Genetic Algorithm

Project Plan

Hameed Roleola

CS4822 - MSCI Project

Supervised By: Dr Eduard Eiben
Royal Holloway, University of London

Abstract

One of the most popular techniques for evolutionary computation is genetic algorithms [6]. A genetic algorithm (GA) [1] is a type of metaheuristic inspired by natural selection. Genetic algorithms are employed to develop high-quality solutions to optimization and search issues. Optimization of decision trees for improved performance, solving sudoku puzzles, and hyperparameter optimization are some examples of GA applications. Researchers in evolutionary computation believe that the mechanisms of evolution are well suited for solving some of today's most difficult computational issues [5]. Many computational issues require looking through a vast array of potential solutions. From this we can see that the application of GA's can truly benefit and help to solve real world problems and ordinarily difficult to solve computational issues.

GA's give a useful insight into how software can be used to improve real world problems and solve complex issues. These problems can often be defined as constraint satisfaction problems. Constraint satisfaction problems are mathematical questions designed as a set of objects whose state must satisfy a number of constraints or limitations. Our objective is a set of constraints on these variables, and our goal is to define a problem as a set of variables [7]. A range of examples that can be viewed as a CSP is the Travelling salesman problem, n-queens game, and graph-coloring. Real world applications of this include optimize travel, vehicle routing and astronomy to help determine the movement of a telescope for the shortest distance between different stars [2] [4]. So, it is possible to see the real-world applications of genetic algorithms and how they can be used in different facets to help solve problems. There are many problems that need to be optimized but the method of optimization may incur costs. This is where genetic algorithms come into place as they can be used on problems through simulation without any costs or real-world implications. Since a-level I have been interested in genetic algorithms, leading me to create a simple program for my end of year project to do with how technology i.e., software can be used in the real world removing unnecessary costs from a biological perspective involving animals and survival. From this I became increasingly interested in genetic algorithms and their application on the real-world.

Studying biology at A-level I wanted to see how the process of evolution can be integrated with computational processes and tools to help optimize and improve problems from a different perspective. With this project I aim to have developed an application capable of solving or improving real-world situations from a wider approach. From this project I hope to gain a deep understanding of genetic algorithms, how they are implemented and their use of integration with evolutionary processes. Whilst also learning how genetic algorithms can be applied to multiple real-world problems looking specifically at their effectiveness and results of this. To be more specific I am going to begin by working on a singular constraint optimization problem, then with a firm understanding to apply this in a much broader sense. If possible, I also aim to extend the project by comparing different optimization techniques and their effectiveness on solving CSP's.

Timeline

My plan will be for the first term to look at constraint optimization and then apply it to one problem in this case being the TSP. Then to go with a wider approach to look at a ga's implementation on a wider scope.

Term 1

Week 2

- Create 2d prototype sketches for the proof of concept.
- Create layout of report
- Installation of necessary software and Ide to produce program, algorithm, and GUI. So far likely python and Tkinter.
- Experiment with creating GUI with sublime text

Week 3

- Begin research on Genetic algorithms and write the basis of the report.
- Draft template for GUI
- Begin implementing the GUI - a simple version

Week 4

- Research the implementation of Genetic algorithm
- Create a simple genetic algorithm

Week 5

- Work more on building genetic algorithm for TSP
- Write report on encoding various problems for GA's

Week 6

- Try to use Tkinter with genetic algorithm
- Begin research on Design patterns for write up e.g., MVC Design Pattern

Week 7/8

- Animating and simulations for genetic algorithms
- Write up on the theory of coalescence and genetic drift

Week 9

- Link GUI and Genetic algorithm together
- Write up remaining parts of the report to a good standard

Week 10/11

- Finalise TSP problem for project
- Finalise report

Term 2

I expect by term 2 that a lot of the project plan will change due to the size of the project, so I have purposely made this section more loose but still with detail.

Week 1/2

- Research constraint optimisation in a broad sense and how it can be further extended

- Start to describe the software engineering processes involved

Week 3
- Write a survey of optimisation methods (carryout survey)

Week 4/5
- Investigate other problems for optimisation problem
- Compare different optimisation methods applied to the same problems

Week 6
- Implement the broader Constraint optimisation aspect of the project
- Look to add other optimisation methods

Week 7/9
- Continue to integrate GUI with genetic algorithm
- Review and write up on project results

Week 10/11
- Prepare for the final report and presentation

Risks and Mitigations

There are inherent risks with every project, some of which cannot be avoided. My project is no different; while controllable, it may carry particular risks. These are some risks and how they might be mitigated.

1. As the project is large it can be prone to bad parameter settings. This could be caused by the sheer amount of parameters having to be made and can result in very slow progress and quite possibly the algorithm being stuck. Which can delay the project as a whole. Setting meaningful names for parameters and using global variables can help to avoid this issue.

2. Another problem that can be developed is overcrowding, so if the population density is too high or if different individuals have the same location/genes, diversity is lost. If the mutation rate is too low or if many copies of the same individual can be preserved in the following generation, overcrowding will most likely develop. This is why the mutation aspect of the project must be conducted properly and avoid the possibility of a low mutation rate.

3. Using genetic algorithms can be quite time consuming, particular depending on the amount of data being used. Due to a lack of robust code, experiments may be carried out that subsequently add minimal effect to what has been done previously. To prevent these problems, I will use industry-recognized software engineering methods and Test-driven Development to write all of the code for this project. In order to avoid wasting time, I will also regularly assess the outcomes of my code and make necessary adjustments to my program.

4. The complexity of genetic algorithms does not scale well. That is, the size of the search space frequently grows exponentially in regions where the number of elements subject to mutation is high. This makes applying the technique to issues like home design incredibly challenging. Such issues must be reduced to the most basic representation in order to be tractable by evolutionary search. Therefore, rather than encoding precise construction designs, evolutionary algorithms are frequently used to encode ideas for building shapes. To circumvent this, we must regulate the size of the search space and the complexity.

5. Artificial genetic algorithms frequently have a segment that is the most constrained: repeated fitness function evaluation for complex tasks. Complex problems frequently call for highly costly fitness function analyses in order to find the best solution. In practical issues like structural optimization problems, evaluating a single function may take many hours to several days of exhaustive simulation. Such problems cannot be solved by standard optimization methods. It might be required in this situation to forego an exact evaluation in favor of a more computationally effective approximation.

Acronyms

CPS – Constraint Satisfaction Problem

GA – Genetic Algorithm

TDD - Test-driven Development

TSP - Travelling Salesman Problem

Bibliography

[1] En.wikipedia.org. 2022. Genetic algorithm - Wikipedia. [online] Available at: <https://en.wikipedia.org/wiki/Genetic_algorithm> [Accessed 22 September 2022].

[2] Route Optimization Blog. 2022. What is the Traveling Salesman Problem (TSP)?. [online] Available at: <https://blog.route4me.com/traveling-salesman-problem/> [22 September 2022].

[3] Cs.mcgill.ca. 2022. [online] Available at: <https://www.cs.mcgill.ca/~dprecup/courses/AI/Lectures/ai-lecture05.pdf> [Accessed 27 September 2022].

[4] Goldberg, D., 1989. Genetic algorithms in search, optimization, and machine learning [Accessed 30 September 2022].

[5] Mitchell, M., 1989. Introduction to Genetic Algorithms [Accessed 1 October 2022].

[6] Sivanandam, S. and Deepa, S., 2010. Introduction to genetic algorithms. Berlin: Springer [Accessed 1 October 2022].

[7] Ghedira, K., 2013. Constraint Satisfaction Problems [Accessed 26 September 2022]