Final Assessment COMP5400

Student ID: 201373470

Genetic Programming

1. Introduction:

Genetic programming (GP) is a systematic method for getting computers to automatically solve a problem starting from a high-level statement of what needs to be done. When using genetic, people even don't need to pre-select model structure, like a simple linear model or complex neural network. On the contrary, people only need to specify operators and the input variable of the model. Genetic programming will automatically generate, select and optimize a mathematical model through evolutionary ideas to make it perform optimally on the objective function we give.
In short, GP is a special form of evolutionary computation, which can evolve a math expression or, more generally, a computer program to optimize some measure.
Another characteristic of GP is that it's a domain-independent problem-solving approach in which computer programs are evolved to solve a problem. Therefore, the genetic algorithm is very suitable for the application field, where the current human knowledge is very little. And GP is also extensively useful in many different areas like automatic generate of model structures for circuit design and symbolic regression for data analysis and so on.
The whole idea of GP is based on the Darwinian principle of reproduction and survival. The first proposal to evolve program is that of Alan Turing in 1950. In the first Genetic Algorithms conference in Pittsburgh that Michael Cramer published the first statement of modern "tree-based" genetic programming. After, John Koza, who patented his invention of a genetic algorithm and owned more than 205 publications on GP, started the annual GP conference in 1996. Since then, many GP papers continue to be published at a diversity of conferences and associated journals. GP continued to flourish, and have been used in predictive modelling, data mining, financial and other different fields.

2. Relationship with Genetic Algorithms:

Genetic algorithms (GA) and genetic programming (GP) are often considered as separate but related fields.
They have something in commons. First of all, they both have the genetic term in their name, which represent they both adapt to the Darwinian principle of reproduction and survival. Some of the critical elements of this principle must be considered in GA and GP, like genotype, phenotype, fitness function, crossover and mutation. Some features of Darwinian principle also has been well implemented in both GA and GP. For example, the variation caused by different crossover and mutation, the selection according to fitness function and heredity relates to crossover. All in all, GA and GP are developed based on the same principle or idea.
However, if we are looking into details, there are some differences between GA and GP.

First of all, it's their representation. Standard GA uses a fixed-length linear binary representation, whereas GP uses a variable size tree representation. In GA strategy, it focuses on a bit string and create more string by crossover and mutation. In GP, the initial is a random composition of the computer programs. And it can create new populations of computer programs by crossover, which is recombining parts of two existing programs and mutating a randomly chosen part of the program.
From different sort of data structure, we can tell that their form of genotype must be different as well. For GA it's made of a set of bit strings. And in GP, the genotype is the space of trees. In other words, the standard tree-based form or someone calls it as Koza-style GP.
After list all of the commons and differences above, I think GP is the upgrade or pro or ultimate version of GA, but they still share the same principle. GP has variable size representation, which means it would be more suitable for those problems that it's challenging to prespecify the size of the solution. And this could be one of the upgrade things compare to GA. However, the computer program is essentially bit binary strings rather than trees. So the representation of GA probably is more suitable for a computer to process.
In conclusion, GA and GP are related as they both inspired by Darwinian evolution, and they are regarded as separate fields. As John R.Woodward states that "while accurate classification is important, over fragmentation can mean that related areas become separated, and this will have a detrimental effect on the progress of both field". Thus, we should discover each potential advantages of GA and GP in a specific scenario and develop better applications or solutions in many fields.

3. Recent application based on GP:

GP research topics and applications are diverse, including predictive modelling, image processing, industrial design and so forth. As a gamer and a big fan of Github, I found an exciting and impressive open-source project on Github, which is playing flappy bird by using genetic programming.
In this case, we use the Cartesian genetic programming (CGP), which is the GP based on the graph structure. It was came up by Julian F.Miller in the University of York in 2000. In short, CGP uses a graph to encode a program. In most application scenarios, this program is a mathematical formula, which we call symbolic regression.
First of all, we should determine three inputs, which are horizontal distance h, vertical distance v and the pipe gap g. And three inputs are shown in Figure 1 below.

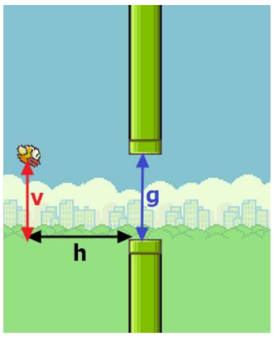


Figure 1: Represent the three inputs: v, h and g in the game.

Then come up with a function that we want to evolve, which is y = f(v, h, g), and f is used to control the flappy bird movements. If y > 0, then the bird wave it's wing so that it will go up. Else, no action for the bird. We are not going into the detail of f function, because we want the f function to generate or evolve by GP.

The objective is to find f to make the output y = f(v, h, g) control the birds fly as far as possible.

Set up for function set, including arithmetic operators like +, -, *, / and neg (return negated number).

Set the graph with 1 row and 500 computational nodes.

Set the weights between nodes as a random number between -1 and +1.

Then come up with the graph (Figure 2). We don't want to calculate this expression according to the laws of physics or something else. Because this only needs to automatically find a formula, which is the fixed inputs variables with a different combination of arithmetic operators. Or we can say more broadly, to explore a program and build a model with an unknown structure.

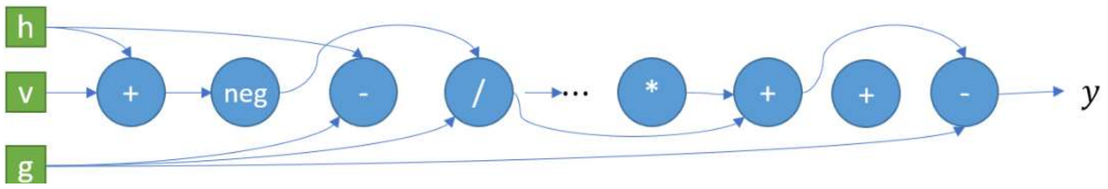
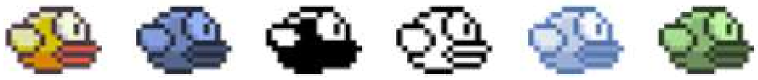


Figure 2: CGP graph

The fitness function can be defined by the distance the bird has flown.

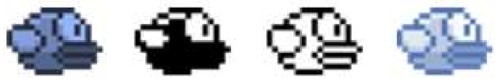
Then we going to GP evolution step:

1. GP generate some random solution (individual). Six flappy birds with a different f function.

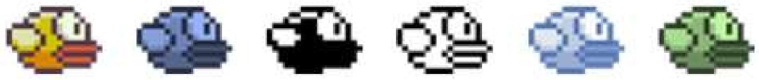


2. Decode the current individual and solve the target value (evaluation), and selection according to fitness score. Select the top 2 individuals with the best score.

3. Through crossover and mutation, the selected individuals with a certain probability to get four offspring.



4. Put them together with their parents. Try the game again; assign fitness to each individual. We are then going back to step 2 until the optimization target value meets our requirements.



Would the solution have been feasible without GP? The answer is yes. I also found many other ways of playing flappy bird. For example, using neural network and genetic algorithm. Its main idea is to use evolutionary algorithms such as a genetic algorithm to train artificial neural networks. Others are using reinforcement learning, fuzzy control, Q-learning, and many other ways to play flappy bird.

There are few things differents or characteristics for CGP when it compares with the neural network.

1. The structure is variable by randomly changing the connection between computing nodes, which is the mutation. The graph of CGP will continuously change with the iterations.
2. A computing node can be connected to any node before it. In CGP, it describes the connection relationship between two computing nodes, not the connection relationship between layers. Therefore, the topological relationship of computing units in CGP is more flexible.
3. The weights between the two computing nodes are randomly generated instead of optimized by backpropagation.

There are many other solutions to play flappy bird, but one CGP network can encode almost infinite models from simple to complex to solve different problems as well.

4. Conclusion:

Genetic programming belongs to evolutionary computation and has a strong relationship with genetic algorithm but different. The major difference is GP uses variable-length trees as its representation. The Major common is that they are both inspired by Darwinian evolution. Then I briefly state one interesting application based on GP, which is on video game field. I introduce Cartesian genetic programming, which is a form of GP. And compare it with other solutions for the specific, popular, childhood game, flappy bird.

5. Reference:

1. Gao, S. (2020). *ShuhuaGao/gpFlappyBird*. [online] GitHub. Available at: https://github.com/ShuhuaGao/gpFlappyBird [Accessed 29 May 2020].

2. Keller, R. and Banzhaf, W. (2001). *Genetic Programming using Genotype-Phenotype Mapping from Linear Genomes into Linear Phenotypes*.

3. Koza, J. (1992). *Genetic Programming*.

4. Lin, Y.-C. (2020). *yenchenlin/DeepLearningFlappyBird*. [online] GitHub. Available at: https://github.com/yenchenlin/DeepLearningFlappyBird [Accessed 29 May 2020].

5. Miller, J. (2000). *HOME*. [online] Mysite. Available at: https://www.cartesiangp.com/ [Accessed 29 May 2020].

6. Miller, J. (2014). *Cartesian Genetic Programming*. [online] Available at: http://cs.ijs.si/ppsn2014/files/slides/ppsn2014-tutorial3-miller.pdf [Accessed 29 May 2020].

7. Poli, R., Langdon, W., Mcphee, N. and Koza, J. (2008). *A Field Guide to Genetic Programming*. [online] Available at: http://libros.metabiblioteca.org:8080/bitstream/001/184/4/978-1-4092-0073-4.pdf.

8. Susnic, S. (2020). *ssusnic/Machine-Learning-Flappy-Bird*. [online] GitHub. Available at: https://github.com/ssusnic/Machine-Learning-Flappy-Bird [Accessed 29 May 2020].

9. Wikipedia. (2020). *Genetic programming*. [online] Available at: https://en.wikipedia.org/wiki/Genetic_programming.

10. Woodward, J. (2003). *GA or GP? That is not the question.*