1 Scripture

Amos 8:12
12. And they shall wander from sea to sea, and from the north even to the east, they shall run to and fro to seek the [exit of the maze], and shall not find it.

2 Abstract

2.1 Focus

The focus of this project is genetic algorithms.

2.2 Application

The application of this focus is creating a genetic algorithm library and using it to use to solve well-formed mazes.

3 Background

3.1 Definitions

3.1.1. **Breeding:** Producing one or more children from a population of parents through an iterated process of selection and tweaking. [1]

3.1.2. **Child:** A child is the tweaked copy of a candidate solution (its parent). [1]

3.1.3. **Chromosome** A genome in the form of a fixed-length vector. [1]

3.1.4. **Crossover:** A type of tweak where a combination of the genomes from two parents are used. [1]

3.1.5. **Deceptive Problem:** A problem where solutions tend to converge on local optimums instead of the global optimum. [2]

3.1.6. **Fitness:** A measurement of the quality of a candidate solution. [1]

3.1.7. **Gene:** A particular position in a chromosome. [1] Applied to this project, a gene will be a direction to attempt to travel (e.g., north, east, south, west).

3.1.8. **Generation:** One cycle of fitness assessment, breeding, and population reassembly. [1]

3.1.9. **Genetic Algorithm:** A genetic algorithm is a search heuristic. [3] Genetic algorithms search for a solution to a problem by mimicking evolution's processes of crossover and mutation within a population. [2]

3.1.10. **Genome:** An individual's data structure, as used during breeding. [1]

3.1.11. **Global Optimum:** The best possible solution to a problem. [4]

3.1.12. **Individual:** A candidate solution. [1]

3.1.13. Local Optimization: A solution to a problem that is better than all other solutions that are slightly different, but worse than the global optimum. [5]

3.1.14. Mutation: Pseudo-randomly tweaking an individual's genome. [1]

3.1.15. Population: A set of candidate solutions. [1]

3.1.16. Parent: A parent is a candidate solution whose genome was used to create a new candidate solution (its child). [1]

3.1.17. Selection: Picking individuals to tweak based on their fitness. [1]

3.1.18. Tweak: To make a change to the genome of a candidate solution. [1]

3.1.19. Well-formed Maze: A well-formed maze has the following properties:

- It must be a finite, undirected, connected graph.
- It must have only one entrance vertex.
- It must have only one exit vertex.
- The entrance vertex and the exit vertex must not be the same vertex.
- There exists at least one simple open path from the entrance vertex to the exit vertex whose length is less than the maximum length of the genomes of the individuals traversing it.
- The degree of each vertex must be less than 5.

3.2 Why Genetic Algorithms?

My interest in Genetic Algorithms was sparked by watching a YouTube video[1] about Mari/o. The author combined neural networks and genetic algorithms to create a program that can beat various levels from the classic SNES game Super Mario World. The idea of a program building up a complicated program from simple pieces was very appealing to me. Previous to hearing about Mari/o I had briefly read about an "evolved antenna[2]" which used an evolutionary algorithm to make an antenna suited to the X-band. [6]

3.3 Prior work by others

Applying genetic algorithms to maze solving has been done by others in the past. It is a relatively simple application of genetic algorithms, with the main difficulty being the deceptive nature of the problem (See **3.1.5**). Thomas Pasquier and Julien Erdogan from the Institut Superieur d'Electronique de Paris solved this problem by removing "rooms" from the maze (meaning they limit the path through the maze to only be one space wide), implementing a specialized segment crossover, and using an optimization for mutation that corrects the first move that hits a wall into a valid move. [3] Arild Berg also uses optimization in his approach. [7] Additionally, he created a website that allows you to adjust several of the variables involved, including population size, crossover scheme, max generation steps, population

[1] `https://www.youtube.com/watch?v=qv6UVOQ0F44`

[2] The one I read about was NASA's ST5 antenna. `https://en.wikipedia.org/wiki/Evolved_antenna`

size, gene size, mutation ratio, etc. [7] Forrest Sondahl's approach was to create a general maze solving algorithm creating a genome out of rules rather than discrete steps to solve the maze. [8] His website also features the ability to adjust the variables and watch the program run. [8]

3.4 Prior work by me

Excluding the research I've done to write this paper, I have done no work on genetic algorithms or programmatic maze solving. I have studied graph theory in discrete mathematics, which will be useful for this project. I also enjoyed solving mazes when I was a kid.

4 Description

4.1 Description of project

The goal of this project is to fully implement a genetic algorithm library such that it can be used to solve well-formed mazes, and then successfully use it.Genetic Algorithms differ from other evolutionary algorithms by selectively breeding and by including individuals from previously evaluated generations in subsequent generations. [2] I plan to apply genetic algorithms to maze solving as follows.

There are 4 phases to genetic algorithms, the last 3 occuring in a loop until an end condition is met. (In this case solving the maze or a certain amount of time passing.)

- Setup - Setup your fitness function, your genome, and generate a starting population.
- Fitness Evaluation - Evaluate the fitness of each individual in your current population.
- Selection - Select the individuals to keep in the population. From those you keep, select who you will breed to create the next generation.
- Breeding - Using a mix of crossover and mutation, breed a new generation of individuals and add them to your current population.

Each movement in the maze will be encoded as a gene. 0 for North, 1 for east, 2 for south, 3 for west. The chromosome for each individual will be a sequential list of movements taken. Once I have created the genetic algorithm library and solved the maze with this solution, if I have additional time I may attempt a more complicated approach. For this attempt, instead of encoding movements, I would write a set of functions and each gene would represent a different function the individual can call. The end goal of this attempt would be a solution that would work for any well-formed maze. Regardless of reaching the latter goal, I will also create a UI for the project that allows the user to initiate the maze solving and see the results for each generation. I may also include in the UI a way to watch the process of each individual as it makes its way through the maze.

4.2 Success Criteria

For this project to be considered successful it must:

- Create a genetic algorithm library that is fit for solving well-formed mazes.
  The minimum qualifications of fit for use are:

  - It must be able to create a starting population of individuals
  - It must be able to evaluate the fitness of each individual in the population
  - It must be able to select the best individuals and breed them.
  - It must be able to mutate the individuals selected for the new generation
  - It must be able to repeat this cycle using the last created generation
  - It must be able to terminate when a desired fitness level or a predefined stopping time is reached.
  - It must include the option to change the algorithms used for selection, mutation, optimization, and fitness.

- Use the genetic algorithm library to create a solution capable of reaching the exit of a well-formed maze.
- Allow the user to initiate maze solving.
- Display the results to the user

4.3 Tasks

4.3.1. Preliminary research and proposal preparation

4.3.2. Research

4.3.2.1. Research a maze generating algorithm or decide to create one.

4.3.2.2. Research selection, breeding, mutation, optimization, and fitness algorithms.

4.3.2.3. Research a method to display the results

4.3.3. Requirements specification

4.3.4. Design

4.3.4.1. Select a maze generating algorithm.

4.3.4.2. Select what genes to include.

4.3.4.3. Select the fitness, selection, breeding, mutation, and optimization algorithms.

4.3.4.4. Create a UML diagram for the library.

4.3.4.5. Create a UML diagram for the UI.

4.3.4.6. Create Jira issues to prepare for development

4.3.5. Development

4.3.5.1. Setup a maven project in NetBeans

4.3.5.2. Develop the Java interfaces for:

- Fitness

- Selection
- Breeding
- Mutation
- Optimization
- Genes
- Chromosomes
- Individuals
- Population
- Mazes

4.3.5.3. Develop the genetic algorithm driver

4.3.5.4. Implement the interfaces for:

4.3.5.4.1. Fitness[3]

4.3.5.4.2. Selection

4.3.5.4.3. Breeding

4.3.5.4.4. Mutation

4.3.5.4.5. Optimization

4.3.5.4.6. Genes

4.3.5.4.7. Chromosomes

4.3.5.4.8. Individuals

4.3.5.4.9. Population

4.3.5.4.10. Mazes

4.3.5.5. Develop the UI

4.3.6. Quality Assurance

4.3.6.1. Perform unit testing on the interface implementations[4]

4.3.6.2. Perform integration testing on the genetic algorithm driver with the implemented interfaces

4.3.6.3. Perform integration testing on the UI

5 Scope

This section describes the MVP for the project. If this project is completed ahead of schedule, I reserve the right to include items from the "What is not included" subsection.

[3]I recognize that this level of reference nesting is ridiculous. In spite of that, I will likely need to refer to these tasks individually in my weekly report, so I used it anyway.

[4]Did you get as tired of reading the laundry list of interfaces as I was of writing it?

5.1 What is included

- A non IEEE standard design document
- A non IEEE standard requirements document
- A genetic algorithm library written from scratch in Java by me.
- The implementation in Java of crossover, mutation, selection, and fitness algorithms.
- A Java program that uses my genetic algorithm library to solve a well-formed maze.

5.2 What is not included

- The conceptual creation of crossover, mutation, selection, or fitness algorithms[5]
- The creation and implementation of a maze generating algorithm
- A graphical user interface
- Solving mazes that are not well-formed
- Solving mazes supplied by the user.
- Preparing the genetic algorithm library for distribution.
- Complete documentation (User Manual, IEEE standard SRS or SDD[6], etc.)

6 Task and Schedule

Task Reference and Name	Start Date	Stop Date	Estimated Hours
4.3.1 Preliminary Research and Proposal	**12/01/2015**	**12/11/2015**	**30**
4.3.2.1. Research maze generating algorithm	12/01/2015	As needed[7]	*
?? Research genes to use	12/01/2015	As needed	*
4.3.2.2. Research various algorithms	12/01/2015	As needed	*
4.3.2.3. Research display method	12/01/2015	As needed	*
4.3.3 Requirements specification	**12/11/2015**	**12/16/2015**	**32**
4.3.4 Design	**12/11/2015**	**12/19/2015**	**32**
4.3.4.1. Select a maze generating algorithm	12/17/2015	12/19/2015	2
4.3.4.2. Select what genes to include[8]	12/01/2015	12/05/2015	2
4.3.4.3. Select various algorithms	12/17/2015	12/19/2015	16

[5]I will be finding and adapting the algorithms I find and implementing those algorithms in Java. I am not responsible for innovating a new algorithm for crossover, mutation, selection or fitness

[6]I will still be providing the non-standard SRS and SDD documents

[7]I performed most of my research concurrently with the creation of this proposal. I feel I have a good general knowledge base to work with. Further research will only occur as needed to further my design/development work, and related time will be grouped under the related design and/or development categories.

[8]already complete

Task Reference and Name	Start Date	Stop Date	Estimated Hours
4.3.4.4. UML for Library	12/21/2015	12/23/2015	8
4.3.4.5. UML for the UI	12/21/2015	12/23/2015	2
4.3.4.6. Create Jira issues	12/17/2015	12/19/2015	2
4.3.5 Development	**12/17/2015**	**01/02/2016**	**70**
4.3.5.1. Create NetBeans project	12/17/2015	12/19/2015	2
4.3.5.2. Create interfaces/stubs	12/21/2015	12/23/2015	24
4.3.5.3. Develop the driver	12/28/2015	01/02/2016	4
4.3.5.4.1. Implement fitness	12/28/2015	01/02/2016	6
4.3.5.4.2. Implement selection	12/28/2015	01/02/2016	4
4.3.5.4.3. Implement breeding	12/28/2015	01/02/2016	8
4.3.5.4.4. Implement mutation	12/28/2015	01/02/2016	2
4.3.5.4.6. Implement genes	12/28/2015	01/02/2016	2
4.3.5.4.7. Implement chromosomes	12/28/2015	01/02/2016	2
4.3.5.4.8. Implement individuals	12/28/2015	01/02/2016	2
4.3.5.4.9. Implement population	12/28/2015	01/02/2016	2
4.3.5.4.10. Implement a maze	12/28/2015	01/02/2016	4
4.3.5.5. Implement UI	12/28/2015	01/02/2016	8
4.3.6 Quality Assurance	**12/28/2015**	**01/05/2016**	**8**
4.3.6.1. Unit test interfaces	12/21/2015	01/05/2016	6
4.3.6.2. Integration test driver & UI	01/02/2015	01/05/2016	2
Total Hours			140

7 Deliverables

- All source code for the project
- A Java jar or war file[9] of the maze solving program.
- UML for library, driver, and UI from design time.[10]

8 Applicability

8.1 Integration with previous material

- Discrete Mathematics
  Working with mazes and maze solving requires me to use graph theory
- Object Oriented Programming and Java
  I will be programming this assignment using Java and following the Object Oriented paradigm.
- Data Structures
  I will need to represent the maze as a graph of connected nodes.

[9]Haven't made this design decision yet.
[10]If required, I can also deliver updated UML diagrams that reflect the state of the deliverable jar/war

- Technical Communication
  Creating this proposal, the simplified SRS, the summary, and the final presentation will draw on the professional skills I learned in technical communication.
- Software Engineering
  The evidence of design and the requirements specification will draw on the engineering skills I learned in Software Engineering.

8.2 Exclusive of CS curriculum

Genetic algorithms may be a part of the Machine Learning class, but I don't know for sure as I haven't taken the class. No other class on campus I know about reviews the principles of genetic algorithms, and I haven't studied genetic algorithms in any of my course work (excepting the research for this class).

9 Required Resources

Resource	**Cost ($)**
A computer connected to the Internet and capable of running NetBeans	0.00[11]
Jira 7.03 Server edition	10.00[12]

References

[1] S. Luke, *Essentials of Metaheuristics*, second. Lulu, 2013. [Online]. Available: `https://cs.gmu.edu/~sean/book/metaheuristics/`.

[2] P. Charbonneau, "An introduction to genetic algorithms for numerical optimization," *NCAR Tech. Note TN-450+ IA, 74pp*, 2002.

[3] T. Pasquier and J. Erdogan, "Genetic algorithm optimization in maze solving problem," *Institut Superieur d'Electronique de Paris*, [Online]. Available: `http://geneticmaze.googlecode.com/files/paper.pdf`.

[4] P. E. Black. (2004). Global optimum, [Online]. Available: `http://www.nist.gov/dads/HTML/globalOptimum.html` (visited on 12/07/2015).

[5] ——, (2004). Local optimum, [Online]. Available: `http://www.nist.gov/dads/HTML/localoptimum.html` (visited on 12/07/2015).

[6] G. S. Hornby, A. Globus, D. S. Linden, and J. D. Lohn. (2006). Automated antenna design with evolutionary algorithms, [Online]. Available: `https://www.researchgate.net/profile/Al_Globus/publication/228909002_Automated_antenna_design_with_evolutionary_algorithms/links/547375300cf216f8cfaff65a.pdf` (visited on 12/08/2015).

[7] A. Berg. (Dec. 2, 1999). Genetic algorithm - maze solver, [Online]. Available: `http://home.sambee.co.th/MazeSolver/mazega.htm` (visited on 12/01/2015).

[8] F. Sondahl. (Nov. 28, 2005). Genetic programming maze rules, [Online]. Available: `http://cs.northwestern.edu/~fjs750/netlogo/final/mazerules.html` (visited on 12/08/2015).