Development Report: Incorporating a Genetic Algorithm Framework into a Multi-agent Modeling System

Jeffrey Fuller and Greg Wolffe
fullerje@student.gvsu.edu, wolffe@gvsu.edu
Dept. of Computer Science
Grand Valley State University
Allendale, MI 49401

Abstract

This project involved the incorporation of varying levels of intelligence into a Swarm-like simulation system. In particular, it entailed the incorporation of a genetic algorithm framework into the RePast agent-modeling tool. We describe the details of our implementation, present some preliminary results, and indicate potential directions for future research.

Introduction

In order to better understand, and possibly enhance, agent-modeling, we began a project to merge the capabilities of a genetic algorithms package into an existing simulation toolkit.

Multi-agent Modeling

RePast (REcursive Porous Agent Simulation Toolkit) is a Swarm-like software framework written in Java and used to create agent-based simulations. It consists of a rich library of classes designed to facilitate the modeling of flexible agents imbued with social characteristics.

Genetic Algorithms

JGAP (Java Genetic Algorithms Package) is a component used for developing an evolutionary computing approach to problem solving. It includes a set of classes providing genetic operators (reproduction, crossover, mutation) and the infrastructure necessary to exploit the principle of natural selection as an optimization mechanism.

Description

RePast and JGAP are both written in Java. They are frameworks with a clean, modular design that simplified the task of merging the functionality of the two packages. However, the concept of merging presented several design alternatives as to what level the genetic algorithm (GA) should be incorporated.

Agent level

When the genetic algorithm is incorporated at the agent level, the possible behaviors of the agent function as genes. Each agent runs a genetic algorithm that selects the optimal behavior for that individual agent. A specific agent's behavior "profiles" are mutated and exchanged in an effort to produce a behavior that is more fit within the circumstances of that agent.

Model level

Using a GA at the model level implies that agents should be considered genes. The model runs a genetic algorithm to select, or favor, the optimal agent within the model. Agents exchange behaviors during reproduction, or pass mutated behaviors on to their offspring. The fittest agents, original or offspring, persist and flourish throughout the duration of the simulation.

Population of models level

At the top level, an entire simulation is considered a gene. The genetic algorithm is used to choose the optimal model from among a population of simulations. Simulation models exchange agents with each other to create new hybrid models or mutate agents to create modified models. A global model fitness measure is used to determine which simulations persist over time.

Results and Discussion

To date we have completed merging the two frameworks at the model level. In order to incorporate evolutionary behaviors into a simulation, the user should:

Create an agent, which extends the Chromosome class. All genetically mutable behaviors should be created as genes within an array of genes in the agent. The agent must override the clone(), and compareTo() methods of the Chromosome class to include any relevant information in addition to the gene array.

The next step is to implement a fitness function. This is done by extending the fitness function class, and overriding the evaluate() function.

Within the model, instantiate a new configuration object and fitness function object. To initialize the configuration object, create a sample chromosome containing the set of behaviors defined by the gene array designed in the agents. Then initialize the population, encode the population into an array of chromosomes, and create a genotype object using the encoded population and the configuration object. Finally, add a scheduler action to the model that calls the evolve() function of the genotype.

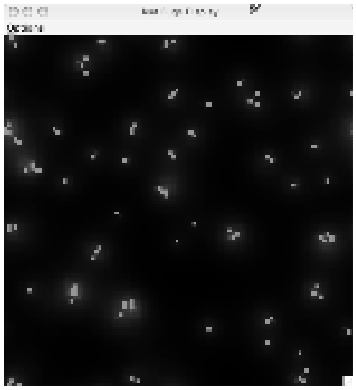
Example

We used the above methodology to add natural selection to the Heat Bug model included as an example with the RePast toolkit. The model consists of multiple agents (heat bugs) that each seek their ideal temperature. Each bug emits heat that contributes to the

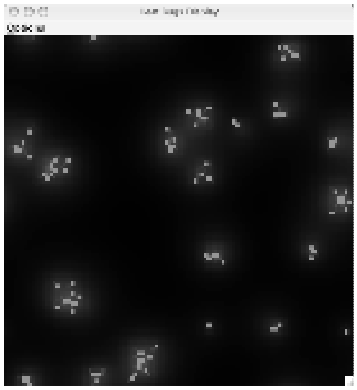
ambient temperature at the discrete locations within the simulation environment. Since most bugs' ideal temperature is higher than the ambient temperature, they seek proximity with other bugs and tend to "cluster" together for warmth.

The figure below shows the evolving state of the two systems over time (i.e. "ticks" of the simulation). The group of images labeled "RePast" is the original model included with the toolkit. The images labeled "GA enhanced" show the effect of incorporating a genetic algorithm into the model. It appears the genetic algorithm served to amplify the behavior observed in the original model.

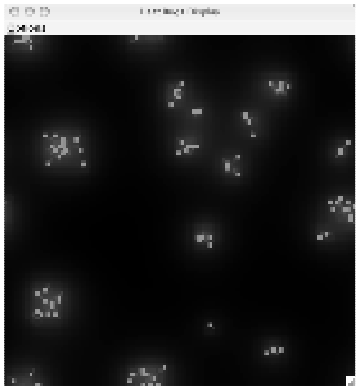
Heat Bug Model



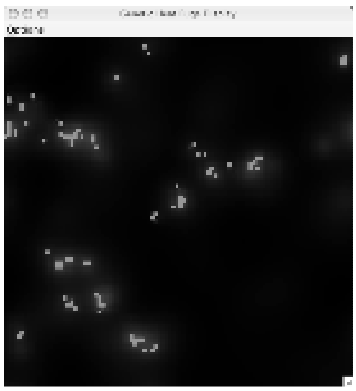
RePast – time 100



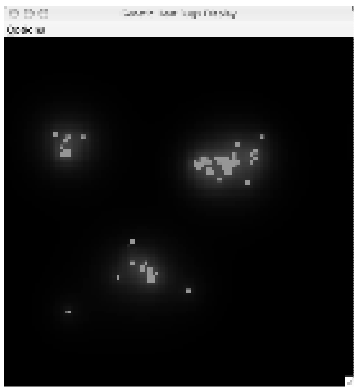
RePast – time 1000



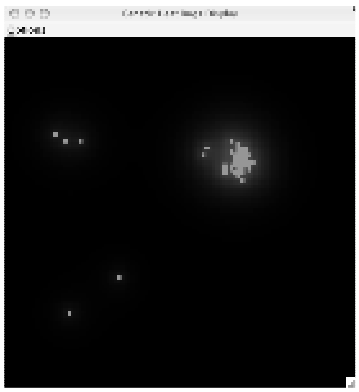
RePast – time 2000



GA enhanced – time 100



GA enhanced – time 1000



GA enhanced – time 2000

Summary

We are currently implementing the remaining two levels of integration. We are also performing additional tests in order to understand the functioning, the possible capabilities, and the appropriate use of our integrated frameworks.

We are very interested in presenting this work in a venue (poster, workshop) that will allow us to obtain feedback on our design decisions and interpretation.