Genetic Algorithms and Optimization

Project Group 1

Date: November 26, 2023

Group Members:

Arno Nazarian, Umer Bahsir, Abinav Binkumar, and Ethan Mcleod

Abstract

Genetic algorithms (GAs) are a class of optimization algorithms inspired by the process of natural selection. It is now widely used to solve complex optimization problems in various industries. This research report provides an in-depth analysis of genetic algorithms, their underlying principles, applications, and benefits they provide in optimizing projects The paper also addresses and reviews the challenges of GA's possible directions for future research in this area.

Introduction

Genetic algorithms, or GAs for short, are a class of algorithms that are widely used in numerous technological systems. It adheres to evolutionary mechanisms like mutations and natural selection. They are part of a broader scope of evolutionary algorithms intending to approximate time-consuming solutions to a self-solving evolutionary program.

The idea of a self-evolving evolutionary process to survive and adapt in a given environment is derived from the idea that all living organisms share, originally proposed by Charles Darwin in 1859 in his work on “Origin of Species” [1].

Within the quickly growing sector of AI, evolutionary algorithms, especially GA’s are proving to be an incredibly valuable approach to any task AI’s are tasked with. Given the rapid advancement of AI, it is reliable to say that it will play a large role in the future of human society. To adapt to the needs of humanity, AI will need to adapt to utilizing GA’s, thus establishing the need for further research and development into Genetic algorithms.

This paper will delve further into the origin, understanding and application of GA’s in terms of their effectiveness and methodization in modern-day technology.

1. Evolution of Genetic Algorithms and Optimization

George Boole’s invention of a system of logic that made calculations based on true and false values is known as Booleans today. This later became binary algebra and was the foundation for not only algorithms but all computer code [2]. Later, this was expanded upon by Alan Turing to create a hypothetical mathematical model that can simulate any algorithm regardless of its complexity, called a Turing machine. In essence, it was a mathematical model that was an infinitely long piece of tape with bits written on it, being changed out from the processes of creation, reading, updating, and deleting, by a head that moves along the tape [2].

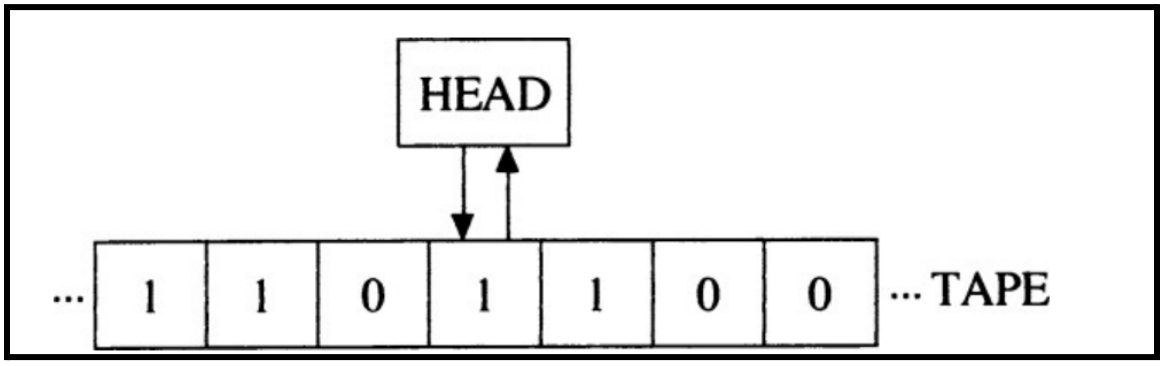

Figure 1: A simple Turing machine, showing how an algorithm could be created through brute force, 1 (variable is true) and 0 (variable is false)

Alan Turing laid the foundation for genetic algorithms in the 1950s. He proposed the basic concepts of biology and mathematics could be used to come up with new algorithms. However the term "genetic Algorithms" was first used in the 1960s by John Holland who was an American computer scientist and scientist. In his book "Adaptation in Natural and Artificial Systems" published in 1975 he described complex systems and their ability to improve over time. John Holland helped popularize genetic algorithms to a broader audience [3].

Holland introduced some key concepts such as selection, crossover, mutation, fitness function, and chromosomes and genes. which today have become the fundamental components of genetic algorithms. Holland propped the use of chromosomes to represent potential solutions. He also introduced the concept of a fitness function, which evaluates how well a solution solves a problem. Holland then went on to describe the genetic algorithm operators, selection, crossover and mutation. The body of our report covers all of these concepts, as they are fundamental to understanding how genetic algorithms work. This goes to show how influential John Holland's work has been in this field.

David E. Goldberg played a major role in popularizing genetic algorithms. His 1989 book *"Genetic Algorithms in Search, Optimization, and Machine Learning"* provided a great introduction to the field of genetic algorithms, making them more accessible to the general public [4]. Goldberg's work significantly contributed to the practical application of genetic algorithms in optimization problems.

During the 1990s researchers started focusing more on combining genetic algorithms with optimization techniques. This led to the development of an entire new field, called *"Evolutionary Computing"* [5]. The application of genetic algorithms was expanded by combining them with decentralized and self-organized systems, neural networks, and other evolutionary techniques.

In the 2000s genetic algorithms became more efficient at solving a wide range of problems. This was mainly because researchers started combining genetic algorithms with machine learning and optimization techniques. During this time more and more companies started to realize the potential of genetic algorithms to solve problems, which led to more increased application.

Today, genetic algorithms have become widely accepted in the field of computer science and programming. Genetic algorithms play an important role in solving complex problems efficiently that cannot be solved using traditional algorithms. They have applications in various fields such as communications, finance, and more. The success of genetic algorithms comes from their foundation is based on biology, mathematics, and the principles of evolution. Genetic algorithms offer a better and more efficient way to solve problems.

2. Genetic Algorithm: Fundamental Principles

2.1 Principles of Natural Selection

Genetic algorithms are primarily based on the process of natural selection. They operate based on the principles of evolution, including selection, crossover, and mutation. These operators are then applied to a population of potential solutions to iteratively improve the quality of solutions over generations.

Selection involves the identification and favouring of individuals with higher fitness within a population. Fitness represents the ability of an individual solution to perform well in the given problem domain. Through mechanisms like roulette wheel selection or tournament selection, GAs mimic the natural process of differential reproductive success, ensuring that individuals with advantageous traits are more likely to contribute genetic material to the next generation [6].

The Idea of Natural Selection was proposed by Charles Darwin, in Chapter IV of his publication *"On The Origin of Species"* [1], Darwin expands on how given a particular population, when trifled with problems from both inner and outer species situations, those of the population who are the most adept, will survive and produce offspring who are generally better than the previous generation. In our demonstration of a Q-Learning Snake Algorithm, we will demonstrate how the offspring from previous generations will aid in the completion of an end goal.

2.2 Representation of Solutions

The success of genetic algorithms depends very much on the representation of the solutions. Common representations include binary strings, real-valued vectors, and permutations. The choice of representation can impact the efficiency of the algorithm. Choosing a specific solution is influenced by the different types of problems and the different representations that are available. Some common representations include binary strings, real-valued vectors and permutations.

2.2.1 Binary Strings

Binary Strings are just simple 1's and 0's. The binary string can be as long as we need it to be, and each bit in the string represents some variable. The representation is perfect for problems where the variables are simple decision variables that only have two possible outputs. The binary string approach provides a clear and efficient way of representing problems where only two decisions can be made.

A classic example is the travelling salesman problem, where the goal is to find the shortest possible route that visits each city exactly once and returns to the origin city. Using binary strings, each city is represented by a binary string, where each bit corresponds to whether the city is included or not. The binary representation allows for a straightforward encoding of which cities are included in the route. 1 represents a city included, and 0 represents a city not a part of the route. An example solution might look like 10001101100001

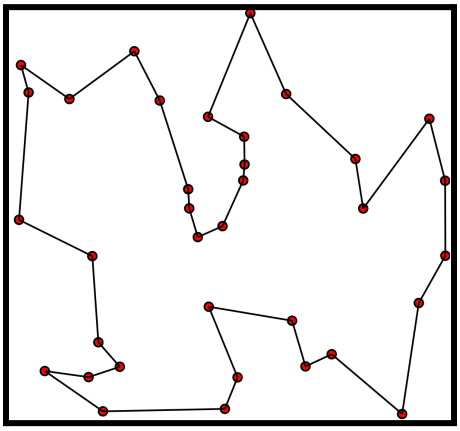

Figure 2: Visual representation of a solution to the Travelling Salesman Problem

As we can see in the figure above the visualization solution to the problem is very hard to represent and understand, whereas the binary string solution is very clear and concise.

2.2.2 Real-Valued Vectors

Real-valued vectors represent solutions as arrays of continuous values. This representation works extremely well for optimization problems where variables can be any real number. A basic example of

when real-valued vectors can be used is in the case of function optimization. Consider a parabola where the goal is to find values of variables that minimize or maximize the parabola. Vectors work well in this case. A parabola has an equation in the form: $y = ax^2 + bx^2 + c$. Suppose the parabola is at its maximum at $y = 2.5x^2 - 1.8x + 3.2$. We can represent this as a real-valued vector as [2.5, -1.8, 3.2]. This representation is well-suited for optimization problems where variables can take any real value.

2.2.3 Permutations

Permutations are used when the order of elements is crucial, such as in scheduling or routing problems. Permutations are different from combinations where the order does not matter. The main goal of permutation is to preserve the order. Permutations can be extremely helpful in scheduling problems. For example, let's say in a person's daily job that the order in which they complete the tasks has an impact on their overall efficiency. In this case, it would make sense to use a permutation as we want to find the most efficient order in which to complete the tasks.

The choice of the representation depends very much on the type of the problem. A good choice representation can enhance the algorithm's ability to find better solutions more efficiently. Choosing a good representation for the solution is a very important step in applying genetic algorithms to specific problems. Picking a solution represents a great deal of balance between the expressiveness of the representation and the computational efficacy of the algorithms.

3. Genetic Algorithm Operators

The process for GAs was derived from ideas proposed in a scientific book published by J.H Holland, based on Darwinian principles such as Natural Selection, Mutations etc. These principles culminated into a model which is widely utilized in many mediums. The idea that from a given population the best will arise when faced with a challenge and produce offspring that inherit those traits is proven in reality hundreds of thousands of times over. In GA’s the particular approach is as follows.

3.1 Starting Population

A particular species, or in the case of GA’s, a set of calculations that determine a form of input, are inserted into a situation and act accordingly. The starting population in a GA will always be less efficient than later generations, simply because there is no foundation to work with. In an unmonitored setting, the starting population's moves would cause a drastic change in the future populations. However, in a monitored setting explicit rewards and punishments can be given to speed the process up.

3.2 Fitness

After the initial population, we evaluate the best members and award and or punish them based on their actions. This derives from Chapter Four of On the Origin of Species. Particularly when discussing the idea of Natural Selection, and how the most fit will live to reproduce and the next generation will inherit some aspects from their parents. "As more individuals are produced than can possibly survive, there must in every case be a struggle for existence, either one individual with another of the same species, or with the individuals of distinct species, or with the physical conditions of life [1]." The struggle in GA’s is the member of a generation who should be the most rewarded. By using this conditional behaviour it becomes feasible to attain results in a feasible amount of time.

3.3 New Generation

From the best of the previous generations a new generation is selected to continue the algorithm and the idea is to attain better results than their parents as the new generation can build off the previous one.

3.3.1 Selection

Multiple Parents, typically two parents will be selected to produce the new generation they are chosen according to their *Fitness* results.

3.3.2 Crossover

From the selected parents, their genes and or inputs are inherited by the child. By passing on the best parents from the *Fitness* results, we can ensure a high likelihood that the next generation will utilize moves that will forward progression by a higher margin compared to a new generation with no parents. This again calls upon the idea put forward by Charles Darwin, the idea that Natural Selection will ensure the highest chance of survival. In GA's it ensures that a new generation is closer to completing the solution.

3.3.3 Mutation

The idea of Mutation is derived from how the reproduction process from one generation to the next is not perfect because no deviations or *Mutations* can occur which would normally change a child of perfect offspring to one with different traits that are neither inherently from the parents. Many researchers believe that mutations are not inherently debilitating, "Based on their effects on fitness, mutations can be divided into three broad categories: the 'good' or advantageous that increases fitness, the 'bad' or deleterious that decreases it and the 'indifferent' or neutral that are not affected by selection because their effects are too small [7]". In many circumstances, a mutation may prove extremely advantageous in a controlled environment. However, when dealing with such complexity as the human body, possibly mutations become hurtful to the host.

3.4 Accepting and Replacing

After all of the genes of the new generation are created, we begin to replace the older generation with the new generation, similar to how an older generation will eventually become entirely replaced by the following generations. After this, the new generation is tested again for fitness.

3.5 Looping

The process of a new generation being born is continued indefinitely until the desired criteria are met. If done correctly, the last generation will be the most successful and thus the solution is solved and the algorithm is complete.

In concluding the discussion on Genetic Algorithm Operators, it's intriguing to note the foundational influence of Darwinian principles on these algorithms, as utilized by John Holland in his influential work, "Adaptation in Natural and Artificial Systems." The collection of concepts like natural selection and mutations into the structure of genetic algorithms forms the core of their computational strategy.

4. Genetic Algorithm Applications

The original concept for a Genetic Algorithm was proposed by J.H Holland in his book *Adaptation in Natural and Artificial Systems*, the work done by J.H Holland has truly revolutionized how situations that would take thousands of years can be simplified into a self-sorting and learning algorithm.

The application of GA's can be seen everywhere in modern-day society. To demonstrate the efficiency of a GA, we employed our group of Second Year Software Engineers to utilize a GA with Q-Learning. Our task was to solve a maze using the model employed by using GAs.

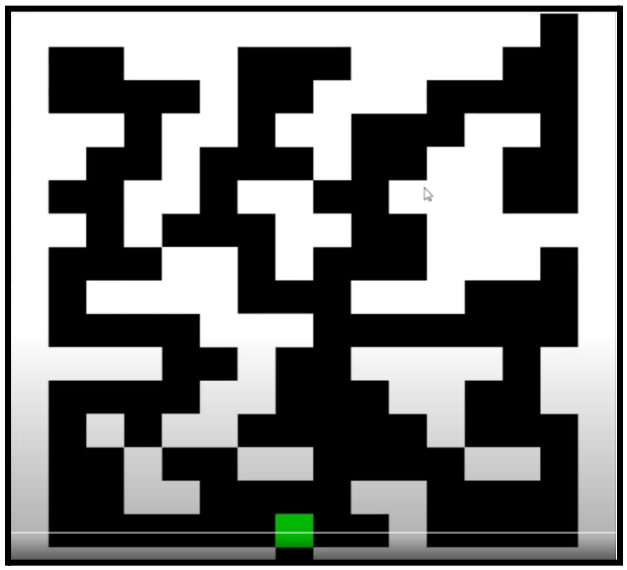

Figure 3: Demonstration of Maze

In the figure, the maze is displayed. It follows many different paths, however, the objective is to complete the maze by reaching the top right corner of the maze. An explanation of this will be given.

4.1 Initial Generations and Incentive System

Following through the GA model, we begin with 32 members of a new generation. For example, each member of a generation is allowed to make 2 moves. This marks the new generation. Next, the fitness of each member was evaluated. To bring order to the chaotic nature of random movements, we utilized a punishment and reward system (Q-Learning). Depending on the moves made a certain amount of points would be made, in the example, +2 points would be given for any moves that were up and to the right, and +1 for any moves at all. However, -1 points were given if they hit a wall. We instructed the algorithm to obtain as many points as possible, this helps the algorithm to understand that hitting a wall is a bad move.

4.2 Reproduction

After a fitness analysis, we choose the next parents of the generation. With the way the GA is set, it is possible for multiple groups of parents, in cases where there are a large number of parents we utilize code to ensure that a member of a generation inherits the points from every parent.

```
max_index = scorelist.index(max(scorelist))

if max(scorelist) < 0:
    generaltrack -= 2
    for i in range(8):
        guylist[i].pop()
        guylist[i].pop()
else:
    for i in range(len(guylist)):
        if i != max_index:
            for j in range(generaltrack):
                guylist[i][j] = guylist[max_index][j]

    for i in range(int(len(guylist) / 2)):
        for j in range(generaltrack):
            temp = random.randint(1, int(generaltrack / 2))
            if temp == 1:
                guylist[i][j] = random.randint(1, 4)

    for i in range(len(scorelist)):
        if i != max_index:
            scorelist[i] = scorelist[max_index]
```

Figure 4: Code Snippet of Fitness Test

4.3 Mutation

From the inherited traits, out of the 32 members, 16 of them will have a randomized move. This is done to both emulate the algorithm and to assist the algorithm in solving the program. In multiple runs of the program, we noticed several times where a mutation occurred and positively impacted the algorithm. The algorithm would typically become stuck at a certain point in the points system we assigned.

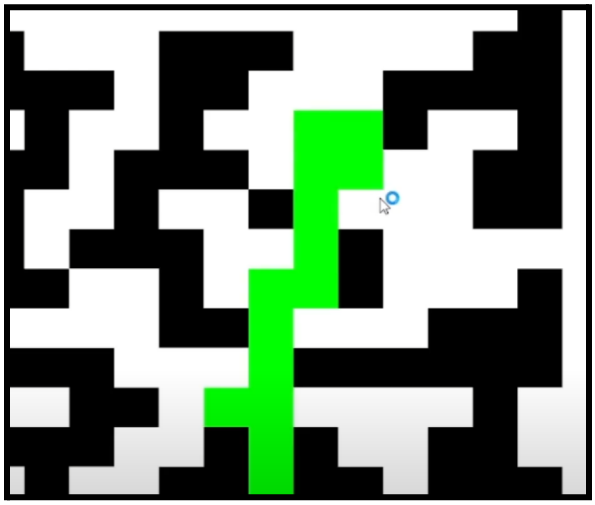

Figure 5: Mutation Assistance

In that particular region, the algorithm would typically become stuck as going up it would hit a wall, and the space to go right has a very small failure margin. Often a mutation would kick in that would help the algorithm make a rightwards move. This was consistent in multiple runs.

4.4 Looping

The proposed algorithm would loop as many times as needed to complete the maze, real completion times averaged around 6 seconds. Thus proving the algorithm's incredible efficiency.

However, the algorithm that was used was fairly simplistic. It demonstrated how computational methods can be used to solve problems that may take humans a very long time to complete.

5. Strengths and Limitations

5.1 Strengths

Global Search Capability: Genetic algorithms excel in exploring the entire solution space, making them excellent for finding global optimizations.

Adaptability: GAs can adapt to various types of problems and do not rely on specific problem systems to solve them.

Parallelization: The parallel nature of genetic algorithms allows them to handle multiple solutions to problems at the same time.

Speed Efficiency: Due to the mathematical nature of algorithms, computers can solve them extremely fast. The main advantage is that it can narrow down problems that may take years to solve to ones that take days, or even hours.

5.2 Limitations

Computational Intensity: Genetic algorithms can be computationally expensive, especially for problems with large solution spaces.

Parameter Sensitivity: The performance of GAs is sensitive to the assigned parameter settings, and finding appropriate parameter values can be challenging.

Lack of Problem-Specific Knowledge: Algorithms are rarely paired with prior information which can lead to inaccurate results in a much more complicated model.

6. Challenges and Open Issues

6.1 Scalability

Scalability remains an issue, as genetic algorithms do not scale well with complexity. Complex problems include where the number of variables is huge, and often the search space increases exponentially. This makes it difficult to implement GA's on broader problems such as designing a racecar, as there are too many variables. To use genetic algorithms on such problems, the problem must be broken down into simpler problems first. Typically you see algorithm encoding designs for fan blades, rather than an entire engine. The second problem is protecting the parts of the car that have evolved to represent good solutions from further mutation. Mutating a solution that works perfectly well can make it worse in some cases.

6.2 Hybridization

Hybridization involves combining genetic algorithms with other optimization techniques. It is a powerful strategy to enhance the performance and versatility of the algorithms. However, hybridization also has its own set of challenges. Integrating an algorithm with another optimization technique requires you to understand both algorithms. Integrating can become difficult if the algorithms have different variables or representation solutions. Not all algorithms are compatible with each other. Ensuring that two algorithms are integrated properly and work together is a difficult task. Hybrid algorithms also run the risk of "overfitting", where they are overly tailored to a specific problem but fail to generalize to new instances of that same problem.

6.3 Dynamic Environments

Genetic algorithms struggle when operating on dynamic data sets. In a dynamic environment, the optimal solution changes over time, and this becomes an issue for GAs as they begin to converge early on toward a solution. We see this in the maze example in section 4 where the green block starts moving upwards as they start figuring out the correct path. However, if the maze was constantly changing then the algorithms would have failed and would have not reached the correct solution.

Several solutions have been proposed by researchers to solve this problem. One method that has been proposed includes increasing genetic diversity somehow and preventing early convergence. Another technique suggests increasing the probability of mutation when the solution quality drops. Evolution strategies and evolutionary programming can also be used when dealing with dynamic data sets.

7. Future Directions

The future direction of genetic algorithms is likely to be shaped by advancements in technology, research trends, and what the various industries need. While it is hard to predict what the future holds for genetic algorithms, we can use research and trends in the field to predict the future of genetic algorithms and optimization.

7.1 Evolutionary Machine Learning

Integration of genetic algorithms with machine learning techniques is likely to increase in the future. Genetic algorithms already have some application in machine learning such as the optimization of neural networks, inheriting qualities of neurons and more[8]. Future researchers can use techniques such as feature extraction, parameter tuning, and model optimization to contribute to and advance the field of evolutionary machine learning.

7.2 Quantum Genetic Algorithms

Quantum computing is a very exciting and upcoming field. Quantum algorithms can greatly improve the computing performance of a normal computer. Combining quantum computing with genetic algorithms would allow the algorithm to take advantage of quantum computing, which would improve the speed and efficiency of the algorithm. Currently, some quantum genetic algorithms use parallelism and superposition to search solution space more efficiently and rapidly [9]. Research in combining genetic algorithms with quantum computing could lead to more efficient quantum genetic algorithms.

7.3 Ethical and Responsible AI practices

In the very near future, we could see greater emphasis on the ethical practices of AI practices in the application of genetic algorithms. With the rise of various AI applications such as Google Brad or Chatgpt, there is bound to be a rise in the applications that combine various genetic algorithms with artificial intelligence. However, using AI also comes with additional tasks of putting a great emphasis on

the ethics of using AI in genetic algorithms. Ensuring that fairness, transparency, and accountability in optimization tasks will be essential for researchers.

8. Conclusion

The report has delved into the history, process and utilization of genetic algorithms with the maze game. Through exploration of various examples, it has become apparent that GA's are crucial to the future of technology.

Our report on the development of GAs has demonstrated the effectiveness of GAs and it helps to understand the importance of how such a model will be able to aid the future of humanity. Particularly because GAs mirror our human history. The very idea for GA originated from Charles Darwin himself nearly 200 years ago as an extension of Darwin's influential title On the Origin of Species. The genetic operators, namely selection, crossover, and mutation, contribute to the continual evolution of solutions, allowing for the discovery of optimal or near-optimal solutions in diverse problem domains.

While Genetic Algorithms are an important part of modern-day technology, it is essential to note that they have quite restrictive limitations. Primarily the lack of premature knowledge makes completing particular tasks quite difficult. Not to mention no AI, has a given form of common sense, meaning that no conventional frameworks exist to easily apply a GA.

It is our firm belief that further research should be done into the world of GAs, with the release of AI such as ChatGPT, it has become apparent that Artificial Intelligence will surely play a role in the future of humanity. With proven models such as GA's, we can be sure that the fruit of our labours will be paid off.

In conclusion, through the realm of GA humanity will reach new heights that were never thought possible before, as we begin to cultivate how problems that would normally take ages to solve are reduced to a fraction of that time. Though we have many shortcomings with GAs at the moment, engineers, scientists and researchers must contribute to solutions that will allow GAs to assist humanity for centuries to come.

9. Contribution Matrix

Contribution Matrix				
Group Members:	Umer Bashir	Abinav Binkumar	Ethan Mcleod	Arno Nazarian
Research Report Contribution	25 %	25 %	25 %	25 %
Presentation Contribution	25 %	25 %	25 %	25 %

Table 1: Contribution Matrix showing each group member's contribution.

10. References:

[1] C. Darwin, “On the origin of species,” Encyclopædia Britannica, https://www.britannica.com/biography/Charles-Darwin/On-the-Origin-of-Species (accessed Nov. 23, 2023).

[2] J. Pyram, “A brief history of algorithms and their impact on the world,” Medium, https://levelup.gitconnected.com/a-brief-look-at-algorithms-and-their-impact-on-the-world-2794f8dd70aa (accessed Nov. 25, 2023).

[3] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, Mass: MIT Press, 2010.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. New Delhi, India: Pearson, 2013.

[5] “Evolutionary computation,” Wikipedia, https://en.wikipedia.org/wiki/Evolutionary_computation (accessed Nov. 23, 2023).

[6] “Genetic algorithm,” Wikipedia, https://en.wikipedia.org/wiki/Genetic_algorithm (accessed Nov. 23, 2023).

[7] L. Loewe and W. G. Hill, “The population genetics of mutations: Good, bad and indifferent,” Philosophical transactions of the Royal Society of London. Series B, Biological sciences, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2871823/ (accessed Nov. 25, 2023).

[8] S. Chaudhary, “Genetic algorithm applications in machine learning,” Genetic Algorithm Applications in Machine Learning, https://www.turing.com/kb/genetic-algorithm-applications-in-ml (accessed Nov. 26, 2023).

[9] Quantum approximate optimization and K-means algorithms ... - iopscience, https://iopscience.iop.org/article/10.1088/1742-6596/1719/1/012100 (accessed Nov. 27, 2023).