Design of Machines and Structures, Vol. 13, No. 2 (2023), pp. 98–104.
https://doi.org/10.32972/dms.2023.021

A BRIEF OVERVIEW OF GENETIC ALGORITHMS

KRISTÓF SZABÓ
University of Miskolc, Institute of Machine Tools and Mechatronics
H-3515 Miskolc-Egyetemváros
kristof.szabo@uni-miskolc.hu
https://orcid.org/0000-0002-4126-6687

Abstract: This article examines the topic of Genetic Algorithms, discussing the basic concepts and terminology related to Genetic Algorithms. The advantages, disadvantages and limitations of the mentioned algorithms are defined. The different population models and the process of parent selection are presented, as well as the areas of application of the procedure are summarized.

Keywords: *genetic algorithms, optimization, evolutionary algorithms*

1. INTRODUCTION

Continuing advances in manufacturing and materials science have enabled design engineers to produce complex digital models that are difficult or impossible to produce using traditional manufacturing methods and technologies. This unprecedented manufacturing flexibility will continue to offer even more opportunities and challenges for computer-aided design of digital models. With these possibilities, even the most experienced design engineer's intuition can be underwhelmed. To address design challenges, computer algorithms have been developed to solve the problem based on specific design goals and constraints. This type of genetic algorithm-driven design process is now called Generative Design.

This design approach is not a single algorithm or ready process, but an approach by which the designer defines a series of instructions, rules and relationships that precisely identify the steps required to implement the proposed design and the resulting data or geometry.

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of genetics and natural selection. It is often used in solving optimization problems, in research and in machine learning stages as it is very useful when the objective function is highly complex, just as in case of Generative Design.

In the optimization process, we have an input and an output, and, in the process, we look for the input that gives the ‘best’ output. In mathematical terms, the definition of ‘best’ is the maximization or minimization of one or more objective functions by varying the input parameters. The set of all possible solutions constitutes the search space. In this search space, there is a point that gives the optimal solution. The objective of optimization is to find a given point or set of points in this search space (Erdős-Sélley, Gyurecz, Janik, & Körtvélyesi, 2013), (Hegedűs, 2002).

2. GENETIC ALGORITHMS

Genetic algorithms are search-based algorithms based on the principles of natural selection and genetics. Genetic algorithms were developed by John Holland, his students and colleagues at the University of Michigan, and David Edward Goldberg has achieved great success in solving various optimization problems (Goldberg & Samtani, 1986), (Holland, 1992), (Mitchell, 1998). In genetic algorithms, possible solutions to a given problem undergo recombination and mutation, as in natural genetics. Through this process new individuals are created, and the process is repeated over several generations. Each individual or solution is assigned a so-called fitness value, which is determined by the value of the objective function. More fit individuals are given a better chance of producing individuals with a higher fitness value. This is in line with the Darwinian theory of 'survival of the fittest' (Darwin, 1859). Thus, over generations, better solutions are continuously produced until the stopping criterion is reached. Genetic algorithms are spontaneous processes, but they perform much better than random local search, in which only different random solutions are tried, following the best ones so far (Haupt & Haupt, 2004), (Deb, 2011).

2.1. The justification for genetic algorithms

Genetic algorithms have many positive features that have made them extremely popular. First, they do not require derivative information, which is not available for many real-world problems. In many cases they are faster and more efficient than traditional methods. They can handle both continuous and discrete functions as well as multi objective problems. It provides a list of 'good' solutions, not just a single solution. You always get an answer to the problem, which can only get better over time. It is useful when the search area is very large and there are many parameters.
In addition to the so-called advantages, the disadvantages or, in a sense, limitations should also be mentioned. Genetic algorithms are not suitable for solving all types of problems, especially those that are simple and where derived information is available. The calculation of the fitness value is repeated, which can be computationally expensive for certain problems. Assuming a stochastic process, there is no guarantee that the solution is optimal and of the best quality. If not properly implemented, the algorithm may not converge to the optimal solution on the other hand it shows how close a given design solution is from the designer's goals. Genetic algorithms are also cannot be applied when the aim is to find ideas to the design task, and we need originally new ideas (Takács & Kamondi, 2006).

For many problems in computer science, even the most powerful computing systems may take a very long time to solve a problem. In this case, genetic algorithms have proven to be a powerful tool to provide usable, near-optimal solutions in a short time. Traditional gradient-based methods work by starting from a random point and moving in the direction of the gradient until a maximum value is reached. This technique is efficient and works very well for objective functions with a maximum point. But in most real situations this is not the case. Mostly, we have to deal with objective

functions that have multiple peaks, which is why these methods fail. This is because they suffer from getting stuck in the local optimum.

2.2. Basic concepts of Genetic Algorithms

Table 1
Terminology of Genetic Algorithms
(Michalewicz, 1994), (Mitchell, 1998)

Title	Definition
Population	The subset of all coded solutions to a given problem.
Chromosomes	A possible solution to the problem.
Gene	An elementary position on a chromosome.
Allele	The value that a gene takes for a given chromosome.
Genotype	Population of the computational space.
Phenotype	The population in the real solution space in which solutions are represented as they appear in the real world.
Decoding	For simple problems, the phenotype and genotype spaces are the same. In most cases, however, the phenotype and genotype spaces are different. Decoding is the process of transforming a solution from the genotype to the phenotype space, while coding is the process of transforming the solution from the phenotype to the genotype space.
Fitness Function	A simply defined fitness function is a function that takes the solution as input and produces the fitness of the solution as output.
Genetic operators	They change the genetic make-up of the offspring. This includes cross-breeding, mutation and selection.

When examining genetic algorithms, it is very important to go into sufficient detail about the basic terminology of the procedure, which is summarised in *Table 1*.

2.3. The basic structure of Genetic Algorithms

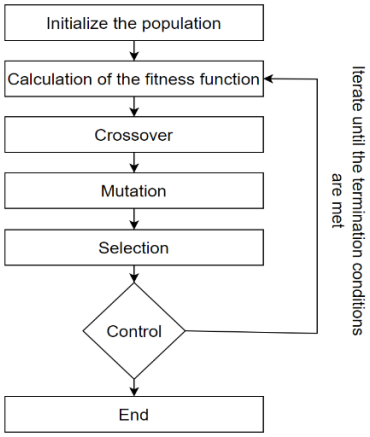


Figure 1. *Flowchart of Genetic Algorithms*

Figure 1 shows the flowchart of the method, which starts with an initial population and can be randomly generated. Every individual or design option can be mentioned as a possible solution to the complete design problem. From this population, so-called parents are selected for subsequent mating. Thanks to the crossing and mutation operators used, new offspring are generated. Finally, these offspring replace the existing individuals in the population and the process is repeated. In this way, genetic algorithms attempt to mimic human evolution to some extent (Mitchell, 1998).

2.4. Coding of solutions

When implementing a genetic algorithm, one of the most important aspects is the representation we want to use to represent our solutions. Observations have shown that the selection of the appropriate representation to properly define the mappings between phenotype and genotype spaces is essential for the success of Genetic Algorithm (Goldberg & Samtani, 1986), (Mitchell, 1998).

The binary representation is one of the simplest and widely used representations in genetic algorithms. This representation is shown in *Figure 2*. In this type of representation, the genotype consists of a set of bits. For some problems, when the solution space consists of Boolean decision variables: yes or no, the binary representation is natural.

0	0	1	0	1	1	1	0	1	0

Figure 2. *Binary representation of genotype*

Figure 3 shows a solution where genes can be defined by continuous rather than discrete variables. In this case, a real-valued representation is most appropriate. However, the accuracy of these real-valued or floating-point numbers may be a function of the computer.

0.5	0.2	0.1	0.4	0.6	0.8	0.3	0.9	0.2	0.8

Figure 3. *Representation of a genotype with continuous variables*

For genes with discrete values, the solution space cannot always be restricted to binary values. If four criteria have to be distinguished, then in such cases the representation of integers may be appropriate, as illustrated in Figure 4.

1	3	2	4	1	4	3	2	2	3

Figure 4. *Representation of a genotype with discrete variables*

In many problems, the solution itself is the order of the elements. In such cases, the permutation representation is the most obvious solution, as shown in Figure 5.

1	5	9	6	4	3	2	8	10	7

Figure 5. *Permutation representation of the genotype*

2.5. Evaluation of solutions

In steady-state genetic algorithms, one or two offspring are generated in each iteration and replace at least one individual in the population. These algorithms are also called incremental algorithms. In the generative model, 'n' offspring are generated, where 'n' is the population size, and at the end of the iteration the entire population is replaced.

A fitness function is the objective function for the genetic algorithm – it is a function that takes as input a possible solution to a problem and gives as output a measure of how 'good' the solution is relative to the problem. It shows how close a design solution is from the designer's original goal. The fitness value is calculated repeatedly. In most cases, the fitness function and the objective function are the same, with the objective being to maximise or minimise the objective function. The fitness function must be fast to compute and, importantly, it must be measurable. In some cases, it may not be possible to calculate the fitness function directly due to the complexity of the problem. In such cases, approximations are made.

2.6. Parent selection

Table 2

Methods of parent choosing

(Mitchell, 1998), (Michalewicz, 1994), (Yang & Soh, 1997)

Method	Description
Roulette wheel selection	A pie chart is divided into 'n' units and the area of each unit is proportional to the fitness value of the parents. Using a fixed point, the selection is made by rotating and stopping the diagram.
Stochastic universal sampling	Similar to roulette wheel selection, but there are several fixed points so that all parents can be selected at the same time.
Tournament selection	"k" individuals are randomly selected from the population and the best of them is chosen to become the parent. It is also very popular in the literature, as it can work even with negative fitness values.
Rank selection	All individuals in the population are ranked according to their fitness. The selection of parents depends on the rank of the individual and not on fitness. It is used when the fitness values of individuals in a population are very close to each other.
Random selection	Parents are randomly selected from the existing population. There is no selection pressure on the fitter individuals, so this strategy is not popular.

The choice of parents is very important for the convergence rate in genetic algorithms, as the right parents lead to better solutions. Maintaining population diversity

is crucial to the success of genetic algorithms, so it is important that a highly suitable solution does not produce convergent results within a few generations, as this leads to a loss of diversity. This process is called premature convergence, which is an undesirable state in genetic algorithms.

Fitness Proportional Choice is one of the most popular ways of parenting. In it, each individual has a probability of becoming a parent proportional to his or her fitness score. Those with higher fitness scores have better genes and are more likely to be selected. There are several possible cases of fitness proportional selection. The most common methods of selection are summarised in *Table 2*.

3. SUMMARY

Genetic algorithms are not only used in case of Generative Design, but it is most often used in optimization tasks where the value of a given objective function needs to be maximized or minimized under a given set of constraints. In economics, genetic algorithms are used to characterise various economic models. Genetic algorithms are used to train neural networks, in various digital image processing (DIP), in dense pixel matching tasks, to solve various scheduling problems, and to process spectrometric data. Genetic Algorithm Based Machine Learning (GBML) is a narrow field of machine learning in which algorithms have been used to plan the trajectories of robotic arms. The method has also been exploited in parametric design of aircraft, which has been used to design aircraft more efficiently. Genetic algorithms are good approaches for multimodal optimization, in which multiple optimal solutions must be found (Mitchell, 1998), (Borsodi & Takács, 2022).

The following article summarises the topic of genetic algorithms, covering the basic concepts and terminology related to genetic algorithms. The advantages, disadvantages and limitations of these algorithms were identified. The different population models and the process of parental selection were presented, and the applications of the procedure were summarised.

REFERENCES

Borsodi, E., & Takács, Á. (2022). Generative Design: An Overview and Its Relationship to Artificial Intelligence. *Design of Machines and Structures*, 12 (2), 54–60, https://doi.org/10.32972/dms.2022.013.

Darwin, C. (1859). *Darwin, C. R.: 1872. The origin of species by means of natural selection, or the preservation of favoured races in the struggle for life.* London: John Murray.

Deb, K. (2011). Multi-objective Optimisation Using Evolutionary Algorithms: An Introduction. In:. In L. Wang, A. Ng, & K. Deb, *Multi-objective Evolutionary Optimisation for Product Design and Manufacturing.* London: Springer. https://doi.org/10.1007/978-0-85729-652-8_1

Erdős-Sélley, C., Gyurecz, G., Janik, J., & Körtvélyesi, G. (2013). *Mérnöki optimalizáció.* Bodapest: Typotex.

Goldberg, D. E., & Kuo, C. (1987). Genetic algorithms in pipeline optimization. *Journal of Computing in Civil Engineering*, 1 (2). https://doi.org/10.1061/(ASCE)0887-3801(1987)1:2(128)

Goldberg, D., & Samtani, M. (1986). Engineering optimization via genetic algorithm. *Engineering*.

Haupt, R., & Haupt, S. (2004). *Practical Genetic Algorithms.* 2 ed. Hoboken, New Jersey: John Wiley and Sons.

Hegedűs, G. (2002). A módszeres géptervezés alkalmazása ipari mérőgép fejlesztése esetén. *Doktoranduszok Fóruma.* Miskolc: Miskolc University Press.

Holland, J. (1992). *Adaptation in Natural and Artificial Systems* (Reprint ed.). Bradford Books.

Michalewicz, Z. (1994). *Genetic Algorithms + Data Structures = Evolution Programs.* 2. ed. Springer.

Mitchell, M. (1998). *An Introduction to Genetic Algorithms.* MIT Press.

Takács, Á., & Kamondi, L. (2006). A genetikus algoritmusok. In V. Csibi (Ed.), *OGÉT 2006: XIV. Nemzetközi Gépész Találkozó* (pp. 332-335). Kolozsvár: EMT.

Yang, J., & Soh, C. (1997). Structural Optimization by Genetic Algorithms with Tournament Selection. *Journal of Computing in Civil Engineering*, 195–200. https://doi.org/10.1061/(ASCE)0887-3801(1997)11:3(195)