

Evaluating Different Genetic Operators in the Testing for Unwanted Emergent Behavior using Evolutionary Learning of Behavior

Jörg Denzinger, Jordan Kidney
Department of Computer Science
University of Calgary, Calgary, Canada
{denzinge,kidney}@cpsc.ucalgary.ca

Abstract

We present an experimental comparison of different genetic operators regarding their use in an evolutionary learning method that searches for unwanted emergent behavior in a multi-agent system. The idea of the learning method is to evolve cooperative behavior of a group of so-called attack agents that act in the same environment as the tested agents. The attack agents use action sequences as agent architecture and the quality of a group of such agents is measured by how near their behavior brings the tested agents to show the unwanted behavior. Our experiments within the ARES II rescue simulator with an agent team written by students show that this method is able to find unwanted emergent behavior of the agents. They also show that rather standard genetic operators (on the team level and the agent level) are already sufficient to find this unwanted behavior.

1 Introduction

One of the hot research topics in multi-agent systems (MAS) is achieving emergent behavior of a group of agents. The form of emergent behavior researchers are most interested in is the creation of synergetic effects that have as result that the group of agents achieves goals that are well beyond what can be expected from them if we just add up the individual abilities of these agents. But unfortunately, emergent behavior can also result in a system behavior that is not wanted by the developers of the system.

So far, very few researchers have looked into the problem of unwanted emergent behavior. Detecting unwanted behavior of a system in general is usually the task of the system test phase in the development, but software testing as a discipline also has not looked intensively into this problem. There the best practice is the use of randomly generated interaction sequences to see if something bad is happening

(see [4]). The detection of unwanted emergent behavior of a system, which can also be caused by adaptive abilities of a single agent, still is the sole responsibility of human testers, with very limited tool support (see [6], [9]).

In [3], we presented an approach to support testing for unwanted emergent behavior that fights "fire with fire" by using techniques from learning of cooperative behavior to search for a particular unwanted behavior. The general idea of this approach is to have a group of tester (or attack) agents act as users and other systems that a group of agents, the tested agents, interact with. We then use learning techniques for the attack agents to have them produce interaction sequences with the tested agents that come nearer and nearer to showing the particular behavior that is unwanted. In [3], we used as learning method an evolutionary learning method that uses sequences of actions as individuals, one sequence for each attack agent.

[3] presented a proof of concept application that tested multi-agent systems developed by students of a basic MAS class. While fitness functions for two rather different behaviors were presented, other parameters of the evolutionary learning method, like genetic operators or population size, and their influence on the results were not evaluated. There have been many studies on the influence of such parameters on the performance of evolutionary algorithms in general (see, for example, [13] or [7]), but the gist of these studies is that it depends on the particular application what setting of these parameters offers a chance for a good performance, with the usual caveat that there still might be some problem instances for which this setting might not be good.

Since we used a class of genetic operators, namely so-called targeted operators, that were specially developed for action sequences, in this paper we examine and analyze different genetic operators and their combinations with regards to their influence on finding unwanted emergent behavior. Our experiments with simulated rescue teams written by students show that quite a few of the genetic operators allow the system to find a particular unwanted behavior (the students' agents freeze at some point in the simulation). But in-

terestingly, the combination of rather standard operators on the level of the agent team and on the level of a single agent was consistently the best choice in our experiments both with regard to quality of the evolved attack team and the number of generations needed to produce the attack team.

2 Agents and multi-agent systems

Since we provide a rather general framework for testing multi-agent systems we use a rather general definition of agent (although our attack agents will naturally be a very specific instantiation of the following definition). An agent Ag is a quadruple $Ag = (Sit, Act, Dat, f_{Ag})$. Sit is a set of situations that the agent can find itself in, Act is the set of actions that Ag can perform and Dat is the set of possible values that Ag 's internal data areas can have. When having to determine its next action, Ag uses $f_{Ag} : Sit \times Dat \rightarrow Act$ applied to the current situation and the current values of its internal data areas to make this decision.

Obviously, a multi-agent system requires a set of agents A . But these agents need to share an environment (or at least parts of it) in order to interact with each other, and this environment Env has quite some impact on what the agents can do. So, formally a multi-agent system MAS is a pair $MAS = (A, Env)$. Actions of the agents might change the environment (or it can change on its own) and therefore Env should consist of a set of environment states. Going back to our agent definition, a situation in an agent's set Sit might contain a view on the current environment state, information about the agent itself that has not found its way into a data area in Dat , and information about the other agents based on the perception of the agent.

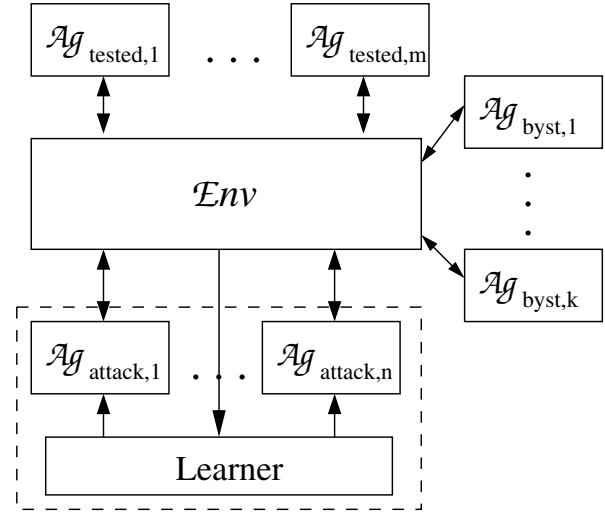
3 Using learning of behavior to test for emergent behavior

Our general idea for testing a multi-agent system $MAS_{tested} = (A_{tested}, Env)$ for unwanted emergent behavior is to use a set of so-called attack agents¹ A_{attack} that play the role of entities that the agents in A_{tested} usually interact with in Env . Such entities can be some of the team mates of the agents in A_{tested} , human users interacting with MAS_{tested} or other systems that interact with MAS_{tested} . In our general setting, we also allow for the existence of a third set A_{byst} of agents, the bystander agents, which are agents that are neither in A_{tested} nor under control of the tester, but interact with at least some agents in A_{tested} in the environment.

Figure 1 presents a graphical view of the general setting. Key for our general approach is that the behavior of

¹We use the term attack in the following, since "tester" and "tested" are very difficult to distinguish and what the attack agents are doing can definitely be seen as an attack on the tested agents.

Figure 1. General setting of our approach



the agents in A_{attack} is the product of learning that aims at figuring out what interactions of these agents with the environment, the tested agents and the bystander agents may produce (or at least comes near) a certain behavior by the tested agents. While Figure 1 suggests that this learning is performed by a central learner that feeds behaviors into the attack agents (which is the approach we have taken in Section 4), we can also envision that the learning takes place inside of the agents of A_{attack} , which then obviously requires concepts for coordinating the learning.

More formally, our general approach is as follows: If we look at the setting in Figure 1 as an outside observer, then the interaction of an agent $Ag_{attack,i} \in A_{attack}$ with the other agents can be described as a sequence of actions $a_{i,1}, \dots, a_{i,l_i}$ ($a_{i,j} \in Act_{attack,i}$). The action sequences of all the agents in A_{attack} then result in a sequence e_0, e_1, \dots, e_x of (enhanced) environment states. An (enhanced) environment state is a view on $Env \times \bigcup_{Ag \in A_{tested}} Dat_{Ag}$, which means that we allow the observer (which could be the learner) to see some aspects of the environment (hence our use of the term "view") and the content of some or all the internal data areas of the tested agents. An unwanted emergent behavior of the tested agents can then be characterized by defining a condition \mathcal{G}_{emerge} on a set of (enhanced) environment states, i.e. $\mathcal{G}_{emerge}((e_0, e_1, \dots, e_x)) = \text{true}$.

One of the distinctions we can make regarding multi-agent systems is whether they interact with the environment and each other in a synchronous fashion or in an asynchronous fashion. If interaction takes place synchronously, then the length of the action sequences of the attack agents is always the same and by going through the action sequences simultaneously, we also create a sequence of environment states of this length (formally: $l_i = l_j$ and $l_i = x$ for all i, j). Allowing for asynchronous interac-

