Multi-Environment Self-Adjusting Virtual Creatures with Nature-Inspired Genetic Algorithm

Team Beta

660019038, 660002898, 700048709, 700033506, 700074292

12th January 2020

1. Introduction

For this project, the task was to apply a nature-inspired technique to tackle a problem from one of The Genetic and Evolutionary Computation Conferences (GECCO) from the last decade. The problem chosen was the Virtual Creatures Competition from the 2019 conference [1]. This was carried out using a Python interface for Open Dynamics Engine called Python Robot Simulator (pyrosim), which was designed by members of the MEC-Lab at The University of Vermont [2]. The challenge was to evolve virtual creatures with a focus on creativity and perceptual animacy rather than performing a single task better than another system. With that in mind, our project aims to evolve quadruped creatures that can tackle multiple tasks and move with a lifelike motion. The virtual creatures are each evaluated in 4 different environments, each with a different task; walking, carrying, climbing hurdles and stairs.

2. Research

The first step in creating virtual creatures was to complete a series of lessons, recommended by the competition organisers [3]. These lessons on evolutionary robotics covered how the simulation software worked and how to create and evaluate the performance of the creatures [4]. During these lessons we tried evolving the virtual creatures using random search, hill climber and genetic algorithm methods. Initially distance alone was used to compute fitness, but we also built a creature capable of phototaxis so distance towards a light source could also be used as a measure of fitness.

At this point it was decided to continue with the genetic algorithm as this produced better results compared to the other approaches such as random search and hill climber methods [5]. This is because the random search method begins each iteration with a clean sheet, whilst the hill climber and genetic algorithm build from the best versions of previous iterations. The genetic algorithm is more powerful than the hillclimber method as the hillclimber method is easily stuck at a local maxima, as it can only “climb” one hill. We did try a parallel hill climber which created a population of many individual solutions giving it the ability to “climb” many hills, meaning higher fitnesses can be found and the local maxima hopefully avoided. The genetic algorithm added to this by using a selection criteria for parents producing offspring so that more fit parents were more likely to produce children. This means full use of the population and aids the algorithm in avoiding local maxima. Various structures of genetic algorithms were researched and it was decided to build a similar structure to the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [6], to improve upon the very simple one initially implemented. We added elitism, crossover and mutation and tournament selection and also a form of weakest replacement.

3. Tasks

The initial lessons allowed for each group member to have the same understanding of the simulation and to create an identical base code, allowing for easier integration of individual tasks completed later. The work to be carried out in developing and improving the project was then

divided, with each member selecting a different area to focus on. The areas decided on included investigating the effects of gravity on the virtual creature, creating different environments with various obstacles for the virtual creature to navigate, testing the ability of the virtual creature to carry objects and finally, the optimisation of the algorithm itself. After sharing our progress and planning our final algorithm design we then split the tasks into experimenting and results collection and research and report writing. Once the results were collecting, everyone contributed to finalising the group report. It should be noted we met nearly everyday to share our progress and discuss ideas or problems that we were encountering. The meeting minutes clearly record who attended and what was discussed. The active team members worked very well together to ensure everyone took on a range of tasks whilst still working with people's individual strengths.

4. Algorithm

We considered a range of algorithms to solve our task and our final choice is summarised in the flowchart.

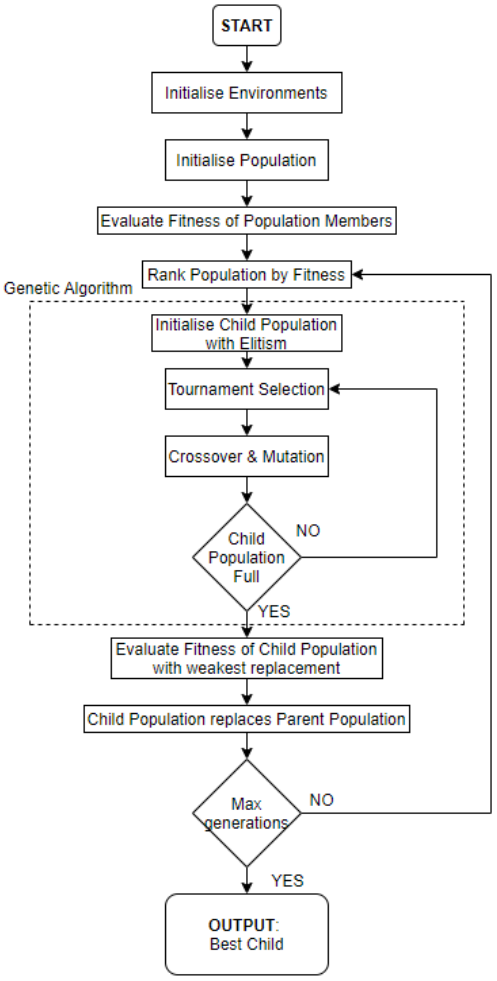


Figure 1: A flowchart of our final generational elitist algorithm.

To initialise, the four environments are created and then the first population is generated. Each individual member of the population is represented by a genome which consists of two arrays of integers with array shape 4x6 and 6x8. These genome arrays are initiated randomly (ranging from -1 to 1) and used as the weights in the neural network that controls the virtual creature. The fitness of the population members is then found by running the simulator and calculating the average fitness across every environment. When the virtual creature was not carrying an object, its fitness was the

distance travelled in the y-direction and when the virtual creature was carrying an object, its fitness was an average of how far both the object and the virtual creature travelled in the y-direction. The environment images with labelled directions are shown in the Appendix. The structure of the neural network is shown below. The weights are updated during the simulation.

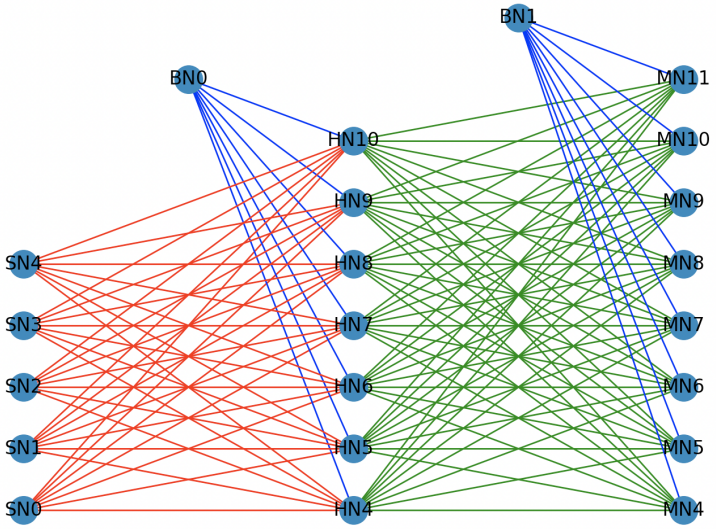


Figure 2: The neural network used within the simulator to update weights used as the genome representation for each individual in the genetic algorithm. SN: sensor neurons, HN: hidden neurons, MN: motor neurons, BN: bias neurons. The sensor neurons take an input from touch sensors on each limb of the virtual creature and the motor neurons output to the joints (two per limb) of the virtual creature.

The next part of the algorithm evolves the population. This occurs by initialising an empty child population and using elitism to select the best parent from the parent population to pass to the child population. The rest of the child population is then filled using tournament selection with crossover and mutation techniques. Two tournaments occur to select each of the two parents and then crossover is carried out. This happens at a random point limited to half the length of the genome, to produce two different children. The best virtual creature from these four is passed through to the next generation and stored in the child population. If a child with an identical fitness value already exists then mutation occurs before the individual is stored. Mutation is based on a predetermined maximum mutation rate, the actual value can fall between one and this value, and once the genome elements to be mutated have been randomly selected then a normal distribution centered on the original array value is used to mutate the value. This whole process of tournament selection, crossover and mutation is then repeated until the child population is full. After this, the fitness of the child population is evaluated, with the weakest member being mutated before the child population replaces the parent population for the new generation. The cycle continues until the maximum number of generations is reached. Finally, the program opens a graphical user interface, GUI, which shows the movements of the virtual creature with the highest fitness over the four different environments; the final fitness value is also displayed.

We chose to base our algorithm on the NSGA-II algorithm because it is an elitist multi-objective genetic algorithm (MOGA), these now dominate the field of MOGAs as they execute and converge faster [7]. NSGA-II in particular does not require extra parameters and is widely used, as it simultaneously optimises each objective without being dominated by any particular solution [8]. Whilst other MOGAs, including PAES [9] and NPGA [10] were looked at, literature confirmed that NSGA-II outperformed them [7].

5. Results

The competition deliverables are a video of the virtual creature in action and a one page description of the methods used (a condensed version of this report). The video has been handed in alongside this assignment.

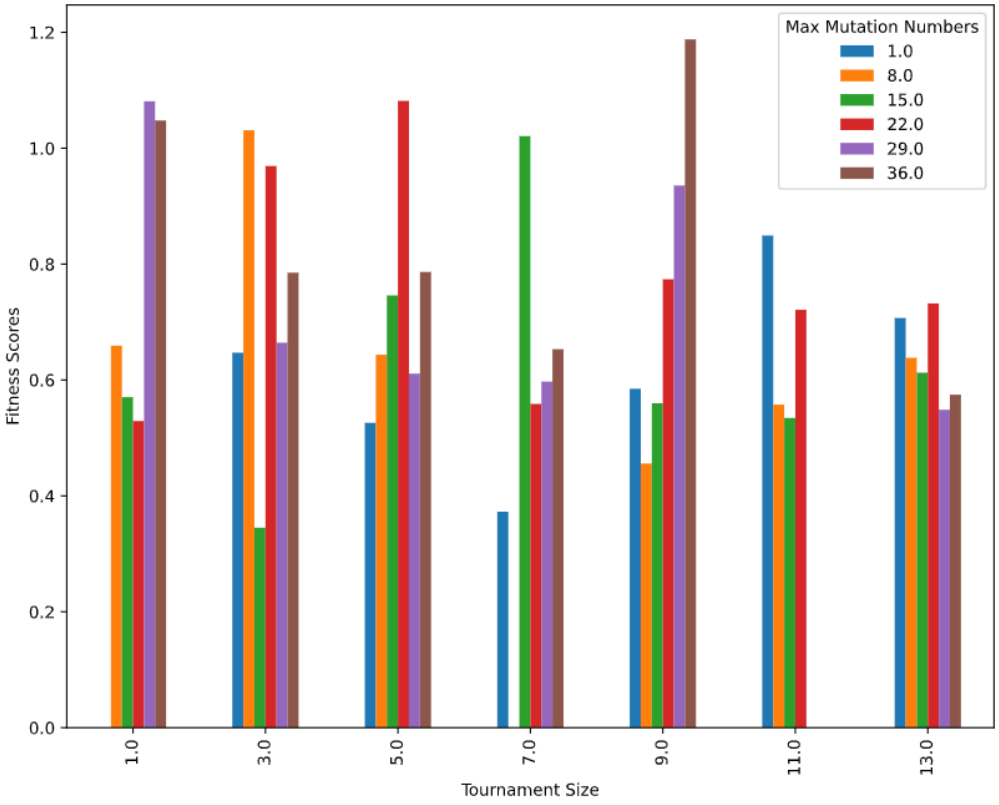


Figure 3: The final results to show the fitness as mutation rate and tournament size vary using the optimal population size of 20. Full results in the appendix.

Our results show that the algorithm was able to converge when the maximum number of generations was set to 300. This was originally set to 200 but needed to be increased. We hand tuned the following parameters; population size, mutation rate and tournament size. We found the best values to be 20, 36 and 9 respectively. The virtual creature performs to a high standard in the first two environments as it excels at walking and carrying but it struggles with climbing over the hurdles and up the stairs. In some cases it was able to climb the stairs but it always became stuck after the first hurdle. This was also the case in our early experiments, the virtual creature was easy to train to walk and show animal-like behaviour in a single environment but it was much harder to get it to perform well at many tasks. This is most likely due to the creature's design not being optimal to the last two environments in terms of leg and body design.

6. Conclusion

Our algorithm has succeeded in evolving virtual creatures that can mimic animacy and perform a variety of tasks to a reasonable standard. As advised by the competition facilitators, it is very hard to optimise one virtual creature in many environments but we believe our algorithm makes a good attempt.

References

[1] GECCO 2019 | Virtual Creatures Competition [Internet]. GECCO 2019. 2019 [cited 1 January 2021]. Available from: https://gecco-2019.sigevo.org/index.html/Competitions#id_Virtual%20Creatures%20Competition

[2] Pyrosim 0.1.1 Documentation [Internet]. Ccappelle.github.io. 2017 [cited 1 January 2021]. Available from: https://ccappelle.github.io/pyrosim/#

[3] The Virtual Creatures Competition [Internet]. Virtualcreatures.github.io. 2019 [cited 2 January 2021]. Available from: https://virtualcreatures.github.io/

[4] Education in Evolutionary Robotics [Internet]. Reddit.com. 2019 [cited 2 January 2021]. Available from: https://www.reddit.com/r/ludobots/wiki/pyrosim/simulation

[5] Mitchell M, Holland JH, Forrest S. When will a genetic algorithm outperform hill climbing. InAdvances in neural information processing systems 1994 (pp. 51-58).

[6] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. 2002;6(2):182-197.

[7] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation. 2000 Jun;8(2):173-95.

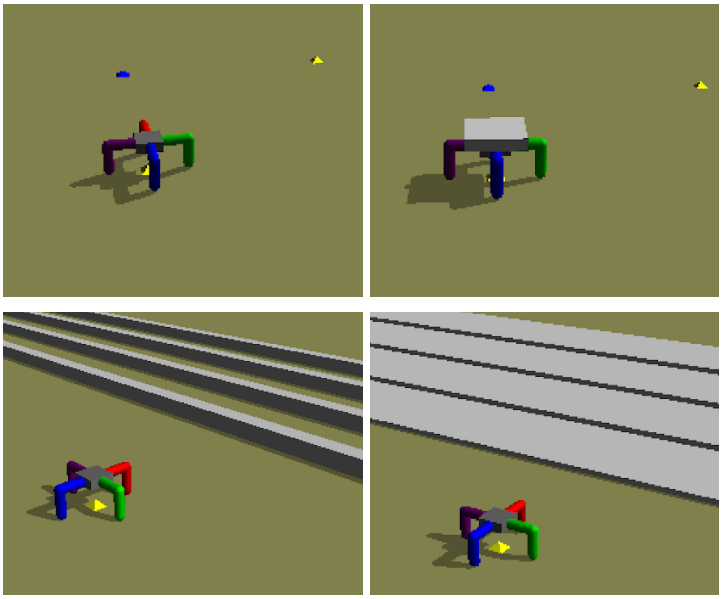
[8] Yusoff Y, Ngadiman MS, Zain AM. Overview of NSGA-II for optimizing machining process parameters. Procedia Engineering. 2011 Jan 1;15:3978-83.

[9] Knowles J, Corne D. The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406) 1999 Jul 6 (Vol. 1, pp. 98-105). IEEE.

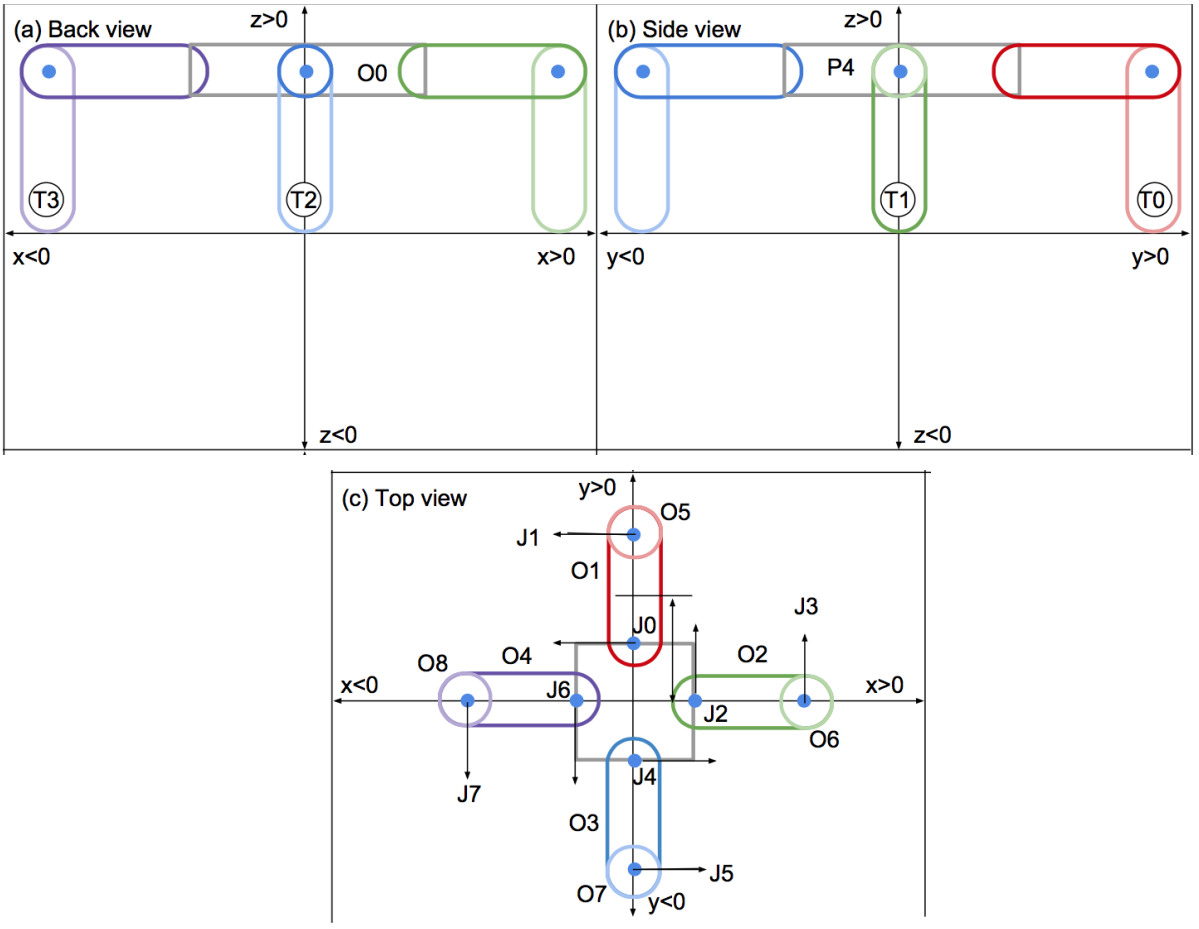
[10] Horn J, Nafpliotis N, Goldberg DE. A niched Pareto genetic algorithm for multiobjective optimization. Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence 1994 Jun 27 (pp. 82-87). IEEE.

Appendix

The four environments (walking, carrying, hurdles, stairs):



The quadruped virtual creature and x, y, z-directions [4]:



Our full results will all 4 population sizes that we tested: