**Project Plan**

**Genetic Programming**

**Project**

Author: Vishal Yelisetti, Ou Li

Creation Date: 3/21/2015

Last Revised: 3/21/2015

Version: 1.0

**Problem Description**

**Background Information/Available Alternatives**

Genetic programming is a model, which closely resembles what we know as Artifical Intelligence or AI. GP, as it is referred tries to provide a needed solution from a variety of congruent functions. These functions mutate, compete, cross-breed and survive to evolve into their parent functions. In instances where large number of factors affect a probable outcome, GP can be used to come up with an optimal solution from a few known and controllable variables.

**Problem Description**

The problem for this project is to create a Java project which implements genetic

programming to find another equation which equals to the target equation ! !!

!

!

within 15 minute.

**Requirements Analysis and Preliminary System Design**

**Requirements Analysis**

• **Data**

1. Data Models

Terminal set

The terminal set is composed of the inputs to the Genetic program. [0..9], Y, X.

2. Function set

The function set is composed of the statement, operators, and function available to the Genetic program. +,-\*,/, SQRT()

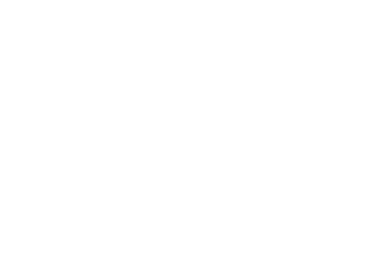
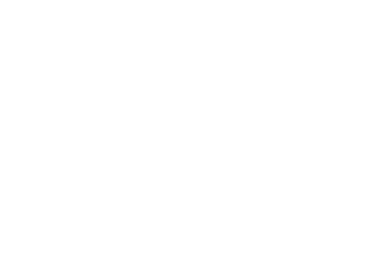
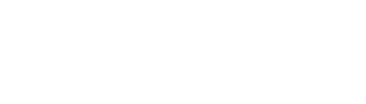
 Crossover: Single point crossover

 Mutation: change function or terminals

 Fitness function: compare produced output with expected outputs

Tree

Nodes: int Height:int Balance:boolean Fitness Value:double



Operations: Crossover Mutation

Fitness evaluation

GPMain

Tree[]funcGen Random rand; Int[][] train; Selection(); Delete();

Tree

Nodes: int Height:int Balance:boolean Fitness Value:double

Binary Tree

Nodes: int Height:int Balance:boolean Fitness Value:double

Node

Operator Node

Compute();

Char Data; Node Left; Node Right; Mute();

Operand Node

Mutate();

• **Function**

**1. DFD Level 0**

Settings

Training Data

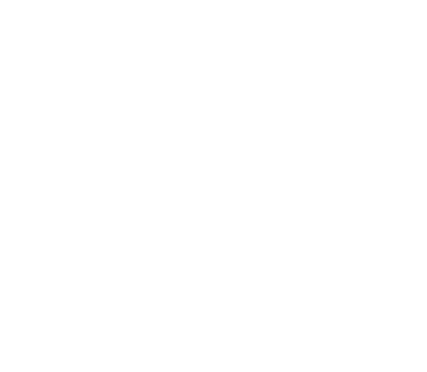
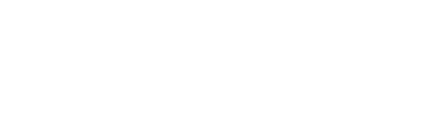
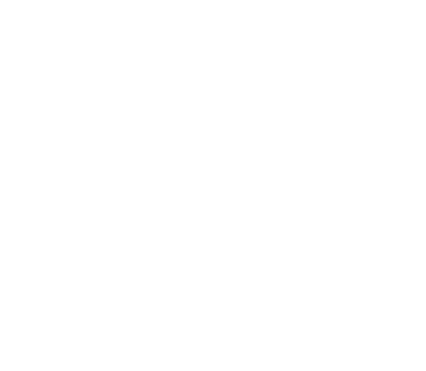
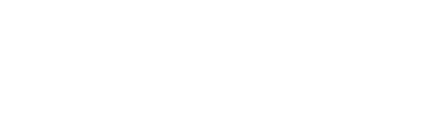
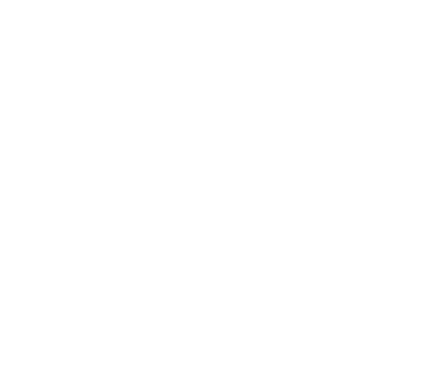
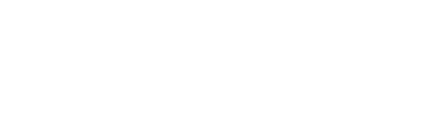
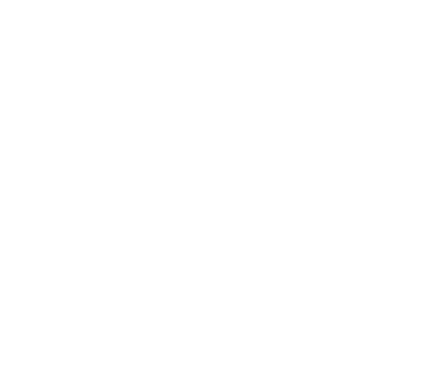
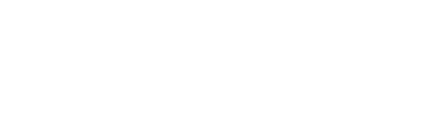
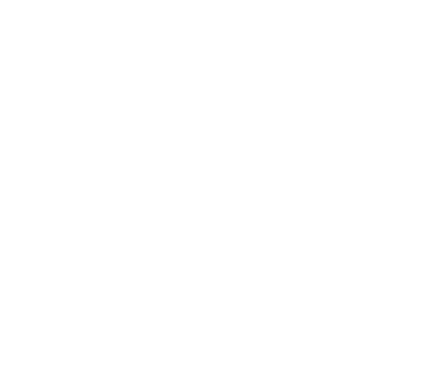
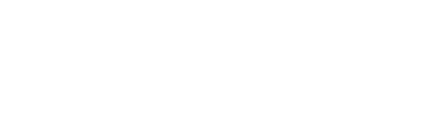
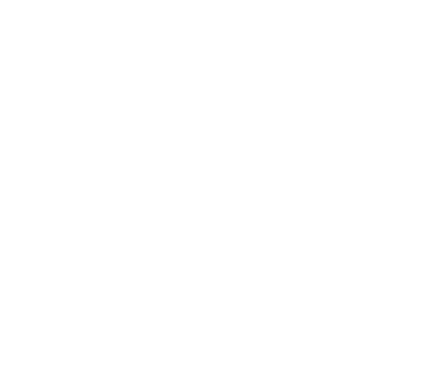
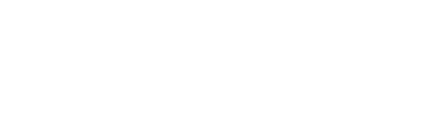
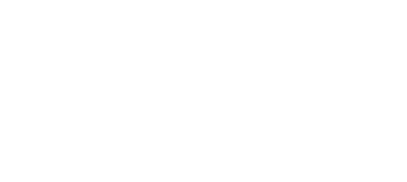
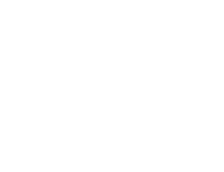
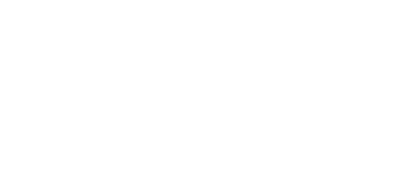
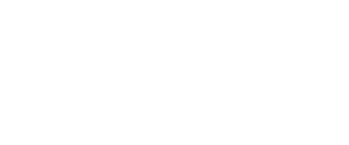
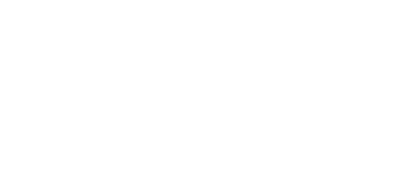
Settings

Evaluatio n of programs

Training Data

Genetic Programming Environment

Display result



**2. DFD Level 1**

Settings

Operands/Op erators/maxH tOfIniTree/Po pSize

Generate Initial Population

Training Data



Fitness

MarginOf

Error mutationProb

/numCrossov er/Operands/ Operators

Training Data

Crossover and mutation

Initial

Population

Fitness programs

Training

Data

Parsing Training Data

Evaluate Fitness of Population

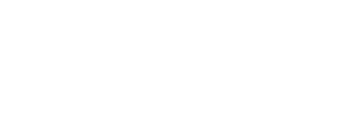
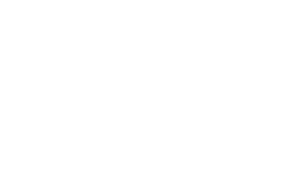
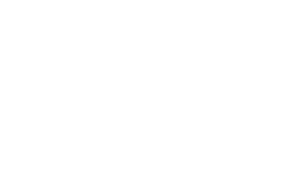
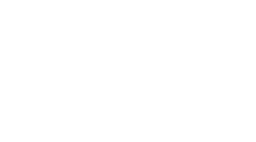
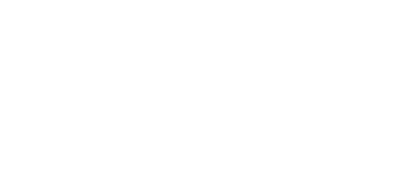
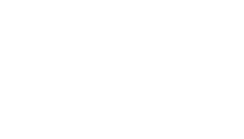
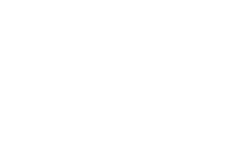
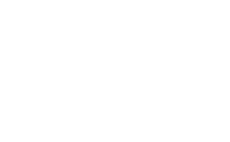
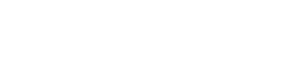
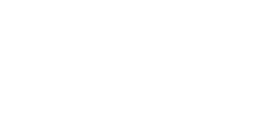
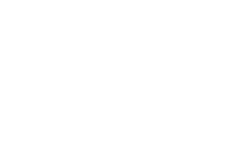
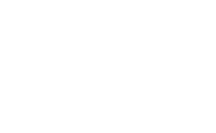
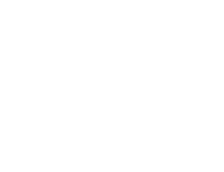
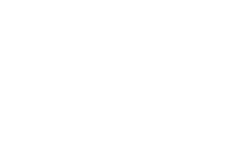
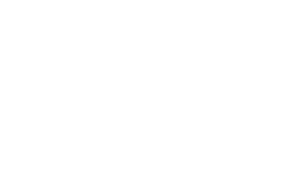
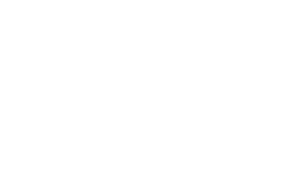
Selection

Evaluatio n of programs

Display

Result

Programs w/fitness values



• **Behavior**

**1. States**

They are observable circumstances that characterize a system.

And possible states are Initial state, training data set is in the system, Fitness of population is evaluated.

**2. Events**

They are the events which cause a state transition.

And Possible Events are parse/read training data, generate high, generate trees, generate notes, calculate fitness values, Sort, Select, Mutation, operates.

Briefly graph shows in following.

Process Training Data



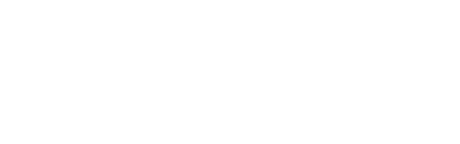
Start

Training Data is ready

Population is generated.

Calculate Fitness

Values



Done with

Regenerate

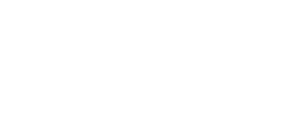
.

Calculate Fitness

Values

Fitness value is calculated

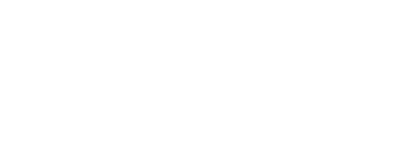
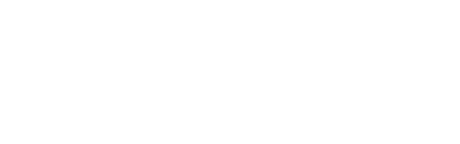
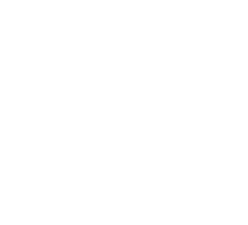
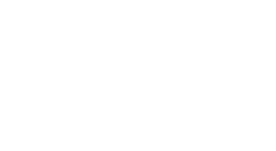
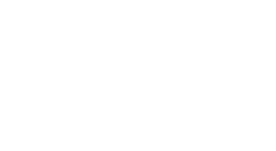
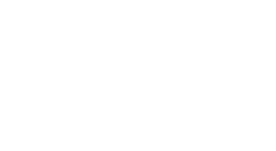
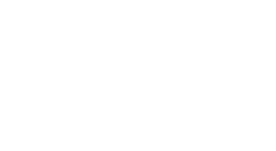
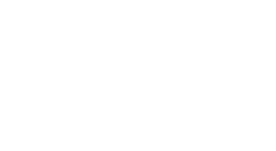
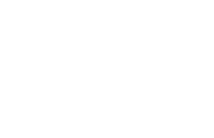
Select



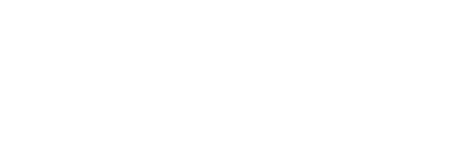
Crossover and mutation

Selection

Done



Compare with Target.



Equal to Target Equation

**System Design**

**System  Architecture**

The Genetic Programming System is developed using object oriented design principles in the Java programming language. Matlab was use an initial tool to gauge the appropriate functions and its common functions.

**Design  and  Development**

We were given the ability to create our GP system or use a pre-existing library. We chose to start from the ground and build the project on these fundamentals, as this would allow comprising of our own UI and giving us the flexibility needed to provide a sufficient answer. With the awareness that our project would change as the project drew to a close, we would have a better grasp of what needed to change in order to incorporate these fixes. The ability to change code on a pre-existing library would have been different and little more tedious.

Some key constraints that could arise in this project are as noted: developing an adaptive system that only works for our current system but any function. Other dependency is that, we were to be under the notion that our libraries would be created perfectly and gel together quite extensively. The other trade-off was that in developing from scratch, there is the notion that our competency with Java is high. Though, there is experience in the team, extensity will be tested nonetheless.

The choice of using an object oriented design and development approach using Java as our programming language was because of several reasons:

● The language has support for developing a graphical user interface. For our project however developing the graphical user interface was a low priority development task.

● Object oriented development is today a proven, mature, widespread, and successful approach in developing modern software systems. It offers a low risk, high re-use, well performing option for developing stable systems.

**Path  Chosen:**

We choose to use the binary tree data structure to represent a Genetic Programming Tree because of the greater flexibility and power of offered by such a structure in recursively traversing, performing genetic operations of crossover and mutation, and in the evaluation of the algebraic expression represented by each tree.

For representing a population of Genetic Programming trees, we used the Java ArrayList structure as it allow us to easily do the following:

● Iterate over the population and access it using indexes

● Convenient allow us to efficiently sort a population of trees

● Ability to travers through a large amount of population under a contain time.

**Weekly SCM Files and Folders**

• **SCM  Template**

**1. Name  of  work**

**2. Revision  Date**

**3. Changes**

**4. How**

**5. Where**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **SCM  Template** |  |  |  |  |  |
| **Week  2** |  |  |  |  |  |
| **Name  of  Work** | **Revision  Date** | **Changes** | **How?** | **Where?** | **Who?** |
| Problem  Description | 2/21/15 | Draft | Word  Document | Google  Document | Ou |
| Initial  Graphs | 2/26/15 | Graphical  Representation  of  target  functions | Matlab | Matlab | Vishal |
| GitHub  Repository | 2/27/15 | Draft | GitHub | Online | Ou/Vi shal |
|  |  |  |  |  |  |
| Week  3 |  |  |  |  |  |
| Intial  Release  to  Google  Docs | 3/4/15 | Matlab  Code  and  Explanation | Matlab | Online | Vishal |
|  |  |  |  |  |  |
| **Week  4** |  |  |  |  |  |
| Requirement  Analysis | 3/6/15 | Initial  Draft  of  the  Requirement  Analysis | Google  Doc | Online | Ou |
| Behavior  vs  Development | 3/10/15 | Initial  draft  of  Require  Behavior  and  Development | Google  Docs | Online | Ou/Vi shal |
|  |  |  |  |  |  |
| **Week  5** |  |  |  |  |  |
| UML | 3/14/15 | Initial  Design:  No  nodes | Google  Docs/  Paint | Online | Ou |
| UML | 3/20/15 | Initial  Add-­‐in  of  nodes/functions | Google  Docs | Online | Ou |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | and  classes |  |  |  |
| OO  Diagram | 3/21/15 | Initial  Start  at  OO  Diagram | Google  Docs | Online | Vishal |

**Work Plan**

**Breakdown Among Members**

• Part 1 Input (Ou) 20 hours

Population Size, training set choose

• Part 2 Operation (Vishal, Ou) [Note: this part is complicated, we will discuss about the plan together] 200 hours

Trees crossover mutation

• Part 3 Display (Vishal) 30 hours

Implementing a GUI to display / output to a textfile and analysis through

Excel or so.

• Part 4 Post-Project Analysis 8 hours

**Milestones**

|  |  |
| --- | --- |
| **Milestone**  **Phase I: Requirement Analysis**  Initial Steering Committee Meeting | **Estimated Completion Date**  2/15/2015 |
| **Phase II: Design**  Initial Steering Committee Meeting | 3/08/2015 |
| **Phase III: Implementation**  Initial Steering Committee Meeting | 4/14/2015 |
| **Phase IV: Testing**  Initial Steering Committee Meeting | 4/25/2015 |
| **Phase IIV: Presenting/Release** |  |

Initial Steering Committee Meeting 5/9/2015