Module Code:	CS3AI18
Assignment Report Title:	Project Report
Student Number:	27017877
Date of Completion:	11/03/2021
Actual Time Spent (hours):	40-50
	Overall Good Assignment
Assignment Evaluation:	Workload maybe a little much
	Brief a little vague

Optimization in Genetic Algorithms

Abstract

Evolutionary algorithms like Genetic Algorithms, can produce a population, breeding it with optimized parameters until a goal is reached. With that idea in mind, they are used to solve different kinds of problems, like continuous and combinatorial, and as well as be combined with other algorithms to help them optimize. By introducing optimization components like selection fitness pressure and other stochastic operands, GAs can produce results that are very close to the global minimum or maximum.

Genetic Algorithms

Genetic Algorithms are stochastic search algorithms, with the intention of mimicking the mechanism of natural selection and genetical evolution. An implementation of a Genetic Algorithm starts with the *initialization* step where an initial population of randomly generated chromosomes. Then, the *fitness evaluation* step takes place where, using a set *objective function* the fitness of each chromosome is evaluated relative all other chromosomes of its generation. Afterwards, the generation is parsed through different *operands* (crossover-breed, mutation) designed to produce a new generation of chromosomes such that it represents a more optimal solution. Each new generation is evaluated and optimized indefinitely, or until a termination parameter is met, of which the optimal solution is one (Mathew, T.V., 2012).

1. Continuous Optimization

With the definition of a Genetic Algorithm in mind, a continuous optimization problem is one whose function variables take any value within a specified range. A function is defined $f\colon \boldsymbol{x} \rightarrow \mathbb{R}$ where, x = (x_1, x_2… x_n) is its variables within a range [*lb,ub*].

1. A. Implementation of Continuous Optimization problem.

An example of a continuous optimization function is *sum of squares*, which is expressed as:

$$f(\mathrm{x}) = \sum_{i=0}^{n} i\boldsymbol{x}_i^2$$

and whose minimum value is 0. Finding such values is called a *minimization* of this function.

Based on the above optimization function the below GA was created.

The Genetic Algorithm generates an initial *population* of *size 50*. Each *chromosome* (individual) has two genes [*x, y*] that are within range [-*10,10*]. Then, for each chromosome, their *fitness* is calculated using the *sum of squares* optimization function. Afterwards, the offspring generation selection process happens where chromosomes are selected to be reproduced, using the tournament method. During that, two chromosomes are compared based on their fitness, and the one with the least is selected (due to minimization problem). Then the crossover and mutation functions are applied to them, with rates 0.8 (first 80% of population) and 0.2 (last 20% of population), respectively. During the crossover, two parent chromosomes are selected, and their genes are swapped, generating an offspring. (child). During the mutation, each of the chromosome's genes are randomly multiplied by a random value between 0.85 (reduce by 15%) and 1.15 (increase by 15%). Therefore, the new generation is bred. Right after, each of its chromosome's fitness is evaluated, and the cycle continues until one of the chromosomes hits a fitness of 0 (global minima) or the generations surpass 100 (Gen 100 is the final). For each generation, the fittest individual is displayed and after termination, the best out of all generations is displayed.

1.B. Expansion of Fitness Function

In an optimization problem, maximizing or minimizing a function allows for comparison of the different solution choices (chromosomes). Essentially, a fitness function evaluates how distant a given solution (chromosome) is from the problem's optimal solution. For continuous variables, the fitness function plays a tremendous role in the effectiveness of the GA as they help deduce information about the fitness of an individual relative to the rest For the above GA, five significantly different optimization functions were added. The criteria of the selection were based on the function's dimensions, input domain and global minimum.

Functions added: Sum of Squares, Matyas, Hump, Three-Hump Camel, Booth, Levy

1.C. Genetic Algorithm Performance Evaluation

As GA's application band is extended, the development of an evaluation methodology becomes increasingly essential. For an algorithm to be considered good, it must be compared to other algorithms for the same problem and be selected for providing the better solution based on the criteria (Sugihara, K., 1997).

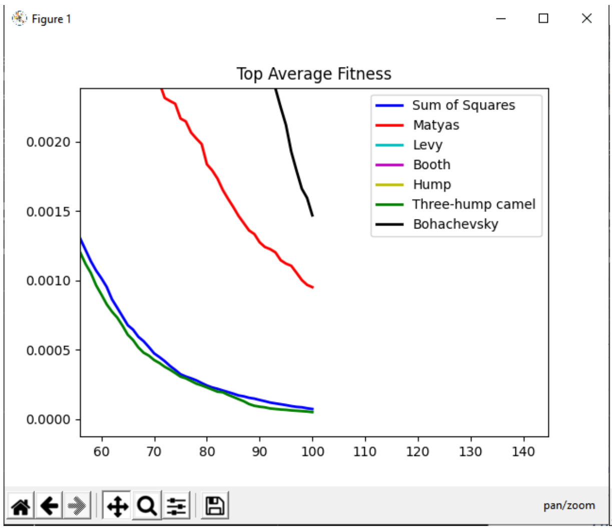
Below, the fitness functions detailed on the above section are rigorously tested. For each test section the compared hyperparameters are stated. The rest results are the average out of 100 tests.

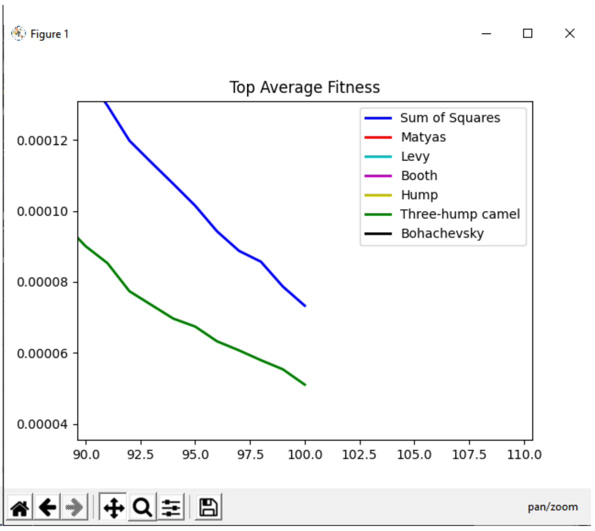
Set #1 Parameters				
Bounds	**Generations**	**Population**	**Crossover**	**Mutation**
-10,10	100	100	0.8	0.2
Set #2 Parameters				
Bounds	**Generations**	**Population**	**Crossover**	**Mutation**
-5,5	150	200	0.5	0.5
Set #3 Parameters				
Bounds	**Generations**	**Population**	**Crossover**	**Mutation**
1,10	200	100	0.7	0.3

Set #4 Parameters				
Bounds	**Generations**	**Population**	**Crossover**	**Mutation**
20,20	300	150	0.4	0.6

Average Fitness of Best Individual after 100 executions				
f(x)	Set #1	Set #2	Set #3	Set #4
Sum of Squares	6.58E-04	3.82E-07	1.05E-05	1.86E-09
Matyas	1.78E-03	1.38E-05	1.57E-04	1.29E-06
Hump	1.93E-02	2.47E-06	3.62E-02	3.21E-02
Three-Hump Camel	4.67E-02	1.75E-06	1.89E-05	3.47E-09
Booth	3.97E-02	5.75E-06	9.94E-06	7.13E-06
Levy	1.75E-02	2.35E-08	9.63E-08	8.93E-09

Graph of Population fitness distribution for the final 10 Generations (Finalist).





1.D. Genetic Algorithm Performance Evaluation

One of the staple aspects of a Genetic Algorithm is the selection procedure. Likewise, with the objective function, the selection method plays a significant role in the overall optimization of the Genetic Algorithm. An example of such, could be a selection method 'favouring' individuals with a more optimal fitness, therefore increasing their chances of getting selected for reproduction to the next generation. This helps 'skew' the fitness results further towards the optimal (Jebari, K. and Madiafi). To the same GA algorithm, three new selection methods were added.

Below, the newly added selection methods are rigorously tested. For each test section all necessary hyperparameters are stated. The rest results are the average out of 100 tests.

Total Generations Testing:

The following test will evaluate the fitness function performance based on the same parameters found in section 1.B. The genetic algorithm was tested on 50, 100 and 200 total generations.

Average Fitness of Best Individual after 100 executions				
Parameter	Set #1	Set #2	Set #3	Set #4
Random Selection	2.77E-01	3.52E-02	2.80E+00	2.80E+00
Tournament Selection	3.33E-02	2.47E-06	3.62E-02	4.54E-02
Elitism + Tournmaent	1.97E-05	1.75E-06	1.89E-05	3.47E-09

Analysis:

The testing phase was split into two. First the selection methods were run without the ELITISM parameter on, then they were run with it enabled. Overall, my observations were that by adding more selection pressure on the algorithm it allowed it to optimize even further and as a result breed individuals that were overall fitter. When elitism succeeds it produces marginally better results than any other selection method tested. On the other hand, if elitism causes selection bias, with low enough mutation rate, it could do the opposite and produce less optimal results each generation. In summary, with my observations I concluded the selection stress is directly correlated to how good the average fitness of a generation is.

2. Combinatorial Optimization

With the definition of a Genetic Algorithm in mind, a combinatorial optimization problem is one whose function variables take discrete values within range. A function is defined $f : \boldsymbol{x} \to \mathbb{R}$ where, x = (x_1, x_2… x_n) is its variables is from a discrete set {*a* ,*b*, ….}.

2. A. Implementation of Combinatorial Optimization problem.

An example of a combinatorial optimization function is *sum 1s in a binary string*, which is expressed as:

$$f(\mathrm{x}) = \sum_{i=1}^{n} xi$$

and whose minimum value is 0 (all 0’s). Finding such values is called a *minimization* of this function. Based on the above optimization function the below GA was created.

The Genetic Algorithm follows the same steps as the continuous optimization problem, trying to solve for a binary string of *n* length, where the global minimum is 0. The *fitness* of each chromosome is calculated by the missing number of *0’s* in its binary genes, relative the length of the solution. The algorithm implements a single point crossover, where the parent's binary genes are swapped, based on a randomised crossover point. The mutation is simply a random bit flip to one of the characters of the binary gene.

2. B. Different Combinatorial Optimization Problems

i. The 0-1 Knapsack

The knapsack problem implemented restricts the number $\boldsymbol{x_i}$ of copies of each kind of item, to either *0* or *1*. Given a predefined set of *n* numbers, indexed from *1* to *n,* each with a weight $\boldsymbol{w_i}$ and a value $\boldsymbol{v_i}$**,** along with a knapsack maximum capacity of $\boldsymbol{W}$**.** The problem:

$$maximize \sum_{i=1}^{n} w_i v_i \qquad\qquad subject\ to \sum_{i=1}^{n} w_i v_i \leq W\ and\ x_i \in \{0,1\}$$

ii. N Queens

The N Queens problem implemented restricts the placement of a queen $\boldsymbol{q_i}$ on a chess board $n * n$ so $x_i, y_{i,} z_i$ are only subject to a single $\boldsymbol{q_i}$

$$f(x) = \sum_{i=n}^{0} x_i + y_i + z_i$$

Such that x_i is the conflict of queen in the x axis, y_i is the conflict of queen in the y axis and z_i is the conflict of queens in the z-axis.

iii. Phrase Solver

The Phrase Solver problem implemented restricts the genes of $\boldsymbol{x_i}$ from a specified space of genes. The result must equal that of the set solution.

$$f(\mathrm{x}) = \sum_{i=1}^{n} xi$$

3. Genetic Algorithm Application

3. A. Genetic Algorithm Solution for Travelling Salesman Problem

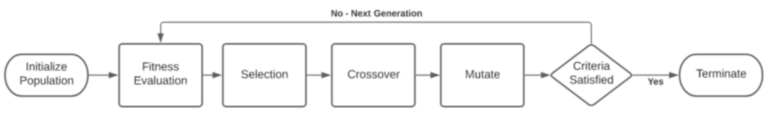
Given a connected undirected tree of nodes n and distance table of edges n^2 containing the weights between all edges, a GA algorithm is implemented to find an optimal solution to the *minimization problem*. The constraints are that there is a starting and ending node, which must be the same, the salesman must visit all nodes, and all other nodes must only be visited once.

$$
\begin{aligned}
&\min \sum_{i=1}^{n} \sum_{j\neq i, j=1}^{n} c_{ij} x_{ij} \colon && \\
&x_{ij} \in \{0,1\} && i,j = 1,\ldots,n; \\
&u_i \in \mathbf{Z} && i = 2,\ldots,n; \\
&\sum_{i=1, i\neq j}^{n} x_{ij} = 1 && j = 1,\ldots,n; \\
&\sum_{j=1, j\neq i}^{n} x_{ij} = 1 && i = 1,\ldots,n; \\
&u_i - u_j + n x_{ij} \leq n-1 && 2 \leq i \neq j \leq n; \\
&1 \leq u_i \leq n-1 && 2 \leq i \leq n.
\end{aligned}
$$

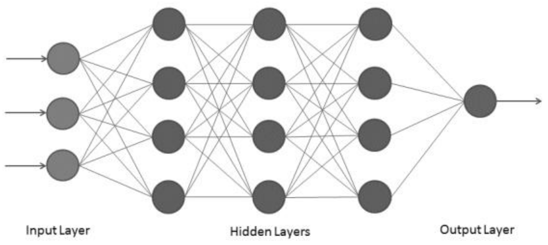
The example applies to real life with the example of a traveling salesman trying to come up with the most efficient way to travel to all cities, starting and ending on the same city, whilst only visiting each city once.

3. B. Applying a Genetic Algorithm to Optimize a Neural Network

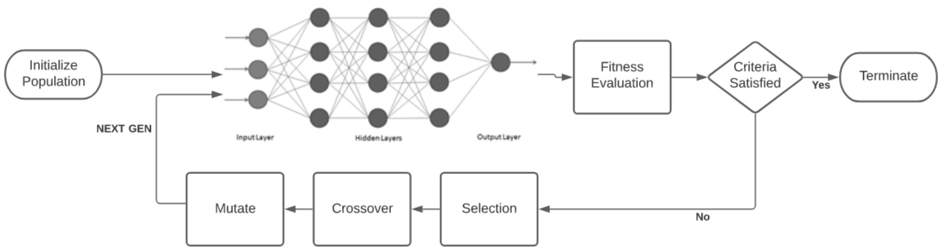
Genetic Algorithms and Neural Networks are two techniques used for learning and optimization, that originate from the human biological system. Although the two have advanced in different directions throughout the years, their strengths can be combined to create a very powerful system (Mahajan, R. and Kaur, G., 2013.). At one hand, Genetic Algorithms are fundamentally based on natural selection, combined with 'survival of the fittest' methodologies, like Darwinism Evolution (Flew, A., 2018).



On the other hand, a Neural Network is largely inspired from biological neural networks that exist in animal brains, where the input is processed by the neuron and an output is produced, parsed to the next neuron.



With that in mind, the combination of the two, results to Evolutionary Artificial Neural Networks (EANNs). It is important to note that the hyperparameters (variables) of the neural network, which determine its structure, are only set before the training. Hence using a GA, the neural networks could be considered the individuals of each population, and their hyperparameters will be the chromosomes getting optimized by the GA's operands, crossover, and mutation.



An example of an EANN, is the use of a Genetic Algorithm to train a Neural Network. With a Neural Network in mind, the GA can evaluate its fitness by calculating the error (f(x) = 1 / error, where error = target- output), which can then be minimized to achieve greater results, by adjusting the neuron's hyperparameters (Yao, X., 1993).

One of the advantages of this combination, is that the GA allows for better optimized hyperparameter generation, which results to outputs closer to global minimum. Another advantage is that GA are considered as very robust due to their adjustable parameters and operands. On the other hand, some disadvantages of using an EANN, could be the complexity of picking the optimal cost function as well as training algorithm (Schaffer, J.D., Whitley, D. and Eshelman, L.J., 1992).

In summary, genetic algorithms are a powerful tool that can be used to optimize towards a goal. In summary, the conjunction of the ability of Neural Networks to learn and the ability of Genetic Algorithms to optimize and reproduce, can be used as a very powerful tool in findings the optimal solution to a problem.

Appendix

Mathew, T.V., 2012. Genetic algorithm. Report submitted at IIT Bombay.

Sugihara, K., 1997, March. Measures for performance evaluation of genetic algorithms. In Proc. 3rd. joint Conference on Information Sciences (pp. 172-175).

Mahajan, R. and Kaur, G., 2013. Neural networks using genetic algorithms. International Journal of Computer Applications, 77(14).

Jebari, K. and Madiafi, M., 2013. Selection methods for genetic algorithms. International Journal of Emerging Sciences, 3(4), pp.333-344.

Flew, A., 2018. Darwinian evolution. Routledge.

Yao, X., 1993. A review of evolutionary artificial neural networks. International journal of intelligent systems, 8(4), pp.539-567.

Schaffer, J.D., Whitley, D. and Eshelman, L.J., 1992, June. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In [Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks (pp. 1-37). IEEE.

Fitness Functions Source: https://www.sfu.ca/~ssurjano/ackley.html