

Genetic Algorithms for Robot Mobility

Deliverable 1: Final Year Dissertation

Jacob Cooper - H00251723 | Dr. Michael Lones | Computer Science|

# Introduction

## AiM

To implement a Genetic Algorithm in a robot for it to learn how to move in a simulated environment for the purpose of showing how Robots learning through Genetic Algorithms would perform in the real world.

## objectives

* Use Webots to Create a suitable simulated environment for the robot to move in ( M )
* Create a body for the robot to operate in Webots ( M )
* Add alternate bodies for the robot to operate in Webots( C )
* Create a genetic algorithm that allows a robot to traverse flat ground over a short distance ( M )
* Create a genetic algorithm for a robot to traverse uneven ground ( S )
* Create a genetic algorithm that allows a robot traverse obstacles ( C )
* Create a genetic algorithm that allows a robot to jump over gaps ( W )
* Apply different approaches to GA’s throughout the different scenarios ( C )
* Optimize the algorithm to move the robot efficiently across the terrain ( W )
* Have the robot learn the path from scratch as quickly as possible ( W )
* Have the robot learn to navigate a multitude of terrains at once ( C )

## Stakeholders

* Supervisor (Dr. Michael Lones)
* Heriot- Watt University
* Robotics Researchers
* Game Developers
* Companies which use robots
* Creators of any code libraries I use
* Creators of Webots
* Users of Webots

# Background

## Push

Designed to have a completely uniform syntax so the ease the process of generating code that manipulates code (say for auto constructive evolution) all while supporting a multitude of different data types as well as recursion and sub-routines. It is basically an extension of stack-based programming languages such as Forth(Salman 1984).

A push program consists of a string of instructions, constants and parenthesis. There is only one syntax rule : the brackets must be balanced. The instructions found in these strings take their arguments from a Global stack. Each data type has one global stack and instructions push their outputs onto these global stacks. Its written in postfix notation meaning that reading it for a human might be a little difficult (more on this later). Some instruction (such as ‘+’) may be skipped if there aren’t enough arguments on the argument stack. So for example if a program reads ‘integer’ it will be stored in the type data stack, then if a ‘+’ method appears it first looks at the type stack to see what type ‘+’ is. This is because ‘+’ could have multiple types (e.g. integer/string). Is there is no type on the stack then it will resort to a default. This default has a list of types consisting of Integer, Boolean ,code ,type ,name in that order from top to bottom.

# Project Management Plan

Research the multiple different implementations of genetic algorithms

## Work Breakdown Structure

### 1 –Initial Set-up and Research

* 1. - Research multiple different Genetic Algorithm implementations
  2. - Pick one Implementation method for GA’s to start with initially

1.3 - Acquire use of Webots and learn basics

1.4 - Create first environment with flat ground in Webots

1.5 - Create Initial robot body in Webots

1.6 - Add a start position and goal for Robot

### – Flat Ground Scenario Genetic Algorithm

* 1. – Implement first Genetic Algorithm for flat ground traversal
  2. – Combine the algorithm with the Webots Robot
  3. - Run GA in the Flat Ground scenario until Robot reaches goal or time runs out
  4. - Modify GA until Robot reaches goal or try different method for GA
  5. – Continue to run GA until Robot converges on method to reach goal

### 3 – Uneven Ground Scenario Genetic Algorithm

3.1 - Change Webots environment to include just uneven terrain on path

3.2 – Run Flat Ground GA on Uneven ground until robot reaches goal(Risk: or time runs out)

3.3 – Modify Flat Ground GA until Robot Reaches goal (Risk: or try different method for GA)

3.4 – Continue to run GA until robot converges on method to reach goal

3.5 – Add in flat ground and Uneven ground to simulation

3.6 – Run same GA on the new terrain until robot reaches goal

3.7 - Continue to run GA until Robot converges on method to reach goal

### 4 – Obstacle Scenario Genetic Algorithm

4.1 – Change Environment so there is a single obstacle on path to goal

4.2 – Run Uneven ground GA on Obstacle path until robot reaches goal

4.3 – Continue running GA until Robot converges on method to reach goal

4.4 – Add in flat and uneven ground to simulation before obstacle

4.5 – Run Same GA on new terrain until robot reaches goal

4.6 – Continue to run GA until robot converges on method to reach goal

### 5 – Final Testing of Genetic Algorithm

5.1 – Take final algorithm and run it on flat ground

5.2 -

### Webots

* Implement Robot in Webots
* In depth learning of the functionality and various uses of Webots
* Create alternate bodies in Webots

### Implementing Genetic Algorithms

* Implement first Genetic Algorithm in robot for traversing flat ground
* Use Flat ground GA for uneven ground and modify accordingly
* Use first ground GA for

### Testing

* Test Robot on flat surface
* Track how long Robot takes to learn path for flat ground
* Track how long Robot takes to learn path for uneven ground
* Track how long Robot takes to learn path for Obstacles
* Track how long Robot takes to learn path when robot has to jump over a gap