## Genetic Algorithm (GA)

Charles Darwin’s Theory of Evolution through natural selection in his “On the Origin of Species” book (1876), though not completely factual as noted by the evolutionary biologist Stuart Newman (Mazur, 2008) and Gordons’s (1999) findings on “The Concept of Monophyly”, has inspired many through analysis of its applications. The theory is grounded on the observation called ‘survival of the fittest’, through which, fitter and more capable individuals of a population naturally achieve higher survival rates in their given environments, providing them longer lifespans and more opportunities to pass on their superior genetic codes to the next generation. The weaker members of the population would typically achieve lower chances to pass on their inferior genetics, eventually being completely overwritten from the genetic history by fitter candidates over the progressing generations. Darwin’s theory hinged on the concept of variation; that there is a range of differences between the genetic makeup of the individuals in the population, which when accumulated through the principles above would be able to push organisms past the barrier of *species* toward something completely different, perhaps new, but ultimately better.

The Genetic algorithms are a family of computational models that drew inspiration from Darwin’s evolutionary theory known to have been originally introduced and investigated by the American engineer John Holland sometime in the 1960s (Coley, 1999; Holland, 1992) and it “is one of the first population-based stochastic algorithms proposed in history” (Mirjalili, 2019). Using a chromosome-like data structure and recombination operators to simulate the mechanics of DNA reproduction, these algorithms have been applied to a very broad range of problems often offering optimized potential solutions. Typically, a genetic algorithm (GA) begins with a list of chromosomes representing a population. These chromosomes are often randomized to provide unique starting points for each member of the population. Then two processes called *evaluation* and *fitness allocation* are used to award each member a measure of ‘attractiveness’ (also called fitness) in such a way that those chromosomes which represent a better solution to the target problem are given more chances to 'reproduce' than those chromosomes which are poorer solutions. The ‘attractiveness’ of any given individual is typically assigned relative to the current population. Lastly, individuals are selected to be bred based on those allocations making a ‘next generation’, and mutations are randomly assigned under an appropriately low percentage to allow more variability in the search space. As the algorithm iterates through the generations, members of the population increase in fitness until a concluding ‘best individual’ is found after the ‘stop criteria’ is/are reached (Whitley, 1994).

A consideration that influenced the design of GAs over the years is *selection pressure* (SP). SP is the “degree to which the better individuals are favoured: the higher the selection pressure, the more the better individuals are favoured” (Miller & Goldberg, 1995). The SP is the driving force for improvement over succeeding generations in the GA and it is a largely influential factor when it comes to GA convergence. If the mating pool selection and breeding section of the GA lend too much focus on the best individuals of a particular generation (the SP is too high), disregarding the potential gain from the other members, then the algorithm converges toward those member’s solution ardently as the optimum. With regards, to the consideration given to local vs global optimums, this is not necessarily always a good trait. So, care is needed to be given when analysing the SP of a given GA design.

All-in-all, the description of the GA is, to a certain extent, left purposefully vague because of the large flexibility in the application that the algorithm offers. As long as the given problem can be encoded as a chromosome-based population and a function for evaluation of individual fitness (or attractiveness), the GA can be utilized. Certainly, this ease in flexibility plays part in the reason why the GA remains one of the most popular evolutionary algorithms in literature (Mirjalili, 2019) with various applications found like the automatic design of convolutional neural networks for image classification (Sun et al., 2020), as a solver for systems of second-order boundary value problems (Arqub & Abo-Hammour, 2014), optimization of cogeneration plant systems (Ahmadi & Dincer, 2010) and optimizing a back-propagation (BP) neural network (Ding et al., 2011), among many others. However, with regard to the problem domain of this study (TSP), delving a bit deeper into its inner workings and variations gains importance.

### Defining Problem domain

To utilize the GA, a method of representing the TSP as chromosomes and an evaluation function must be established. To re-iterate, the TSP can be defined as “Given a list of cities and distances between each pair of cities, what is the shortest round trip from any given city of origin, visiting all other cities on the map once each?”. To break that down, there are two important rules to watch out for:

1. Each city must be visited exactly one time
2. We must return to the starting city

Each city has a unique ID and can also be represented as a vector having x and y coordinates. In the context of Genetic Algorithms, a gene can be symbolized by the city’s ID. An individual, represented as a chromosome, is a list of genes, meaning a list of cities. And lastly, a population would naturally still mean a list of individuals.

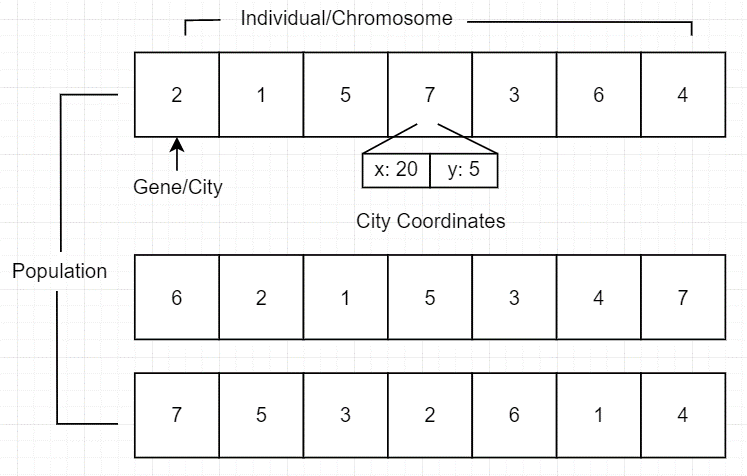


Figure 1 - TSP setup for the GA

In keeping with the first rule, valid individuals are ones containing all cities in a given map exactly once within the sequence without any repetitions. As you may already be able to discern, this would also mean that each individual/chromosome also represents a route or solution to the given map. By visiting the cities on the map in the sequence specified in a chromosome, you would have traveled the entire map visiting all cities once over as the TSP requires.

*Individual/Chromosome = Sequence of Genes/Cities = TSP Route/Solution*

(Synonymous within this context)

The TSP aims to find “the shortest round trip”. So, for evaluation of how good any given solution is, one would have to traverse the map in the sequence instructed and calculate the total distance traveled in this solution. In keeping with the second rule, included at the end of the calculation is the return trip back to the starting position. Because the algorithm aims to find the ‘shortest’ distance, the total distance traveled is then inversed and awarded as the evaluation score for the proposed solution. The lower the distance, the higher and better the score.

Equation 1 - Evaluation Function

Now that our problem domain has been defined concerning the GA, we can begin breaking the inner workings of the algorithm but first, some clarification is in order. Often, the notions of *evaluation* and *fitness* are used interchangeably, however, it is useful to distinguish both functions. Evaluation provides a means to measure the performance of a given individual regarding a set of parameters extracted from the problem domain. The fitness function then takes that performance score and transforms it into an allocation of reproductive opportunities (Whitley, 1994). Fitness and the fitness functionwill be discussed further in a following section of this chapter.

### Initial Population

In the same manner that Darwin’s theory hinged on variations within the population, so also are variations critical for the functionality of the GA. Each member of the population represents a solution in the solution search space and as the algorithm progresses, the population members search and narrow down on promising regions in the search space. As such, the GA gains great benefit from an initial population that is as spread out within the search space as possible because it brings a better chance of finding promising regions of the search space at an earlier time. In this study, the initial population was generated simply using random distribution but, of course, extra techniques like utilizing Gaussian distribution may be used with the aim to maximize search space coverage (Mirjalili, 2019).

### Evaluating Fitness

Now that a population has been established, the next step is selecting individuals for breeding creating the next, better, generation. As mentioned earlier, the process of selecting mating candidates is split into two parts, *evaluation,* and *fitness*. First, using the evaluation function, described in **Equation 1**, an evaluation score is assigned to each member of the population determining how ‘good’ each of their proposed solutions is. It is then the job of the fitness function to select a mating pool filled with candidates by simulating breeding chances for each individual, given their evaluated scores. The size of the breeding pool returned from the fitness functions must be the same as the original population size for reasons that will become apparent when considering the breeding section of the algorithm. In literature, there have been many fitness function variations that have been used with the GA, each offering a measure of success. Consider 4 of the most popular fitness function variations.

#### Roulette Wheel Selection (RWS)

As the title suggests, the concept of natural selection is simulated using a roulette wheel type selection process. In this case, the fitness score refers to the number of slots allotted on the wheel to each member of the population, calculated relative to each member’s *evaluation score*. Because of the wide range of evaluation scores that may arise depending on the map sizes used, a good preparation step would be to normalize all scores beforehand for algorithm consistency. Normalization changes the range of all values supplied from whatever scale they were previously down to the scale of [0-1]. 0 represents the lowest value,1 the highest, and all other values are assigned a corresponding value on that scale relative to their distance from those lowest and highest values. This normalized score can then be scaled up to fit whatever wheel size you are looking for by multiplying all scores by the largest slice you would like to have on the wheel. For example, scaling up by 10 would mean 0 is the lowest amount of possible slots allotted and 10 is the highest. Whitley (1994) offered the suggestion, which I used, where each resulting value’s decimal remainders is used as a probability for offering a bonus slot to that member. For my experiment, I decided to use a scale of 1-10. 1 being the lowest number of slots possible and 10 being the highest.

Equation 2 – Normalization

Equation 3 - Fitness Score

(where M is the largest slice size, R is the decimal remainder from (X’ \* M), and prob(R) is the probability operator returning 0 or 1 depending on if that probability is reached)

A designation pointer is attached to a position on the wheel and every time the wheel is spun, the member landing in that position is copied into the breeding pool. In this way, a breeding pool of the same size as the original population can be collected stochastically from the wheel, also implying that the wheel would have to be spun *[population size]* number of times.

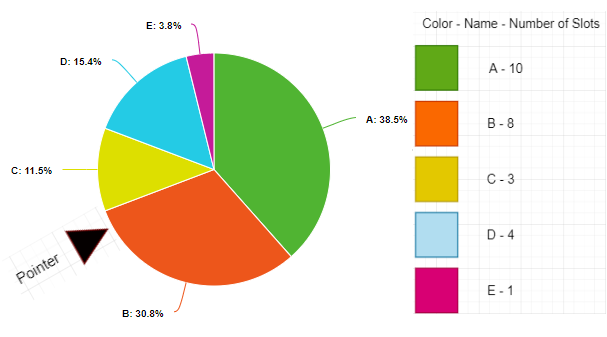


Figure 2 - Roulette Wheel Sampling

#### Stochastic Universal Sampling (SUS)

Over time, weaknesses were highlighted in the workings of the RWS, and variations of that fitness function emerged to solve those problems. One of those was the problem of inefficiency, for which, the SUS function was developed to tackle. In the RWS, there is a requirement for multiple spins of the wheel before a selected breeding pool can be compiled. The SUS algorithm, however, works the same as the RWS but, by having multiple selection pointers evenly spaced around the wheel, all breeding pool members can be selected simultaneously with significantly fewer spins.

Chart, pie chart

Description automatically generated

Figure 3 - Stochastic Universal Sampling

#### Ranked Based Sampling (RBS)

Another problem noted with the RWS technique was its SP which was arguably too high. As seen in figures 2 and 3, because of the great scores found with individuals A and B when compared to the others, they were assigned portions that nearly dominate the entire wheel, leaving little room for selection chances for the other individuals. The goal of the RBS (also known as Linear Rank Selection) is to tackle this by performing the allotment proportional to each individual’s ‘rank’ rather than their evaluation score directly. Using their evaluation scores, all members of the population are ranked from 1st till Nth and then fitness is distributed using those assigned ranks, presenting a more evenly distributed wheel to select from.

Chart, pie chart

Description automatically generated

Figure 4 - Rank Based Selection

#### Tournament Sampling (TS)

Unlike the variations mentioned above which followed the pattern of the RWS, the TS uses a completely different mechanism for selecting a mating pool for breeding. In the TS, pairs of individuals, each chosen randomly from the population, are put against each other in a competition, of which, the winner is then copied directly into the mating pool for breeding. The winner of a competition is selected by comparing the evaluated scores of each member's proposed route. Over time, it has been highlighted that TS also runs into a similar problem as the RWS: that its’ SP is too high. Due to this, some variations have been tested to try to remedy that. For example, Miller & Goldberg (1995) experimented with the effects of noise in the TS applied to the scores of the members before each competition and found success with their proposed model. In my experiment, I introduced a new probability named *delta* affecting method through which the winner is selected. For example, a delta of 0.7 would mean a 70% chance that the better individual is chosen as the winner of this competition and a 30% chance the weaker member is chosen instead. A decrease in the delta value lowers the selection pressure, slowing down the convergence rate of the algorithm. I found … to be the optimum setting for this delta value and the justification for that is demonstrated in chapter 4.

### Breeding

After a mating pool is compiled of those selected for breeding through the fitness function, they have to be bred to create the next generation. In the natural inspiration for the GA, the chromosomes in the genes of a male and female are combined to produce a new chromosome. The same is technique employed by the GA through the *crossover operator*. Here, 2 parents are chosen and section(s) of their genetic makeup are swapped across to create 2 children. There are many different methods of crossover, but 2 of the most common variations are single and double point crossover. Other variations include Uniform crossover, 3 parent crossover, Cycle crossover, and many others (Mirjalili, 2019).

Diagram

Description automatically generated

Figure 5 - Crossover Operator (Mirjalili, 2019)

In this project, an adaptation of single-point crossover was used. Respecting condition number one of the problem domain (no duplicate cities within a route), the genetic code cannot be copied over blindly otherwise this condition may be violated. So in this project, after a random swap point was chosen for each pair of parents, and the first section of the parent's genetic code up to that swap point is copied over directly into their children’s chromosomes, the rest of the chromosome for each child was filled with genes coming from their second parent not already been existing in the child. This second stream of genes is collected in the same order they occur in the second parent’s chromosome. As noted in Figure 6, every pair of parents produces a pair of offspring, so for this reason I noted earlier that the number of parents chosen to be in the mating pool should correspond to the size of the population because that would also be the number of children in the next generation made from that pool.

Diagram

Description automatically generated

Figure 6 - Crossover TSP

### Mutation

In review, the GA has taken a varied initial population and, through selection and breeding operators, has isolated desirable parts of their gene sequence to narrow down a ‘best solution’. But it is noted that we are not guaranteed to have the globally best genetic components within the populations that the algorithm began with. Although some care has been given to SP while developing fitness functions, so far, there is no way to introduce new data into the algorithm as it recycles old patterns in an effort to optimize them. This finding indicates the importance of mutation. Mutation can be seen as the operator charged with maintaining the genetic diversity of the population as it aims to preserve the diversity embodied in the initial generation. It does this by introducing new information into the genetic sequence allowing the population to ‘leapfrog’ over potential sticking points (Coley, 1999).

Through mutation, one or multiple genes within the chromosome are altered after the creation of child solutions. The rate of mutation within the GA is often kept low because higher mutation rates revert the algorithm back into a primitive random search algorithm. As such, studies have been done to isolate an optimal value to set that parameter. The study done by Mirjalili (2019) found that to be at 0.6% which was the rate used in this study.

Within the context of this study, mutations needed to be handled uniquely. The chromosome of any individual within this study was a sequence of unique cities to visit within the map with no repetitions. So, including a mutation that simply randomly altered a single gene within that sequence could break the validity of the sequence. Therefore, in this study, mutations were treated as a swap between two randomly chosen cities along the sequence. In other words, after breeding, each gene of each child has a 0.6% chance of swapping locations with another randomly assigned gene along the sequence.

Diagram

Description automatically generated

Figure 7 - GA Mutation

### Extensions and Improvements

Many extensions that layer over the basic GA operation to improve its functionality have been implemented such as assigning dominant and recessive genes, and the concept of niche and speciation. Two of the most popular ones in use today are Elitism and Steady-State (Assareh et al., 2010; Johnson & Rahmat-Samii, 1997). The natural inspiration for these ties back into the idea of ‘survival of the fittest in which fitter individuals are preserved, carrying on for longer than their weaker contemporaries.

#### Elitism

Due to the stochastic nature of the GA, it is possible for the next generation to have a best individual with lower fitness than the preceding generation’s best representative. Elitism is a technique, developed to address this concern, in which the fittest ‘elite percentage’ of a generation is retained into the next generation. In this experiment, when elitism is used, for each iteration, the members of the population are evaluated and ordered by their score. Then, the top-scoring group, whose size is decided by the elite percentage assigned for the operation, is kept intact while the others are replaced by their children.

#### Steady-State (SS)

This function can be thought of as the overlapping of generations takes the methodology of elitism even further. It operates at the end of the breeding and mutation stages, at which point, the children generated are only allowed to replace their parents if they are fitter individuals than them. This steers the algorithm closer towards greedy optimization but it exhibits advantages, like faster convergence, in many applications. There are a few methods of implementing SS. One method is by storing the new child generation in a separate list and then overriding selected parents with those in the new, fitter individuals. Another method is by appending the children to the end of the parent's list, temporarily creating an enlarged population size. Then, using chosen criteria, members are removed until the population size returns to its origin.

With all the care given to SP earlier, this technique seemed a bit counter-intuitive because it heightens the SP. As such, when implementing it, I decided to try a different technique that localized the SS, limiting its effect on the SP, but still staying true to its essence. Earlier, my crossover operator and mutation sections split the population into pairs of parents used to create pairs of children. So, I applied the SS function at this pair level. This means that when the group of 4 is decided on (these 2 parents made these 2 children), they are then sent to a SS operator, which returns the best 2 out of the 4 candidates. These best 2 are then stored as the new members of the next generation. This proposed method of SS GA limits the power of high-performing children found by allowing them only to replace their direct parents rather than any other, possibly lower, members of the population.

The effectiveness of my proposed model will be explored further in chapter 4. It should also be noted that because the SS function aims to retain the best of each generation for the next, combined use of SS and Elitism brings redundancy.

## Bibliography

Ahmadi, P., & Dincer, I. (2010). Exergoenvironmental analysis and optimization of a cogeneration plant system using Multimodal Genetic Algorithm (MGA). *Energy*, *35*(12), 5161–5172. https://doi.org/10.1016/j.energy.2010.07.050

Arqub, O. A., & Abo-Hammour, Z. (2014). Numerical solution of systems of second-order boundary value problems using continuous genetic algorithm. *Information Sciences*, *279*, 396–415. https://doi.org/10.1016/j.ins.2014.03.128

Assareh, E., Behrang, M. A., Assari, M. R., & Ghanbarzadeh, A. (2010). Application of PSO (particle swarm optimization) and GA (genetic algorithm) techniques on demand estimation of oil in Iran. *Energy*, *35*(12), 5223–5229. https://doi.org/10.1016/j.energy.2010.07.043

Coley, D. A. (1999). *An Introduction To Genetic Algorithms For Scientists And Engineers*. World Scientific Publishing Company.

Darwin, C. (1876). *The Origin of Species: By Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* (6th ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511694295

Ding, S., Su, C., & Yu, J. (2011). An optimizing BP neural network algorithm based on genetic algorithm. *Artificial Intelligence Review*, *36*(2), 153–162. https://doi.org/10.1007/s10462-011-9208-z

Gordon, M. S. (1999). The Concept of Monophyly: A Speculative Essay. *Biology and Philosophy*, *14*(3), 331–348. https://doi.org/10.1023/A:1006535524246

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press.

Johnson, J. M., & Rahmat-Samii, V. (1997). Genetic algorithms in engineering electromagnetics. *IEEE Antennas and Propagation Magazine*, *39*(4), 7–21. https://doi.org/10.1109/74.632992

Mazur, S. (2008, October 11). *The Origin of Form Was Abrupt Not Gradual—Archaeology Magazine Archive* [Interview Article]. Archaeology Archive - A Publication of the Archaeological Institute of America. https://archive.archaeology.org/online/interviews/newman.html

Miller, B. L., & Goldberg, D. (1995). Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Syst.*, *9*(3), 193–212.

Mirjalili, S. (2019). Genetic Algorithm. In S. Mirjalili (Ed.), *Evolutionary Algorithms and Neural Networks: Theory and Applications* (pp. 43–55). Springer International Publishing. https://doi.org/10.1007/978-3-319-93025-1\_4

Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J. (2020). Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Transactions on Cybernetics*, *50*(9), 3840–3854. https://doi.org/10.1109/TCYB.2020.2983860

Whitley, D. (1994). A genetic algorithm tutorial. *Statistics and Computing*, *4*(2), 65–85. https://doi.org/10.1007/BF00175354