Table of Contents

Introduction

Jake Good

Charles Darwin stirred the religious mindset of creationism with his theories on evolution and natural selection. He questioned, "How will the struggle for existence…act in regard to variation?" (Darwin, Chapter 4**)** He took simple man made concepts such as selection and applied them to the natural world to figure out how evolution worked. Then the real question became: Can we apply the basic knowledge that Darwin schemed to other sciences, mainly computer science? Our group tackled both of those questions and extracted information from other sources to come up with our project, a genetic algorithm framework.

Genetic Programming

Using biological theories, mixed with *Darwinism*, computer scientists have formed a new branch in artificial intelligence[1] called genetic programming. Genetic programming takes natural selection, coupled with theories on sexual reproduction and genetics, and combines them to model evolution using a computer. More recently, the focus of genetic programming has centered around the use of evolving computer programs to take on certain actions that might not have been completed by a human.

Through the use of evolution, where generations of individuals undergo changes within their genetic makeup, we inject a survival of the fittest attitude that forces the individuals to seek out a way to live. Feeding off of the inclination to survive, the individuals within a generation will go through changes to become better suited to an environment. This is evolution from a biologist's viewpoint.

Applying these basic biological concepts to computer science yields a branch of artificial intelligence, genetic programming. Through these theories, scientists can implement a model of

[1] Artificial intelligence is the computational study of how a system can perceive, reason, and act in complex environments.

evolution that is fit for many applications within the computer science field, often leading to other discoveries only seen through the process.

Description

Being mindful of Darwin and other scientists in the field of genetic programming, our group chose the project of implementing and demonstrating a genetic algorithms framework. The main idea behind the project is to give programmers an easy implementation of a genetic algorithm to reduce the time spent on working out the details of the algorithm itself. With this, the programmer can spend more time on the actual implementation of an individual in which to evolve. On the other hand, the user of the system, with an implementation in hand, can visually use the application front-end to evolve individuals within a population to be used for other projects.

The framework was built upon the notion that the programmer and user should not know anything about the implementation of the algorithm. The research has been done on the genetic algorithm itself, so the framework allows the user/programmer to focus on what is to be evolved, rather than how it will evolve. Using an object-oriented design patter, our group has developed a set of classes in which the algorithm can easily be instantiated, adapted for other implementations, and to remove the connection between the system and the user/programmer.

Within the framework are a series of objects which become transparent to the user of the system and the programmer. A genetic algorithms strategy was put in place to handle the algorithm itself without any details of the specific implementation. The strategy is the heart of the algorithm and controls the flow of the evolution.

The actual implementations can be easily adapted to suite any problems. This was the second focus of our project. The framework has two interfaces in which the programmer/user can create their own individuals to evolve. Whether it’s a checkers player, a delivery route finder, or a simulation of animal breeding, the framework provides an easy interface in which the object being evolved only has to know how to reproduce itself.

The beauty of this framework is that there are only three connections between the system and the user/programmer. From the standpoint of the programmer, the only points in which have to adhere to an interface are: the object that evolves, the function that represents the fitness of the object within its environment, and the selection of parents within the system. This brings in the creativity of the programmer and user.

The current limitations of the project are within the framework itself. Currently there is one implementation of parent selection that is hard coded into the genetic algorithm strategy. The flipside is that if there is a desire to not use the roulette wheel selection[2] then all the programmer would have to do is subclass the strategy to include the new selection function. The other limitations is when the user imports the specific individual and fitness function classes, they must be compiled and in the root directory alongside of the core files. This enables the framework to easily check the object types to adhere to our interfaces without much trouble of compiling during runtime and other limitations of having raw source code files.

Startup Instructions

Jayapal Prabakaran

Genetic algorithm Framework startup

Follow the numbered instructions for startup.

1. The computer should be installed with ***JDK 1.3*** for optimum performance. Download and Install from http://www.java.sun.com. Installation procedure can be found in the package.
2. Check whether the JDK is installed properly. For instructions, see JDK 1.3.

[2] Roulette Parent Selection is a method of selecting individuals for reproduction by choosing a random number in the range of the total fitness of individuals. Each individual is given a range as large as its fitness, so that stronger individuals will have a better chance of being selected.

3. Operating System

- **Windows Platform**

    1. Open up command prompt window. Go to windows *"Start"*, then to *"Programs"*, click on *"Ms-Dos Prompt"*.
    2. Go to the directory where the classes are located for genetic algorithm framework. If the user didn't change, you should go to the directory *"GAApplication3"*.
    3. Type: **java edu.uni.GAFramework.GAApplication [FitnessClass] [IndividualClass]**
        - Please note: The FitnessClass and IndividualClass parameters are optional**.**

- **Unix Operating System or Linux**

    1. Go to X-windows.
    2. Enter the directory structure where the classes are located. If the user didn't change them, you should go to *"GAApplicaiton"*
    3. Type: **java edu.uni.GAFramework.GAApplication [FitnessClass] [IndividualClass]**
        - Please note: The FitnessClass and IndividualClass parameters are optional**.**

Shutdown Instructions

Jayapal Prabakaran

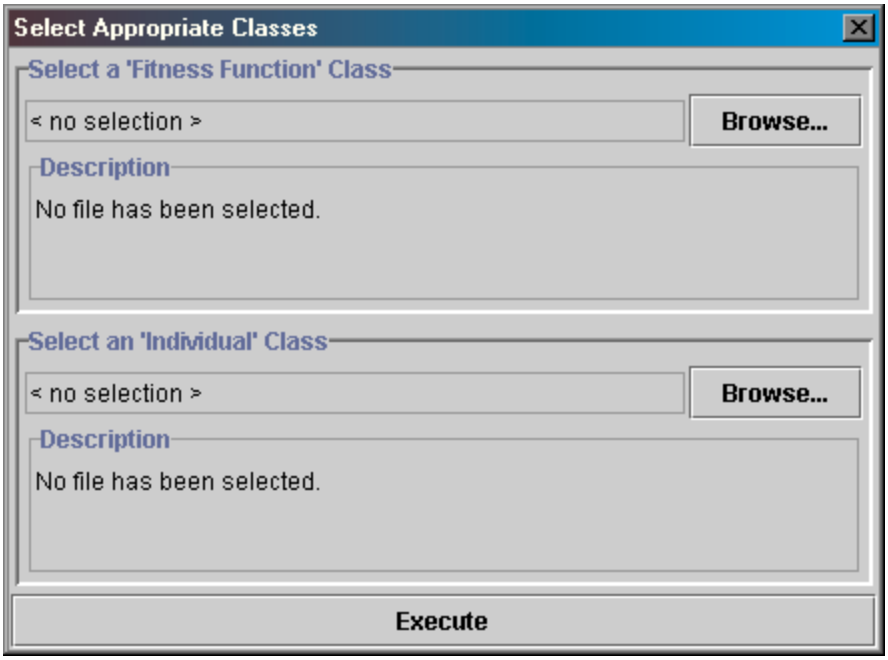
Genetic algorithm Framework shutdown

1. In the interface, top right corner click on ***"X"*** for proper shutdown.

How to Use the System

Patrick Burke

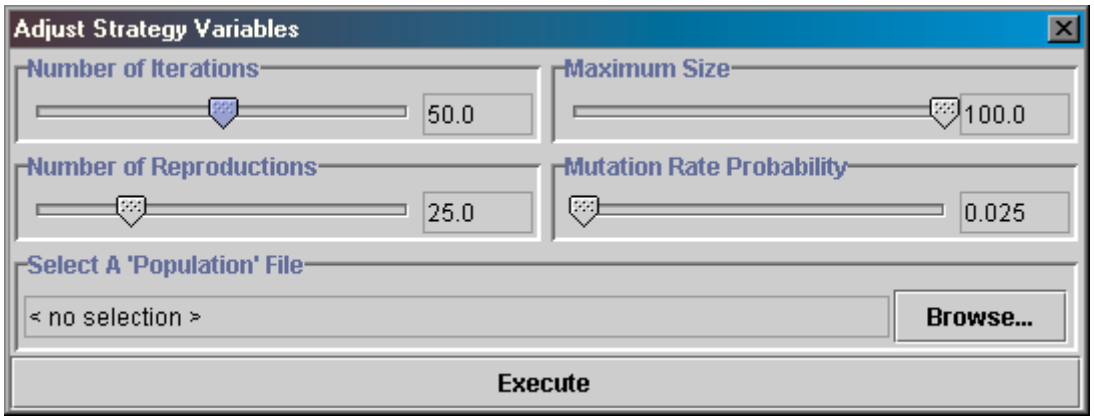
1. Once the application has begun, the following screen will appear, unless command line parameters have been entered to select an “individual” and a “fitness function.”



   Choose “**Browse…**” to locate these files on your computer, and then choose “**Execute**.”

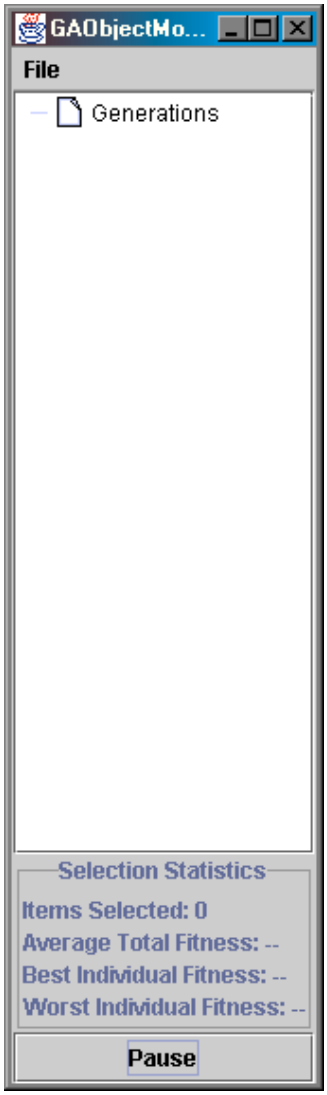
2. The following screen will allow you to set Genetic Algorithm Parameters. Each slider is adjustable to allow for custom settings. You may also choose to leave these parameters at the default values.

   An initial population of individuals will be generated at random, unless you choose to begin with a pre-generated population. You may do this, by browsing through your computer to find a previously saved generation.



When you are finished editing the parameters, choose "**Execute**"

3. Evolution begins immediately at this point. The GAObjectMonitor will then appear, allowing you to view Generations as they appear.

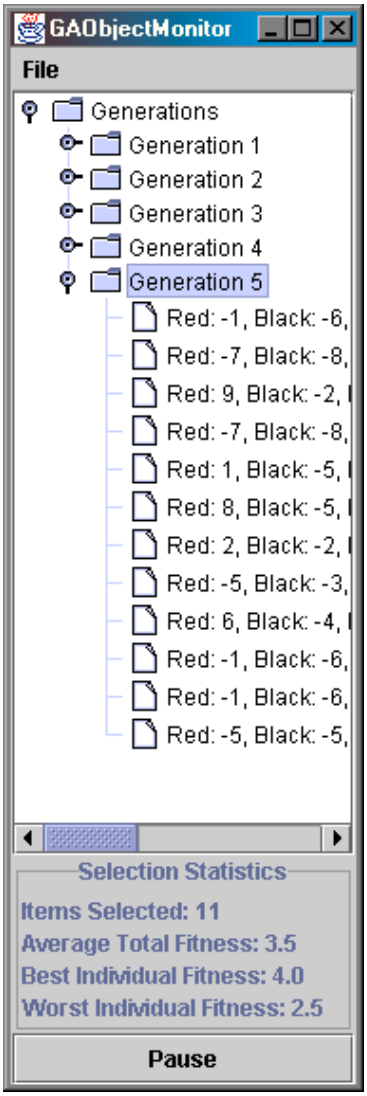


Listings of Generations will appear as the evolution process continues.

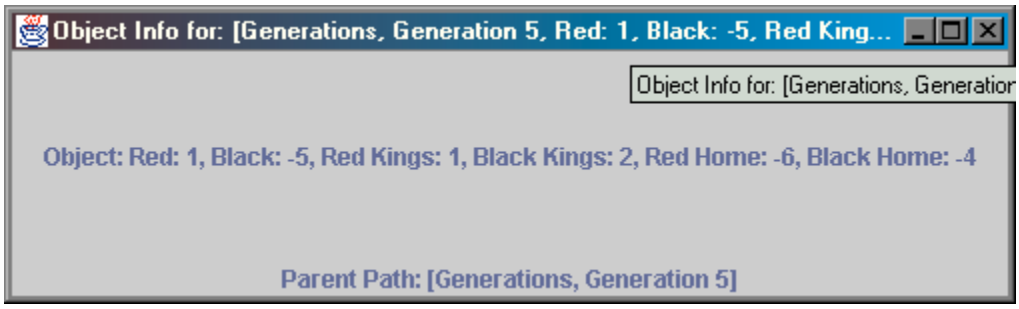
You can expand generations to look at each individual member of the population

Statistics will be updated dynamically on the bottom of the screen.

You can pause, and resume evolution as you see fit.

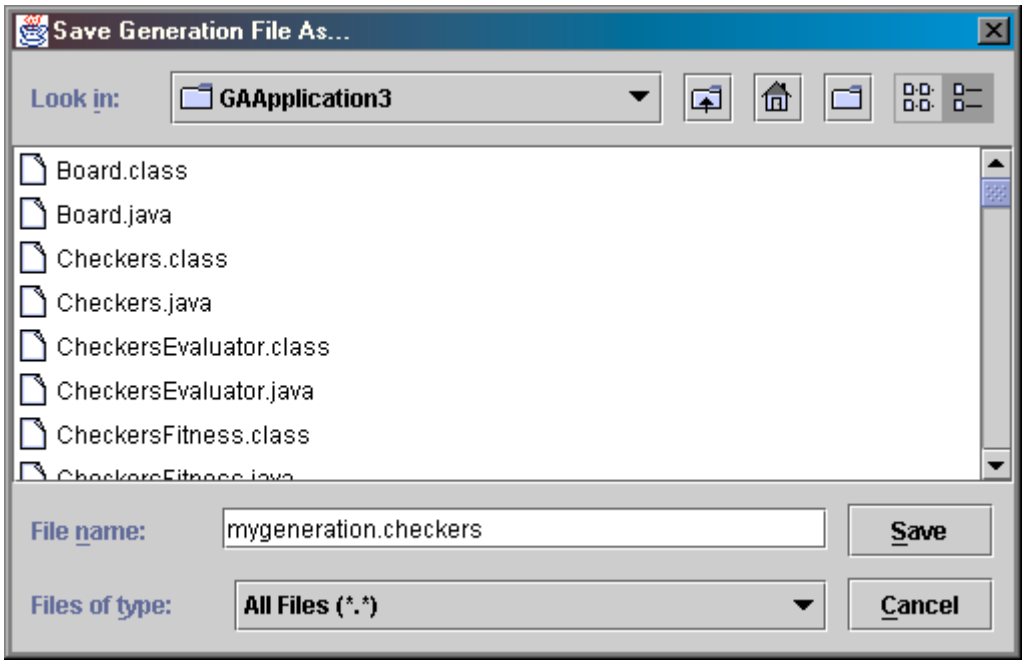


4. If you wish to keep track of a certain individual, you can double-click on it to show its Object Information screen



5. Once you are satisfied with the evolution process, you can save the individuals to be used in later evolution, or for practical use.[3]

   On the File Menu, choose to save an entire generation, or just an individual, and the following screen will appear. Give your generation a meaningful name, and choose "**Save**."



[3] Please see our documentation on Checkers Implementation for an example of practical use.

System's Error Messages

Ryan Dixon

Potential Errors:

Due to the dynamic nature of this framework, all errors documented within this section relate to potential problems that arise during user interaction.

At the time of this first release, only minor exception handling is being used on code that is not affected by the user.

Error with command line parameters

This problem arises upon first instantiating `GAApplication`. Files selected via the file browser must be compiled and located in the current-working directory (where `java GAApplication` is performed) or within one of the directories specified by the system PATH variable. If any of the class files cannot be located, the system will halt and return with a message indicating that one or more files were not found. Certain situations will flag a null pointer exception error; however, the result is identical, the system will halt. To continue, simply attempt another (valid) instantiation of `GAApplication`.

Error while saving in GAObjectMonitor

All write-related file errors are propagated and handled by the Java System; no attempt at recovery is performed! Two likely causes of failure in the creation of a file should be checked. First, ensure that the `GAIndividual` class being saved is serializable. Secondly, ensure that enough free disk space is available for writing the selected objects.

Error while importing saved object data

All read-related file errors are propagated and handled by the Java System; no attempt at recovery is performed! A read error will occur when reading from an object stream if different versions of the same class are used between exporting and importing! Ensure that only identical files are being used when writing and reading data. As of this release, not attempt has been made to create an `Externalizable` object; only the default serialization process is used.

Array index out of bound exception

This issue may arise if the user is allowed to select parameters (as in `GAStrategy's` GUI with slider values) that allow for boundary conditions to be exceeded. To prevent index boundary errors from occurring be sure to restrict user selections to only valid ranges within your code. As a means of prevention, all boundary conditions should be tested before allowing free configuration of internal parameters.

System Recovery Guide

Ryan Dixon

Avoiding Potential Problems:

In the case of a system failure or electrical outage during operation, very little can be done to resurrect un-saved data. As of release one, no auto-save features have been enabled, thus it is very important to save (and backup), any data that has been created via the `GAObjectMonitor`, frequently.

If a system does fail during operation, restart the machine as necessary in order to get back to a familiar starting point. If `GAApplication` or any of the framework components stop working, all of the base files, included with the software CD, may be safely replaced. (Refer to the user-installation manual for a detailed set of instructions on how to re-install software from the CD).

Bibliography

1. Darwin, C. The Origin of Species First Edition, Chapter 4.1859

2. Ginsberg, M. Essentials of Artificial Intelligence. Morgan Kaufman Publishers, Inc. San Francisco, CA. 1993.