## Genetic Algorithm

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* towards 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).

The description of the algorithm is, to a certain extent, left purposefully vague because of the large flexibility of 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, and it can be used to symbolize a gene in this context. An individual, represented as a chromosome, would then be a list of genes (or cities). And lastly, a population is a list of individuals

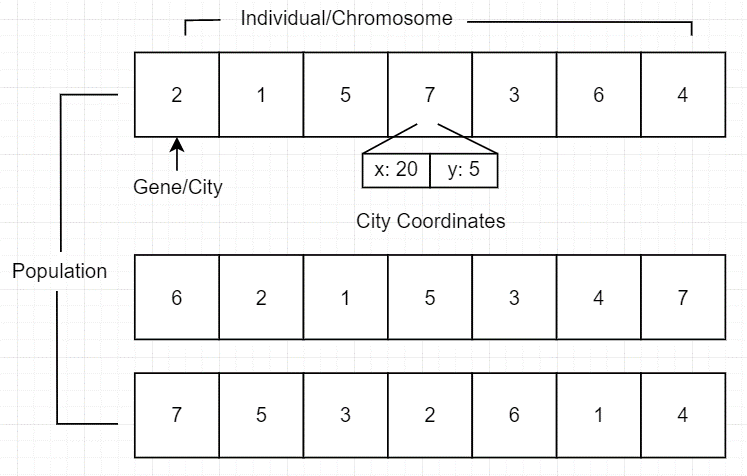


Figure 1 - TSP setup for the GA

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 score, the better.

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 on 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 aims to maximize search space coverage (Mirjalili, 2019).

### Mating Pool Selection

Now that a population has been established, the next step is selecting individuals for breeding creating the next, hopefully fitter, 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**, is used for each member of the population, and an evaluation score is assigned determining how ‘good’ each proposed solution 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 the most popular fitness function variations.

#### Routlette 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* refferes to the number oof slots allotted on the wheel to each member of the population and that fitness score is 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 first step would be to normalize all scores before use to get them within manageable ranges. 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. Each score’s decimal remainders, after all calculations are complete, can be used as a of measure the probability for whether that member gaining a bonus slot or not (Whitley, 1994).

Equation 2 – Normalization

Equation 3 - Fitness Score (where M is the largest slice size, R is the decimal remainder from (Score’ \* M), and prob(R) is the probability operator returning 0 or 1 depending of if that probability is reached)

A designation pointer should be attached to a position on the wheel and after everytime the wheel is spun, the member landing in that position is copied into the breeding pool. A breeding pool of fittest representatives of the same size as the the original population must then be chosen from wheel meaning that the wheel would have to be spun *[population size]* times.

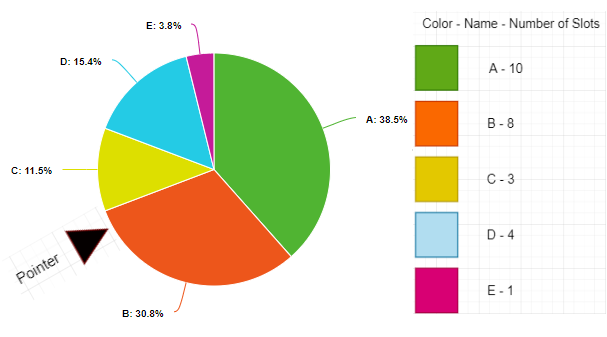


Figure 2 - Roulette Wheel Sampling

#### Stochastic Universal Sampling (SUS)

Over time, faults were highlighted in the workings of the RWS and variations of that fitness function emerged to solved those problems. One of those faults that the SUS function aims to tackle is the problem of inefficiency. In the RWS, there is a requirement for multiple spins of the wheel before a selected beeding 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 within 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 inherent bias towards fitter individuals. As seen in figures 2 and 3, because individual A scored significantly higher than individual E, its’ allotted portion on the wheel dwarfs E’s. Due to the diferenciation between local and global optimums being an important consideration in optimization algorithms, leaving this bias unadressed invites potential problems. The goal of the RBS (also known as Linear Rank Selection) is to tackle this by performing the alloment proportional to each individual’s ‘rank’ rather than their evaluation score directly. Unsing their evaluation scores, all members of the population are ranked from 1st till Nth first and then fitness is distributed accordingly, presenting a more evenly distributed wheel to select from.

Chart, pie chart

Description automatically generated

Figure 4 - Rank Based Selection

#### Tournament Sampling (TS)

### Breeding

### Mutation

As new generations are made from the genetic make-up of selected, fitter, individuals of the last generations, the algorithm begins to narrow down on specific patterns that have provided the best solutions within the search space. Unfortunately, this recycling of favorable genes possesses a flaw highlighted by the consideration of local vs global optimums. Up till now, 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 obviously, we are not guaranteed to have the globally best genetic components within the populations that the algorithm began with. So, the GA may be narrowing down on a local rather than global optimum solution, while also giving an indication of 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 convert the algorithm 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 because members of the sequence must already exist on the map and no repetition is allowed. 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 5 - GA Mutation

### Results and Choices

### Conclusion

## 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

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.

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

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