

ALES-DEVS

An Ecosystem Simulator built with DEVSTJava

John Daigle
johnpdaigle@gmail.com

Arash Akhlaghi
Aarak_73@yahoo.com

ABSTRACT

Ecosystem modeling is traditionally done using flexible, agent based models, or by using ordinary differential equations to estimate behavior over large populations. Both of these approaches have positive and negative aspects. The DEVST modeling system is a discrete event modeler that may be able to combine the positive aspects of discrete event simulation with the ability to run giant simulations that is typically only found in highly abstract systems using ODE. In this paper, we describe a simple ecosystem simulator designed in DEVSTJava.

KEYWORDS

Modeling and Simulation, Agents

1 INTRODUCTION

There are essentially two reasons why computer science should care about flexible, adaptable, naturalistic systems from an algorithmic standpoint. The first is that humanity exists in a sea of such systems, and our behavior affects them in ways that are not obvious. The second is that complex adaptive systems can help us to solve problems of interest to computer scientists.

A complex adaptive system is one in which a large number of autonomous agents each attempt to solve, with a variety of strategies, purely local problems with a minimum of strategic communication with other agents. The idea is to structure systems such that local optimality leads to global

optimality. Complex adaptive systems are often based on metaphors taken from nature, bees, ants, and squirrels have all been used as the basis for complex adaptive behavior.[2]

1.1 Natural Systems

Open questions that could severely impact our world, for example, include such seeming trivialities as the decline in the population of the North American Honeybee,[6] which could result in the extinction of some fruits and vegetables, to the long term effects of carbon emissions, which could destroy civilization as we know it. Both of these questions can only be properly understood or analyzed in terms of a complex adaptive systems.

Even comparatively trivial tasks, such as developing a working ecosystem model that merely appears realistic, are difficult.[3] In order to accomplish the appearance of reality, an algorithm designer must account for terrain, plant characteristics, competition, cooperation, resource allocation, and more. To actually use a climate or ecosystem model to further our understanding of the consequences of human action or natural disaster is orders of magnitude more difficult.[1]

In addition to modeling large scale systems such as the climate, there is a great deal of interest in modeling microscopic systems, such as the human immune system or biological cells. A great difficulty in modeling such systems is dealing with the sheer volume of agents or entities to be modeled. The human immune system, for example, consists of approximately 10^{12} cells in 20 species, engaged in a complex computational task. Each time the immune system encounters a novel pathogen, it has to solve an NP-Complete problem, finding the correct anti-bodies for the intruder, while its infrastructure is being destroyed.[5] If successful, that solution will be saved to be used if that intruder shows up again. Over the lifetime of a human being, we develop a large number of defenses against various common pathogens, although our immunological response begins to taper off later in life. This is a bit of a mystery, and certainly an area of interesting research.

Other compelling reasons to investigate the functioning of the human immune system are cancer research, AIDS/HIV research, autoimmune diseases from Arthritis to MS, and our desire to defeat the common cold. A compelling and accurate model of the immune system, however, is a difficult

©2007, John Daigle, Arash Akhlaghi. This work is licensed under the Creative Commons Attribution-NonCommercial-Share Alike 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.

goal to reach.

1.2 Solving Problems

The second reason we should be interested in learning about complex adaptive systems is in order to solve difficult problems. Resource allocation problems and traveling salesman problems, for example, have efficient solutions that can be developed with swarm intelligent, complex adaptive systems. The advantage of using such algorithms is that they are highly scalable and appropriate for a wide range of inputs.

1.2.1 Squirrels

“Squirrels” is a complex adaptive system for resource allocation based on the hoarding behavior of squirrels.[2] The author’s had three essential evaluative criteria for any CAS, flexibility, robustness, and self-organization. It is not a criteria of any CAS algorithm that the solution be correct, that is, they are usually deployed as highly efficient heuristics rather than guaranteed correct algorithms. This makes CAS highly useful in domains or when approaching solutions that have no efficient solution.

In the case of Squirrels, the problem to be solved is resource allocation in a peer-to-peer network. The squirrel metaphor treats resources as nuts to be hoarded by each particular squirrel, based on a limited number of signs from other squirrels indicating where they have buried their own nuts. Using this, the authors were able to get good data allocation diversity.

The aspects of real life squirrel behavior that the authors wished to capture was

- random gathering and burying of acorn stashes.
- investigation of random locations before stashing acorns.
- altering their hoarding behavior based on the behavior of nearby squirrels
- working in small teams with familiar squirrels.

With the intention that the data to be stored will be evenly distributed throughout the network. Even data storage is facilitated by creating a uniform cache size and differentiating clients on the basis of their number of available caches. So each squirrel starts with a set of data that must be stored (the acorns) and proceeds through the P2P network looking for open caches where it can store those acorns. In experiments using a java implementation of squirrels (Figure 1.2.1) the authors found that the squirrels system maintained a well balanced P2P network and was arbitrarily scalable.[2]

1.2.2 Ad Hoc Swarm

Ziane and Melouk develop a CAS for handling the configuration of adhoc networks.[8] In this case, the metaphor employed is a more common swarm intelligence solution based on the behavior of ants. The problem that the authors focus on is on proactive routing and link allocation.

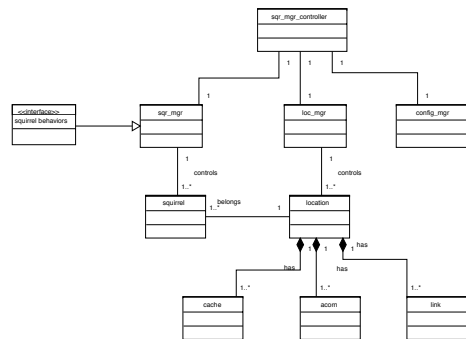


Figure 1: Class Diagram of the Squirrels System

When real ants forage, each ant randomly walks—while laying down scent—until they encounter a food supply. When the supply is found, the ant returns to its hill, following its own back trail, and laying down a different scent. Other ants that encounter the new scent will follow it and reinforce it. Each ant will make multiple trips to the food supply until the entire supply is transported back to the hill. The area where the food supply was located will become a focus for new foraging activity.

As each ant travels the path to the food supply, they make incremental changes to the path. Eventually, because more efficient paths will have heavier traffic, the ants will find a very efficient (usually straight) path between the food supply and their anthill, regardless of how poor the original path was.[7]

The components of this model that researchers typically borrow are the foraging and reinforcement behaviors. In the case of Ziane and Melouk, the methodology is to use several types of “ants”. Forward Ants search the network, creating a probability map of the network. Backward ants follow this trail, and populate the routing table with the information collected by the forward ants. This collective behavior results in an efficient and dynamic routing for the adhoc network.[8]

2 ALES

The Artificial Life Ecosystem Simulator is an open source platform for the development of artificial life, developed by Michael Balaun, Larry Eisenstein, Ed Bullwinkel, and John Daigle at Georgia State University. ALES models single celled life forms, which we call creatures, in a drop of water. It is meant to be a platform to test simple interaction rules between life forms and their environment, as well as evolutionary algorithms and immune algorithms.[4] Our purpose was to create a system that met several design goals. First, we wanted the system to be fun to play with. Second, we wanted the system to be adaptive. Third, we wanted to build a system that was extensible.

The current system falls short of these design goals to some degree. It is certainly possible to tweak the system to make user interaction crucial. New creatures are not difficult to build once you understand the creature class and the

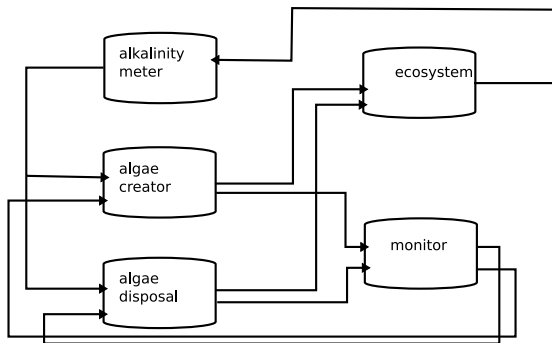


Figure 2: DEVS model for ALES

behave function. And the system does alter configuration constantly. The future roadmap is to include more swarming behaviors, more genetic algorithms, richer interactions, a meta or scripting language for extensibility, and a module for statistical reporting.

Inherent in the ALES system is its utter uselessness. There is nothing that can be done with ALES Java except to explore its own rules. This is because the system is inherently random and is not meant to create consistent output from consistent input.

2.1 ALES-DEVS

In contrast to ALES, DEVS is a coherent modeling system that is meant to create consistent output from consistent input. DEVS is based on atomic models with inputs and outputs. Inputs are translated through the state of the machine into an output. The state of a DEVS machine is time dependent.

In the design of ALES-DEVS (Figure 2.1) we were forced to abstract farther out from the system. In the ALES model, the system is built from the bottom up, the basic unit is the creature, and everything else is built around that idea. In the ALES-DEVS model, the basic unit is the ecosystem as a whole, and the creatures are abstracted to integers, although they can be expanded to objects in future.

What this means is, while the ALES-Java system is essentially unpredictable, the ALES-DEVS model is not. It can be predicted, from a set of inputs, whether the system will reach equilibrium or not. This has an interesting, but unintended, consequence.

The ALESDEVS model consists of 5 parts, the ecosystem, which keeps track of the units of algae, the alkalinity meter, which measures the alteration in alkalinity over time, algae creators and destroyers, which add or remove algae from the system, and a monitor for outputting all levels and measuring when equilibrium is reached.

Because the model is consistent, it actually acts as an effective algorithm for balancing forces. This means that there is potential in the ALES-DEVS model to be used for problems of bringing a dynamic system into equilibrium. If the ecosystem model were replaced by something more complex,

such as a network model, and the algae model were considered a packet input, the alkalinity represents interference, and the algae remover packet loss.

This model could produce, as an output, a measure of the robustness of a network, networks that had increasing algae populations would be very robust, those with decreasing populations very poor. At equilibrium, we have a measure of potential load and expected packet loss.

3 CONCLUSION

CAS programming is an effective means for solving hard problems. In the future, it will become a dominant means of solving hard problems, as our understanding of CASs evolves. The ALES-DEVS model is a step in the direction of using DEVS, a consistent and reliable modeling platform, to investigate such systems.

REFERENCES

- [1] M. Le Bars, J. M. Attonaty, S. Pinson, and N. Ferrand, *An agent-based simulation testing the impact of water allocation on farmers' collective behaviors*, *Simulation* **81** (2005), no. 3, 223–235.
- [2] Sergio Camorlinga and Ken Barker, *A complex adaptive system based on squirrels behaviors for distributed resource allocation*, *Web Intelli. and Agent Sys.* **4** (2006), no. 1, 1–23.
- [3] Oliver Deussen, Pat Hanrahan, Bernd Lintermann, Radomír Měch, Matt Pharr, and Przemyslaw Prusinkiewicz, *Realistic modeling and rendering of plant ecosystems*, SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM Press, 1998, pp. 275–286.
- [4] Zhonghua Li, Jianping Wu, and Zongyuan Mao, *Application of artificial immune algorithm in the dynamic zoning of elevator traffic*, *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on, 2004.*
- [5] R. Puzone, B. Kohler, P. Seiden, and F. Celada, *Immsim, a flexible model for in machina experiments on immune system responses*, *Future Generation Computer Systems* **18** (2002), no. 7, 961–972.
- [6] John Roach, *Bee decline may spell end of some fruits, vegetables*, 2007.
- [7] Peter Tarasewich and Patrick R. McMullen, *Swarm intelligence: power in numbers*, *Commun. ACM* **45** (2002), no. 8, 62–67.
- [8] Saida Ziane and Abdelhamid Melouk, *A swarm intelligent multi-path routing for multimedia traffic over mobile ad hoc networks*, Q2SWinet '05: Proceedings of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks (New York, NY, USA), ACM Press, 2005, pp. 55–62.