Evolutionary Computation

Harry Lynas

Contents

[1. Introduction 3](#_Toc415201139)

[2. Robot Development 3](#_Toc415201140)

[2.1. Robot 1 – Learning State Machine (Prototype – not a true evolutionary robot) 3](#_Toc415201141)

[2.1.1. Design 3](#_Toc415201142)

[2.1.2. Performance 3](#_Toc415201143)

[2.1.3. Discussion 4](#_Toc415201144)

[2.2. Robot 2 – Genetic Algorithm 4](#_Toc415201145)

[2.2.1. Design 4](#_Toc415201146)

[2.2.2. Performance 5](#_Toc415201147)

[2.2.3. Discussion 7](#_Toc415201148)

[2.3. Robot 3 – Genetic Algorithm 8](#_Toc415201149)

[2.3.1. Design 8](#_Toc415201150)

[2.3.2. Performance 9](#_Toc415201151)

[2.3.3. Discussion 10](#_Toc415201152)

[3. Further Robot Comparison 11](#_Toc415201153)

[4. Conclusions 11](#_Toc415201154)

[5. References 11](#_Toc415201155)

# Introduction

“Evolutionary Computing is the collective name for a range of problem-solving techniques based on principles of biological evolution, such as natural selection and genetic inheritance.”[1] It gives the ability to learn the environment that the program runs in and thus become better over many iterations.

This report describes how Evolutionary Computing can be applied to Robocode – “a programming game where the goal is to develop a robot battle tank to battle against other tanks”[2]. The objective subsequently being to be able to create robots that can learn their opponents behaviours and the environment to achieve a high win rate.

# Robot Development

## Robot 1 – Learning State Machine (Prototype – not a true evolutionary robot)

### Design

The initial concept for the first robot was to use a learning state machine that had many sets of predefined behaviour and learnt which ones were the best to use at any given period of time. A simplistic model of this can be seen in Figure 1.

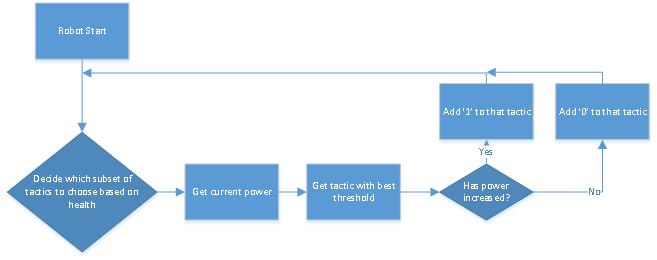


Figure . The Learning State System Design

This robot would pick a tactic, run it for a certain number of ticks, and then determine how much damage has been done compared to health lost (the fitness function). This score would be added to that tactic (if negative, reducing it) and then a new tactic would be picked.

### Performance

Classes were created containing predefined behaviour. Each class was initiated with a score of 100. Every 100 ticks the score was added to the current tactic. The score was determined to be the health change plus by the damage done.

Running this over 1000 rounds against the sample Tracker robot resulted in the state machine robot doing very poorly due to all the predefined tactics being quite bad. This can be seen in Figure 2 where all the scores have become negative from where they keep performing badly every time they are picked.

|  |  |
| --- | --- |
| uk.ac.reading.pm002501.tactics.SuperDefensive\_0 | -4742.0674417650125 |
| uk.ac.reading.pm002501.tactics.Defensive\_0 | -4336.227510873938 |
| uk.ac.reading.pm002501.tactics.SuperAggressive\_0 | -14730.207709275965 |
| uk.ac.reading.pm002501.tactics.Aggressive\_0 | -7937.199519060238 |
| uk.ac.reading.pm002501.tactics.Aggressive\_1 | -7800.650424194376 |
| uk.ac.reading.pm002501.tactics.Aggressive\_2 | -7748.602382824245 |

Figure . The results for each tactic over 1000 rounds against Tracker

### Discussion

This robot is not a true evolutionary robot as it merely learns what tactics are the best ones out of a predefined set and does not evolve new behaviours. Furthermore once one tactic gets good, it quickly has a snowball effect that causes it to always be picked due to it always getting better. Mutations would need to be implemented so that it changed behaviours more often.

Overall this design was merely the initial thoughts on how a robot can learn. Subsequent robots use true evolutionary techniques and develop new behaviours.

## Robot 2 – Genetic Algorithm

### Design

The second robot developed was based on the genetic algorithm concept. The behaviours are represented by a list of token objects which contain a behaviour and a power, as shown in Figure 3. This is different to a binary string but can still be treated in a very similar manner. There are four sequences of tokens each representing a different behaviour, also shown in Figure 4. The normal behaviour runs constantly until a different event behaviour is triggered, which causes that sequence to run immediately and fully, before resuming normal behaviour. The tokens are serialized in integer form as each enum can be mapped to an integer, and each collection of behaviours is comma separated. An example serialized robot genotype can be seen in Figure 5, which defines all the behaviours.

|  |  |
| --- | --- |
| **Token Class** | |
| **Enum**: Behaviour (e.g: FIRE\_GUN) | **Integer**: Power (e.g: 1) |

|  |  |  |
| --- | --- | --- |
| **Collection** | **Size** | **Description** |
| Normal Behaviours | 10 | Contains all behaviours that happen normally when no event is being handled. |
| OnScan | 3 | Behaviours that happen immediately if a robot is scanned by the radar. |
| OnHit | 3 | Behaviours that happen immediately if the robot hits another robot with a bullet. |
| OnTake | 3 | Behaviours that happen immediately if the robot is hit by a bullet. |

Figure . The Token Class Contents

Figure . The Collections a Robot Has

|  |  |
| --- | --- |
| Full Genotype | 00-18-03-93-06-93-06-74-04-01-06-69-05-06-06-87-04-55-00-03,03-51-05-10-02-80,06-08-05-63-00-32,02-42-01-95-00-78,060-013-104-106-244-046-038-211-085-046-043-054-231-057-208 |
| Translated Normal Behaviours | Move Left (18), Forwards (93), Turret Right (93), Turret Right (74), Move Back (1), Turret Right (69), Turret left (6), Turret Right (87), Move Back (55), Move Left (3) |
| Translated OnScan Behaviours | Forwards (51), Turret Left (10), Fire Gun (2.40) |
| Translated OnHit Behaviours | Turret Right (8), Turret Left (63), Move Left (32) |
| Translated onTake Behaviours | Fire Gun (1.26), Move Right (95), Move Left (78) |

Figure . A Full Example Genotype

Initially 100 genotypes are generated at random. Each genotype is tested against a robot, and the score for each genotype saved. The score is determined by the fitness function – that being the damage done plus by the health remaining at the end of the round. After all 100 genotypes have been tested, 100 children are created from this these based on the scores. N-point crossover and fixed-rate mutations are used to create the children. Genotypes that got a higher score are much more likely to be selected as parents, using a roulette wheel based selection. Once all children have been created the parents are deleted and the testing restarts on the new children. This means 1 generation is 100 rounds. Each and every token has a chance to be mutated during the creation of the children.

Not mentioned in Figure 5 is that the colours of the robot are also stored in the genotype, and this can mutate. This allows for the visual inspection of different generations of robot evolving and which children are then kept. It is purely aesthetic. It can be seen as the last section of the genotype with all the three digit integers, each representing a red, green, or blue value.

### Performance

Tracker is a robot distributed with Robocode that moves towards a detected robot and then fires at them repeatedly until they are dead. This is quite simple behaviour and is subsequently theoretically easy to learn how to fight against. With the fitness condition being health remaining at the end of the round plus by damage done, a combination of tactics that allow the evolutionary robot to survive and win is expected to be seen.

In the first performance test, a mutation chance of 10% is used. Pit against the Tracker robot, the minimum, maximum, and mean finesses for each generation are shown in Figure 6.

Figure . Robot 2 vs TrackFire, mutation chance 10%

It can be noted how after ~50 generations the evolutionary robot has learned a most optimal solution and is not varying from that tactic much. The minimum score is almost always zero because at least one child picked and potentially mutated is not able to land a hit on the Tracker tank. It can be noted that this may be because the evolutionary robot spawns randomly near the edge of the map preventing it from succeeding with its learnt tactics which are usually in more open space. This has quite a high chance of happening considering there are 100 genotypes in each generation.

It was tested to see if a higher mutation rate would cause more variations in the robots and thus have a higher chance of getting a better tactic. This can be seen in Figure 7. There was a lot more variation in the maximum fitness score between each generation, however the overall general fitness stayed the same. A lower mutation chance was then tested, as shown in Figure 8, and it can be observed how the opposite happens to that of Figure 7.

Figure 7. Robot 2 vs Tracker, mutation chance 15%

Figure 8. Robot 2 vs Tracker, mutation chance 5%

All three tests thus far have shown that the same general fitness levels are reached and that the minimum is almost always zero. To test to see if it is the mutations causing at least one robot to get a fitness score of zero, the robot was tested with a 1% mutation chance as seen in Figure 9. The results show that this is probably the case because now the minimum score is often much higher than before. It is also a lot more clear in this graph how the robots improve over time until evening out at just under 250 score.

Figure 9. Robot 2 vs Tracker, mutation chance 1%

The robot has successfully learnt a tactic to gain a high fitness score consistently versus the tracker robot. A less predictable robot is walls, since it will travel quite fast around the edge of the map shooting at the evolutionary robot when it enters the radar scan. The results from training against this at a 10% mutation rate can be seen in Figure 10.

Figure 10. Robot 2 vs Walls, mutation chance 10%

It is clear that it learns a tactic that has mixed results. Sometimes it gains a high score whereas other times it does not do that well. This is likely due to the tactic being successful being directly correlated to where the robots start at the beginning of the round having a much greater effect than versus the Tracker robot.

### Discussion

The evolutionary robot learns to beat the Tracker robot successfully and consistently obtains good results. A higher mutation rate leads to a greater variation in results, but does not lead to a better maximum score as might be expected. This is likely because many more generations will be required before a mutation that is significantly useful occurs. The low mutation rate of 1% lead to incredibly consistent results and a higher minimum fitness value. The walls robot had mixed results trying to learn due to the much greater impact the random factors have.

When using a mutation rate of 1% against the Tracker robot, the most successful tactic was extracted and analysed. The information can be seen in Figure 11. It can be observed that the normal behaviour when looking for a robot to attack is to move backwards while spinning the turret left. One turret right behaviour still exists, but the power (5) is so small it will practically do nothing.

|  |  |
| --- | --- |
| Full Genotype | 00-44-05-24-06-05-05-97-04-84-05-84-05-44-04-41-00-61-05-71,04-61-06-63-02-63,06-36-05-02-05-44,02-84-04-17-05-43,110-174-128-250-012-244-066-145-151-241-189-147-251-004-171 |
| Translated Normal Behaviours | Move Left (44), Turret Left (24), Turret Right (5), Turret Left (97), Move Back (84), Turret Left (84), Turret Left (44), Move Back (41), Move Left (61), Turret Left (71) |
| Translated OnScan Behaviours | Move Back (61), Turret Right (63), Fire Fun (1.89) |
| Translated OnHit Behaviours | Turret Right (36), Turret Left (2), Turret Left (44) |
| Translated onTake Behaviours | Fire Gun (2.52), Move Back (17), Turret Left (43) |

Figure 11. Genotype for the most successful robot against Tracker

When a robot is scanned, the robot immediately moves backwards and swings the turret back to the right (normal turret behaviour is swinging left, which will swing faster than it can scan). It then fires its gun, which will likely hit the enemy tank, and the movement backwards avoid any shots fired towards it. When the robot hits the enemy robot it does almost nothing – each turret command countering the last. This is likely keeping it locked onto the enemy. When the robot is hit by a bullet it immediately fires a retaliation bullet and moves backwards while turning its turret.

This tactic was tested over 1000 rounds. It was observed that the actual tactic appeared to behave such that the evolutionary robot rotated around the Tracker robot counter-clockwise while shooting. This was a most impressive behaviour to learn. It was also observed that if it started in a position where the wall blocked its counter-clockwise movements it would get stuck and probably lose. The huge success of the robot can be seen in Figure 12. Ram damage accounts for only 1% of the damage done because this is not taken into account in the fitness score algorithm and thus cannot become a learnt behaviour very easily. The ram damage done was probably more accidental than anything learnt.

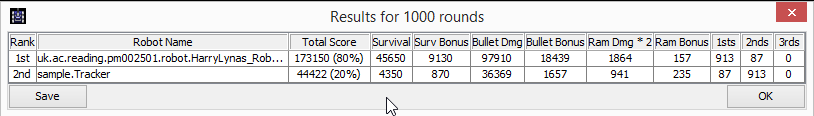


Figure 12. The result of the best tactic learnt against Tracker over 1000 rounds

## Robot 3 – Genetic Algorithm

### Design

The third robot was developed using a genetic algorithm technique where the genotype is represented with a tree data structure. The tree data structure was always traversed using depth-first in-order. The tree structure was implemented from scratch and methods were implemented to count the number of nodes in a tree and to allow the ability to swap two trees at set points. Each robot contained 4 tree data structures handling the main behaviour, the behaviour when a robot is scanned, the behaviour when the robot is hit, and the behaviour when an enemy robot is hit. Tokens were stored within each node of a tree, similarly to robot 2. However two new possible states were added to the token class: ‘If health greater than’ and ‘If health less than’. These tokens represented an if condition to check if the health of the robot is less than or greater than an amount, also stored in the token. When the tree was generated a second pass was made on it to see if any leafs contained an if condition – if they do the tree was extended so that the if condition always had behaviours. If the condition was true, it was navigated left. If it was false, it was navigated right.

The generation and selection algorithms were the same as robot 2 but adapted for the new tree representation. Furthermore the tree data structures needed to be serializable to be able to save and reread them between rounds; this was implemented by having each token stored with 4 digits and semicolons representing a null leaf.

An example tree generated can be visually inspected in Figure 13. This tree was generated with a size of 10 but due to the two if conditions (H\_GT\_N, H\_LT\_N) being generated at the end of branches, two more nodes were generated for each resulting in the final size being 14.

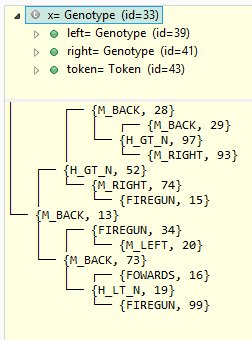


Figure . Example Genotype Behaviour Visualised

### Performance

The tracker robot was used again - the same as the tests for Robot 2. In Figure 14 the results of Robot 3 pit against Tracker with a mutation rate of 1% is shown. This mutation rate achieved the greatest success with robot 2 and it is also very successful with robot 3. It should be observed that the minimum is always quite low even with a low mutation chance, whereas with Robot 2 the minimum was often higher. The maximum fitness score is stable and at similar levels to robot 2 showing it has been able to learn how to beat Tracker successfully.

Figure 14. Robot 3 vs Tracker, mutation chance 1%

It was expected that if the mutation rate was increased that the results would be a lot more variable, however the maximum fitness will probably still be similar. The results from a test with a mutation rate of 10% can be seen in Figure 15, and the predictions can be seen to be correct.

Figure 15. Robot 3 vs Tracker, mutation chance 10%

Since a lower mutation rate was leading to greater stability in the success of the robot a mutation rate of 0% was tested, as can be observed in Figure 16. This greatly reduced the robots ability to achieve success and scores were constantly low. This is because the same tactics end up being tested repeatedly and no new behaviours are learnt.

Figure 16. Robot 3 vs Tracker, mutation chance 0%

### Discussion

Overall the robot achieved similar success rates to robot 2. The tree generation led to some more interesting behaviours, probably primarily because of the two new states that perform new behaviours based on the current health. Interestingly the robot developed very similar behaviours to that of robot 2, in that it would try to circle around the robot while shooting at it. A mutation rate of 1% seems to be currently optimal, but a higher mutation rate may lead to some new better behaviour being developed over many more generations.

# Further Robot Comparison

Using the data learnt from fighting the Tracker robot for both robots 2 and 3, they were pit against another robot written by a fellow student called Jonathan Cooper. Jon also trained his robot against Tracker. When the robots were battled they were not that efficient because the tactics they had learnt were only suitable for fighting against Tracker. They were able to kill each other rather than dying from just running out of energy, but it was clear no good tactic was being employed.

The robots were not trained against each other because it was thought that no tactic would be learnt as they kept on evolving and thus having different behaviours each generation. However on hindsight they would still have been able to learn a subset of tactics each time and thus the robot that learned quicker would in theory be the winner[3], or at least score higher with the fitness function.

# Conclusions

The state machine for the first robot although not being truly evolutionary was a valuable learning experience and demonstrates the beginnings for how to evolve behaviour without just learning the environment.

It can be noted that all the robots run fully independently by using the provided file IO classes by Robocode each round to read and store results and data as needed. This means Eclipse is not needed and it can just be run from the Robocode interface. Using this it would be easy to pit the robots against any other robots and compare them.

The implementations of the roulette wheel based selection was basic and a more advanced selection algorithm could be implemented in the future that takes into account weightings and previous selections to gain a more representative sample. This and an improved fitness function that could help it take into account more factors, such as accuracy, would help the robots achieve much greater success.

The main innovative features are the tree representation of the genotype which has an entire tree class and supportive system written from scratch to function for an evolutionary algorithm (robot 3) and both robots 2 and 3 ability to run independently and evolve quickly.

Overall both robot 2 and 3 succeeded in evolving interesting and successful behaviours against who they were pit against, however they both suffer from similar pitfalls and both achieved a similar success rate due to the similarities in the evolutionary methods.

# References

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

[2] *“Robocode”*., IBM., (20/11/2014)., URL: http://robocode.sourceforge.net/ [03/03/2015]

[3] “The Arms Race”., URL: http://evolution.berkeley.edu/evosite/evo101/IIIF1Armsrace.shtml [21/03/2015]