm intelligence is still in their early research stage to predict what they are really capable of doing. However, thus far I would say their future looks promising.

References:

Holland, J. H. 1995. *Hidden Order.* Perseus Books, Cambridge, Massachusetts.

Bonabeau, E., Dorigo, M. and Theraulaz, G. 1999. *Swarm Intelligence.* Oxford University Press, New York, New York.

Flake, G. W. 2001. *The Computational Beauty of Nature.* The MIT Press, London, England.

Melanie Mitchell. 1999. *An Introduction to Genetic Algorithms.* The MIT Press, London, England.

Robert Axelord. 1997. *The complexity of Cooperation.* Princeton University Press, Princeton, New Jersey

Gerhard Weiss. 1999. *Multiagent Systems.* The MIT Press, London, England.