|  |
| --- |
| University of Reading |
| Evolutionary Computing |
| Evolutionary Robot Development with Robocode API |

|  |
| --- |
| Tom Bedford |

Contents

[Introduction 2](#_Toc446574198)

[Development 2](#_Toc446574199)

[Robot Solution 3](#_Toc446574200)

[**Design** 3](#_Toc446574201)

[**Results** 4](#_Toc446574202)

[Conclusion 5](#_Toc446574204)

[References 5](#_Toc446574205)

# Introduction

Evolutionary and Genetic Algorithms have been a focused area of research for many years. The most commonly found in practice today is the Genetic algorithm which is a sub class of Evolutionary Algorithms. GA’s are based on natural selection using cross-over and mutation operators to produce a heuristic algorithm [1]. Although both genetic and evolutionary algorithms deliver an adaptive and competitive method evolutionary programming has delivered more versatile and promising results [2]. Evolutionary and Genetic algorithms are stochastic formulas. This report documents the design and implementation of a genetic algorithm based robot developed using the Robocode API [3] in order to better absorb the course content delivered on the SE3EC11 evolutionary computing module.

# Development

Generic Algorithm Flow Chart **Figure .1**

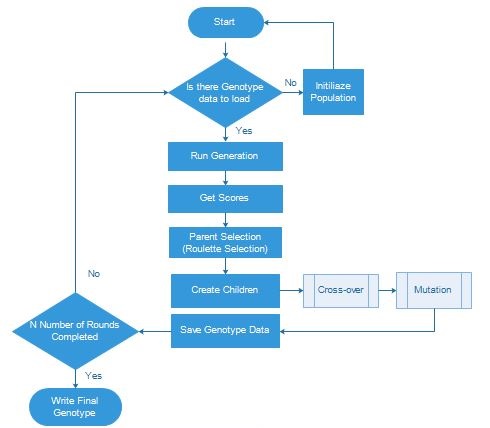


Figure 1 illustrates the hierarchy of the robot solution developed in this assignment. On each round natural evolution is performed on each phenotype representation to complete a generation.

## Robot Solution

### **Design**

This robot developed uses a genetic algorithm to evolve and optimize a targeted skill or trait. Phenotypes of each round within the emulated robocode battle are stored as character strings that represent chromosomes illustrated in figure.1 and 2. Genotype moves and scores are stored separately but whilst scores are sorted the switches are made in the Genotype data.

**Figure.1** Phenotype moves

|  |  |
| --- | --- |
| Chromosome1 | ;06,72,;01,20,;03,91,;07,84,;05,34,;05,21;04,50,;03,53,;05,91,;02,58,;04,94,;02,39,;06,24,;01,26,;01,89,;03,86,;07,46,;04,40,;03,70,;04,59, |
| Token of chromosome 1 | ;06,72, |
| Representation | ;behaviour,power, |

**Figure.2** Genotype scores

|  |  |
| --- | --- |
| Chromosome 1 | ;09,71,13,-07, |
| Representation | ;roundDamageTaken,defenseSkills,attackSkills,navigationSkills, |

The moves are built up of behaviours implemented as a selection of enums in java code and a power to signify the intensity of the behaviour. Figure.3displays the different behaviours that may represent a phenotype.

**Figure.3** Possible Behaviours

|  |  |
| --- | --- |
| ENUM | Behaviour |
| Case 0 | turnLeft(power) |
| Case 1 | turnRight(power) |
| Case 2 | ahead(power) |
| Case 3 | back(power) |
| Case 4 | turnGunLeft(power) |
| Case 5 | turnGunRight(power) |
| Case 6 | fire(power) |

The first round of the algorithm is used to generate a population size of 100 if no phenotype data is detected. The initial population values are generated at random. On each round one generation is carried out by performing behaviours and scoring each phenotype in the population by event. Events that happen within the Genotype dictate the scoring system in which robot fitness is derived. A list of the events that manage these attributes are listed in Figure.4. This robot design is based on a defensive approach where the fitness of the phenotype is determined by the ‘roundDamageAttack’. The next robot design would be catered towards attack.

**Figure.4** Attributes scored on event

|  |  |
| --- | --- |
| Event | Score attribute |
| onScannedRobot | attackSkills++ |
| onBulletHit | attackSkills + 1 |
| onHitRobot | navigationSkills – 1  roundDamageTaken +1 |
| onHitByBullet | roundDamageTaken + 1  defenseSkills – 1 |
| onHitWall | navigationSkills – 1  roundDamageTaken + 1 |

To add structure and flexibility to the crossover and mutation functions a token class was implemented to store phenotype moves and scores as an object in java code. This allowed easier comparison of phenotype scores and manipulation of chromosome attributes during offspring creation. To allow this object orientation an encode/decode function was also developed so that the population could be stored as the string representations discussed early in the report.

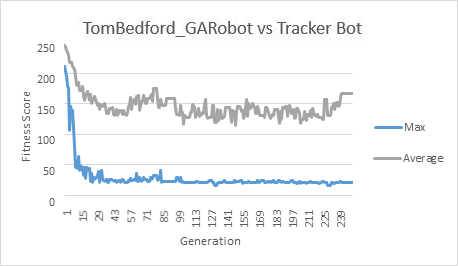
To produce 100 children 200 parents are first selected using a roulette selection method. So to select 100 parents this is carried out by firstly sorting the gene strings by score and then selecting the first 20 parents from the top 20 results, the next 50 from the top 50 and the remaining from parents from anywhere in the list. This essentially acts as a probability based ranking system that delivers a more realistic gene selection. This also helps to counter elitism within the population.

N-point crossover is used to create the initial offspring prior to mutation. A random number is generated between 0 and the gene string length which represents the chromosomes that will be switched in the crossover. Consideration was taken into account that when a 0 point crossover could occur resulting in no modification to the gene. This was left in as again it delivers a better natural selection and adds more variety of phenotype genes. There is a set mutation rate of 5% of the population. Once the generation child population has been created 5% of the mutation

### **Results**

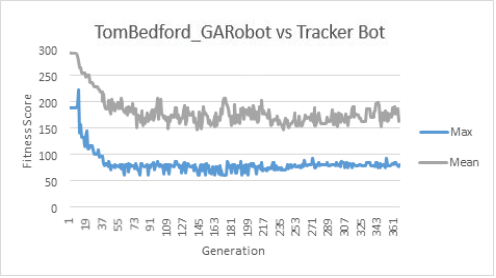
­­­­The ‘TomBedford\_GARobot’ robot was subject to testing against the sample robocode robot ‘Tracker Bot’ over 250 generations. The fitness score of the developed Robot represents the damage taken over all rounds by wall, bullet and robot impacts throughout the generations. The maximum and average damage sustained can be seen in figure.5 below. It can be seen from the diagram that the local optimum is being reached quickly. This test was without mutation introduced into the population

**Figure.5** Round damage taken throughout generations



**Figure.6** Round damage taken vs Tracker bot

### 



The next experiment displayed in figure 6 shows the results of the same test but with a mutation chance of 5%. The proportionate percentage of mutation within a population is generally quite low.

# Conclusion

By scoring the phenotypes with incremented/decremented attributes on event actions a local optimum was reached quite quickly. Creating a new random set of behaviours on each event could off increased the natural evolutionary of the Generic Algorithm.

Applying mutation increased the time it took for the evolved robot to reach a local optimum increasing its evolutionary aspect. The mutation has injected more diversity into the population combating elitist genes.

With the knowledge I have now of Genetic programming and its code structure I would of implemented the Genotype and Phenotypes in tree structure as nearer to the end of this assignment token classes where developed to help manage crossover and mutation. More testing could be done with more time to spare too demonstrate how the evolved robot tactics who measure against other simple robocode bots.

# References

[1] A Beginning guide to Evolutionary Algorithms. Available at: <http://www.perlmonks.org/?node_id=298877>

[2] Comparison study of Genetic and Evolutionary algorithms. Available at: <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=1380654&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D1380654>

[3] Robocode API. Available at: <http://robocode.sourceforge.net/>

[4] A.E. Eiben, J.E.Smith., *“Introduction to Evolutionary Computing”*. Springer, Natural Computing Series.

[5] Genetic Algorithms: <http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html#introduction>