International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, page no. 354-373

To cite this article: MA. Vu Thi Khanh Trinh (2025). RESEARCH ON GENETIC ALGORITHMS: CONCEPTS, MODELS, AND APPLICATIONS, International Journal of Education and Social Science Research (IJESSR) 8 (3): 354-373 Article No. 1085, Sub Id 1689

RESEARCH ON GENETIC ALGORITHMS: CONCEPTS, MODELS, AND APPLICATIONS

MA. Vu Thi Khanh Trinh

Tan Trao University, Tuyen Quang, Vietnam

DOI : https://doi.org/10.37500/IJESSR.2025.8324

ABSTRACT

This research on genetic algorithms focuses on clarifying the concept, model structure, and practical applications of this algorithm. The article systematizes fundamental knowledge about genetic algorithms, from how they operate based on the mechanism of natural selection to core components such as fitness functions, crossover, mutation, and selection. The study is conducted through a theoretical overview, technical analysis of the algorithm, and experimental investigation using sample problems to demonstrate its effectiveness. The research results identify a standard model for genetic algorithms and prove their broad applicability in areas such as combinatorial optimization, scheduling, parameter tuning for machine learning models, and engineering design. The article also opens up prospects for applying genetic algorithms to emerging fields such as image processing, big data mining, multi-objective optimization, and solving complex problems in biomedical science.

KEYWORDS: research, genetic algorithm, concept, model, application

INTRODUCTION

In practice, the search for methods to solve classes of optimization problems is a field that has received significant attention from scientists worldwide. Numerous studies have been conducted in this area, primarily developed within the framework of optimization theory applied to mathematical programming and problem modeling in information technology to develop algorithms for solving classes of NP-Complete (NPC) problems. In recent years, an important research direction has emerged in the field of evolutionary computation: the study of Genetic Algorithms (GA).

The Genetic Algorithm (GA) is a component of Evolutionary Computation (EC), a rapidly growing field of artificial intelligence. EC can be divided into five main research directions: 1/ Genetic Algorithm (GA): Based on the natural genetic process to improve solutions over generations, starting from an initial population of candidate solutions; 2/ Evolutionary Programming (EP): Based on evolutionary rules, aiming to combine multiple methods, each capable of solving part of a problem, into a complete and effective problem-solving method; 3/ Evolutionary Strategies (ES): Based on a set of initial strategies, evolution is used to generate new strategies that best adapt to the real-world environment; 4/ Genetic Programming (GP): An extension of GA in the field of computer

programming. Its goal is to automatically generate computer programs that optimally solve specific problems; 5/ Classifier Systems (CS): Specialized GAs used in machine learning and rule discovery in rule-based systems.

GA, like other evolutionary algorithms, is founded on a principle considered to be a realistic axiom: “The process of natural evolution is the most perfect and rational process and inherently optimal”. This evolutionary process reflects optimization in that each new generation tends to be better than the previous one.

This article presents fundamental concepts, models, key parameters, operators, the general execution mechanism of genetic algorithms, as well as classical genetic algorithms and real-coded genetic algorithms.

2. HISTORY OF RESEARCH

The formation and development of Genetic Algorithms (GAs) around the world can be traced through the following key milestones:

1960: The first idea of evolutionary computation was introduced by Ingo Rechenberg in his work "Evolution Strategies". This idea was subsequently developed further by many researchers.

1975: The Genetic Algorithm was invented by John Holland and developed collaboratively with his colleagues and students. His book "Adaptation in Natural and Artificial Systems" compiled the research and development outcomes of their work.

1992: John Koza applied GAs to construct programs that solve specific problems and introduced the concept of Genetic Programming (GP).

Today, Genetic Algorithms have become increasingly important, especially in the field of optimization—a domain filled with interesting and practical problems that are often difficult and lack efficient solving algorithms.

3. RESEARCH METHODOLOGY

This paper employs the method of literature synthesis and analysis. Scientific documents related to genetic algorithms—including books, articles, and studies from both domestic and international sources—are collected and categorized according to content: concepts, models, and applications. Subsequently, the documents are analyzed and compared to clarify different perspectives, highlight key characteristics, and examine how genetic algorithms are applied. Finally, logical reasoning is used to synthesize the findings and propose future research directions.

4. RESEARCH RESULTS

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

4.1. Basic Concepts

4.1.1. Individual and Chromosome

In Genetic Algorithms (GAs), an individual represents a potential solution to the problem. In general, an individual may consist of multiple chromosomes, but for simplicity, we often assume that each individual contains only one chromosome. Therefore, in GA, the terms "individual" and "chromosome" are considered equivalent.

A chromosome is made up of several genes, each of which can take on different values that determine a certain characteristic. In GA, a gene is viewed as an element in the chromosome string.

4.1.2. Population

A population is a set of individuals that share certain characteristics. In GA, a population is understood as a collection of candidate solutions to a problem.

4.1.3. Selection

In nature, the process of selection and the struggle for survival cause changes in the individuals of a population. Individuals that are better adapted to their environment tend to have higher survival and reproductive chances. Poorly adapted individuals tend to be eliminated.

In GA, based on this principle of natural selection, the selection process involves choosing individuals with better fitness to proceed to the next generation or to undergo crossover, with the goal of producing better offspring. There are many selection methods, but they all aim to give higher selection probabilities to fitter individuals.

4.1.4. Crossover

Crossover in nature involves combining the characteristics of parents to produce offspring. In GA, crossover is considered the recombination of features from two parent solutions to generate a new solution that is hopefully better than its parents. This is one of the main operators in GA.

4.1.5. Mutation

Mutation is the alteration of one (or more) genes in a chromosome to create a new chromosome. Mutation occurs with a lower probability than crossover. It may produce a better or worse individual compared to the original. However, in GA, we always aim to design mutations that improve solutions over generations.

4.2. GA Model

With the above concepts introduced, the GA model is typically represented by the following flowchart (Figure 1):

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

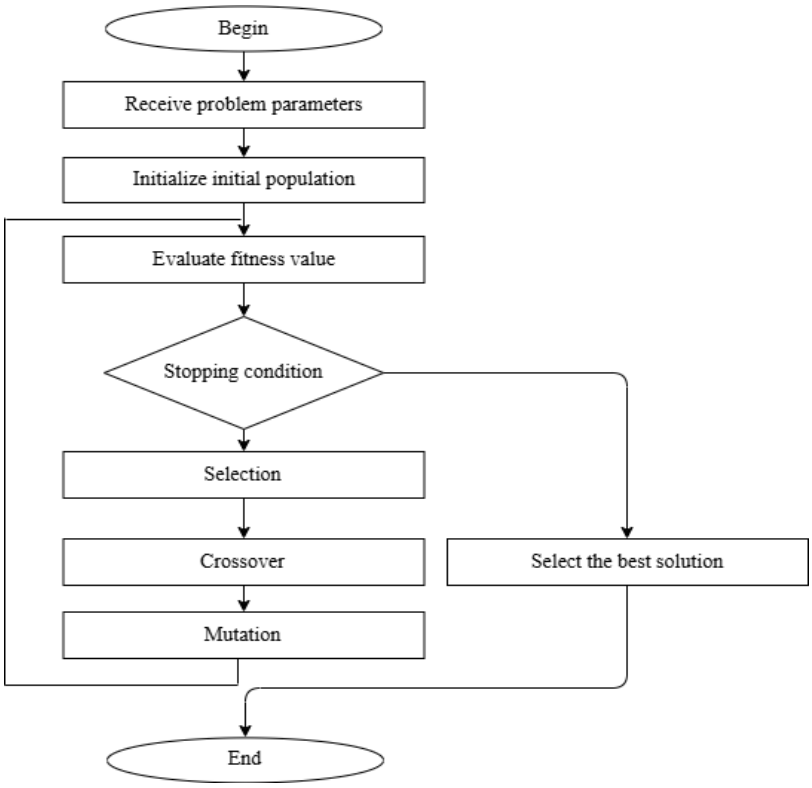

Figure 1: Flowchart of the Genetic Algorithm (GA)

1/ Initialize problem parameters: Set the initial parameters for the problem.

2/ Initialization: Randomly generate a population of n individuals (which are n initial candidate solutions for the problem).

3/ Create a new population: Generate a new population by repeating the following steps until the new population is complete:

3.1/ Evaluate fitness: Calculate the fitness value of each individual.

3.2/ Check termination condition: Evaluate whether the termination condition has been met.

3.3/ Selection: Select parent individuals from the current population based on their fitness (individuals with higher fitness have a greater chance of being selected).

3.4/ Crossover: Perform crossover between selected parent pairs with a predefined crossover probability to generate new offspring.

3.5/ Mutation: Apply mutation to the offspring with a predefined mutation probability to generate mutated individuals.

4/ Termination check: If the termination condition is satisfied, the algorithm stops and returns the best solution found in the current population.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

4.3. Parameters of Genetic Algorithm (GA)

4.3.1 Population Size

The population size indicates how many individuals are present in a population (in one generation). Research and experiments have shown that the population size should not be too small or too large. If there are too few individuals, crossover is less likely to occur, and only a small portion of the search space is explored. This may result in missing good solutions. However, having too many individuals is also inefficient, as it slows down the algorithm and affects its overall performance. Studies have also shown that increasing the population size beyond a certain limit is not beneficial.

4.3.2. Crossover Probability

The crossover probability determines how frequently crossover occurs to create the next generation. If the crossover probability is denoted by pc, then each individual has a pc chance of undergoing crossover. If crossover is not performed, the offspring will be identical to the parents. If crossover is performed, the offspring will inherit some traits from the father and some from the mother.

4.3.3. Mutation Probability

The mutation probability indicates how frequently the genes of a chromosome change. If the mutation probability is pm, then each gene of any chromosome has a pm chance of being mutated. The mutation operator prevents the GA from getting stuck in local optima. However, if the mutation probability is too high, the GA will become a random search algorithm.

Note 1: From the GA implementation diagram, we can make several observations:

+ GA uses probabilistic reasoning to find optimal solutions to complex problems rather than deterministic mathematical approaches. However, this randomness is guided by the fitness value. The fitness function helps GA find the optimal solution among many possible ones.

+ GA does not focus on the detailed nature of the problem but rather on the solution itself - aiming to find optimal conditions for operating and clustering possible solutions.

+ GA is particularly useful for solving problems that require global optimization in a large and uncontrolled search space, thanks to its ability to explore a representative portion of the space without checking every point in the entire domain.

4.4. GA Implementation Mechanism

4.4.1. Encoding

To implement a GA, the first step is to represent the problem solutions in a certain format (mathematical, computational, etc.). This process is called encoding. Common encoding methods include:

a. Binary Encoding

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

Binary encoding is the most common method. In this method, each chromosome is represented as a binary string, where each bit may represent a particular characteristic of a solution.

Binary encoding is widely used in single-variable or multivariable optimization problems. Each binary string represents the function at a set of variable values.

Despite its popularity, binary encoding can sometimes produce a larger encoding space *than the actual solution space, making it inefficient for some problems.*

b. Permutation Encoding

In permutation encoding, each chromosome is a sequence of numbers representing an ordering. This method is suitable for problems involving order, such as the traveling salesman problem or scheduling problems. Manipulating chromosomes in these problems involves permuting the numbers to change their order.

c. Real-Valued Encoding

Direct encoding using real values is useful for problems that involve complex or real-number data. Here, each chromosome is a sequence of values (integers, real numbers, characters, or complex objects).

This method is typically used for numerical optimization problems and requires specific crossover and mutation operators for each problem. Usually, each chromosome is encoded as a vector in a space and is well-suited for continuous optimization problems.

d. Tree Encoding

This method is used in mathematical expressions. Each chromosome is represented as a tree structure of a group of objects.

4.4.2. Initial Population Generation

Generating the initial population is the first step in a GA. Typically, the initial population is created randomly with solutions that satisfy the problem constraints (though it's unclear whether these are optimal). Depending on the specific problem, various initialization methods are used. The higher the quality of the initial population, the better the solution GA will provide.

4.4.3. Fitness Function Definition

According to various studies, the fitness function is one of the two most critical factors determining the success or failure of a GA. The fitness function must reflect the true value of a chromosome in meeting the problem’s requirements.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

4.4.4. Selection Mechanism

The selection mechanism is used to choose individuals from the population P(t) to perform crossover and mutation and generate a new population P(t+1). There are several methods for selecting individuals. Below are commonly used selection mechanisms.

Let us define the following notations:

- Let chromosome i be denoted as v_i.

- The fitness function of chromosome v_i is f(v_i).

- The population size is *pop_size* ().

- The number of chromosomes to be selected is *N*.

Roulette Wheel Selection Mechanism:

Step 1: Calculate the total fitness of the entire population: $F=\sum_{i=1}^{pop-size} f(v_i)$

Step 2: Calculate the selection probability p_i for each chromosome v_i $p_i = \frac{f(v_i)}{F}$

Step 3: Calculate the cumulative probability position q_i for each chromosome: $q_i = \sum_{i=1}^{i} p_i$

Step 4: Perform selection by spinning the Roulette wheel N times. For each spin:

- Generate a random real number r in the range [0,1].

- If $r<q_1$, select chromosome v_1; otherwise, select chromosome v_i (for $2 \le i \le$ pop-size

$2 \le I \le pop\_size$) such that $q_{i-1} \le r \le q_i$

With this selection mechanism, some chromosomes may be selected multiple times. This is consistent with the Schema Theorem, which states that chromosomes with high fitness will have more copies in the next generation, chromosomes with average fitness will maintain their quantity, and chromosomes with low fitness will be gradually eliminated.

a. Ranking Selection Mechanism

The ranking selection mechanism is performed as follows:

Step 1: Sort the chromosomes in the population in ascending order of fitness.

Step 2: Reassign new fitness values to the chromosomes based on their positions in the sorted list, according to the rule:

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

The first chromosome receives a fitness of 1, the second one receives 2, ..., and the last one receives a fitness of *pop_size*.

This method helps reduce the issue of a chromosome being selected too frequently, as can happen in Roulette Wheel Selection. However, it may lead to slower convergence because the fitness difference among the top-ranked chromosomes is relatively small.

b. Stochastic Universal Sampling (SUS) Selection Mechanism

The Stochastic Universal Sampling mechanism is carried out as follows:

Step 1: Represent the selection probabilities of chromosomes on a line segment.

Step 2: Place N selection points evenly spaced at intervals of $\frac{1}{N}$ on this line segment. The first point is placed randomly within the range $[0, \frac{1}{N}]$

Step 3: For each selection point, choose the chromosome located immediately to the right of the point on the number line.

This method ensures that the selection points are evenly distributed along the line segment, thereby enabling a fairer and more accurate selection of chromosomes that deserve to be chosen.

4.5. Classical Genetic Algorithm

4.5.1. Encoding

The classical Genetic Algorithm (GA) uses binary encoding, where each individual is represented by a fixed-length binary string.

Suppose we use a binary vector of length L as a chromosome to represent the value of a variable $x \in [l_x, u_x]$. The length L depends on the specific requirements of the problem. Each bit encoding corresponds to a value in the range $[0, 2^L]$ that will be mapped to a real number in the domain $[l_x, u_x]$.

Scaling factor for mapping: $g = \frac{u_x - l_x}{2^L}$

Real value x corresponding to binary chromosome: $x = l_x + \text{decimal}(chromosome) * g$

Where decimal(*chromosome*) is the decimal value of the binary string.

To initialize the population, randomly generate *pop_size* binary chromosomes bit by bit. Then decode each chromosome to a real value and calculate its fitness function value f. The process of generating a new population involves selection based on fitness-proportional probability, followed by mutation

and crossover to generate the next generation. After several generations, if there is no significant improvement, the best chromosome is considered the optimal solution (often a global optimum). Typically, the algorithm terminates after a fixed number of generations or when computational resources are exhausted.

4.5.2. Selection Operator

a. Roulette Wheel Selection

There are several ways to implement the selection operator, but the simplest and most effective method is roulette wheel selection. In this method, each individual occupies a slot on the wheel, proportional in size to its fitness value.

Assuming f_i is the fitness of the *i-th* individual in a population of size N, the selection probability is:

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i}$$

b. Ranking-Based Selection

In this procedure, individuals are sorted by their objective function values:

- The best individual is placed first; the worst is placed last.
- The $(N-j)$-th individual has the selection probability: $p_{N-j} = \frac{j}{\sum_{k-1}^{N} k}$

Steps:

- Sort individuals in descending order (for maximization) or ascending order (for minimization).
- Assign new fitness values based on their ranks.

c. Tournament Selection

The tournament selection procedure works as follows:

* Randomly select *t* individuals from the population. Choose the best one among them to copy into the temporary population.

* Repeat the process **N** times to create the new population.

The parameter *t* is called the tournament size. When t=2, it is known as binary tournament selection.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

4.5.3. Crossover Operators

a. Single-Point Crossover

Assume two selected parent individuals P_1, P_2:

P_1=(1011011111), P_2=(0111001001) with length L=10

Randomly choose a crossover point k such that 1< k <L, and exchange genes after position k=7.

the offspring are: C_1=(1011011001), C_2=(0111001111)

Pseudo-code:

```
Procedure lai1diem (k, P1, P2; var C1, C2);
Begin
    For i :=1 to k do
        Begin
            C1[i] := P1[i];
            C2[i] := P2[i];
        End;
    For i := k+1 to L do
        Begin
            C1[i] := P2[i];
            C2[i] := P1[i];
        End;
End;
```

b. Multi-Point Crossover

Multi-point crossover is performed similarly to single-point crossover. Given two selected parent individuals, P_1 and P_2, this operator randomly generates k crossover points: i_1, i_2, …, i_k (assuming $i_1 < i_2 < \ldots < i_k$). These crossover points divide the parent chromosomes into segments that are alternately numbered as even and odd. Two offspring are then created by exchanging the genes in either the even-numbered or odd-numbered segments between the parents.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

Example: Let P_1 = (1001110101) and P_2 = (0100111110)

These are binary strings of length 10. Suppose the selected crossover points are 2, 4, 6, and 9. Then the two offspring generated are:

C_1 = (1000111111) and C_2 = (0101110100).

The procedure for multi-point crossover is written in a similar way to the single-point crossover method.

c. Mask Crossover

From two selected parent individuals P_1 and P_2, a random binary string of the same length L is generated — called the mask string.

Let L = 10, P_1 = (1010111001), P_2 = (0101110100).

The offspring are created based on this mask string to determine whether to take the component from the father or the mother individual, according to the following rule:

The i-th gene of offspring C_1 is taken from the i-th gene of P_1 if the corresponding bit in the mask is 1, and from the i-th gene of P_2 if the mask bit is 0. For C_2, it is the opposite.

C_1 = (1010110001), C_2 = (1111011010).

Mask Crossover Procedure (pseudo-code):

```
Procedure laimatna (U, P1, P2; var C1, C2);
Begin
    For i := 1 to L do
      Begin
        If U[i] = 1 then
            Begin
                C1[i] := P1[i]; C2[i] := P2[i];
            End
        Else
            Begin
```

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

$C_1[i] := P_2[i]; C_2[i] := P_1[i];$

End;

End;

End;

2.5.4. Mutation Operator

The mutation operator alters the information of the population at the bit (gene) level. Mutation changes the value of any given bit with a probability of pm. Each bit in an individual has an equal chance of being mutated.

Mutation operators include: standard mutation, uniform mutation, non-uniform mutation, and non-uniform mutation (redundant term – likely a typo in the original).

Mutation algorithm:

For i := 1 to m do

If random [0,1] < p_m then invert(parent[i]);

Where invert(u) is the function that flips bit u.

The classical Genetic Algorithm (GA) can be explicitly described as follows:

Algorithm GA_codien;

Begin

{Initialize a population consisting of m individuals}

t := 0;

Creat initialize population $B_0 = (b_{10}, b_{20}, ..., b_{m0})$;

While (stopping condition not fulfilled) do

Begin

{Proportional selection}

For i := 1 to m do

Begin

r := random(0,1);

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

k := 1;

While (k < m and r < $\frac{\sum_{j:=1}^{i} f(b_j,t)}{\sum_{j:=1}^{m} f(b_j,t)}$

$B_{i,t+1}$:=b_k

End;

{Single-point crossover}

For i := 1 to m -1 step 2 do

Begin

If random [0,1] < p_c then

Begin

pos := random(1, n-1);

for k := pos+1 to n do

Begin

aux := b_{it+1}[k];

b_{it+1}[k] := aux;

End;

End;

End;

{Đột biến}

For i := 1 to m do

For k := 1 to n do

If random [0,1] < p_m then

Invert b_{it+1}[k]

t := t + 1;

End;

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

2.6. Real-Coded Genetic Algorithm (RCGA)

2.6.1. Introduction to RCGA

An RCGA problem is defined as a pair (S, f), where $S \subseteq R^n S$ and $f:S \rightarrow R$ is a function of n variables. The goal is to find a vector $x = (x_1, x_2, \ldots, x_n) \epsilon S$ such that f(x) reaches a minimum value over S. This means that for every $y \epsilon S$, we must have $f(x) \leq f(y)$. The function fff is not continuous but is bounded on S.

In RCGA, each individual is represented by a real-valued n-dimensional vector:

$b=(x_1, x_2,...,x_n)$, $x_i \in R$

Thus, a population of size m is a set of m real vectors in R^n or equivalently, a real-valued matrix of size $m \times n$. This encoding is natural and convenient for applying evolutionary operators.

2.6.2. RCGA Operators

a. Selection Operator

Selection in RCGA is applied similarly to classical GA, including: Proportional Selection, Rank Selection, Tournament Selection

b. Crossover Operators

RCGA applies similar crossover methods as classical GA such as: One-point crossover, Multi-point crossover, Mask crossover (or uniform crossover)

Due to real-number encoding, RCGA also uses specialized crossover types. Assume two selected parents:

$X=(x_1,...,x_m)$ và $Y=(y_1,...,y_m)$.

+ One-point Crossover

Choose a random crossover point k $(1 \leq k \leq m)$, then generate children as:

$X'=(x_1,...,x_k,y_k+1,...,y_m)$,

$Y'=(y_1,...,y_k,x_k+1,...,x_m)$.

+ Multi-point Crossover

Select k random points $j_1,j_2,...,j_k$ $(1 \leq j_1 < j_2 <\ldots< j_k <m)$ Label the segments between [jt,jt+1][j_t, j_{t+1}][jt,jt+1] starting from 0. Then:

- $x_i' = x_i$ for even segments, $x_i' = y_i$ for odd segments.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

- $y_i' = x_i$ for even segments, $y_i' = x_i$ for odd segments.

+ Mask Crossover (Uniform Crossover)

Randomly choose k positions: $1<i_1<i_2<\ldots<i_k<m$.

Generate children:

$$x_i' = \begin{cases} x_i & i \in \{i_1, \ldots, i_k\} \\ y_i & i \notin \{i_1, \ldots, i_k\} \end{cases} \quad y_i' = \begin{cases} y_i & i \in \{i_1, \ldots, i_k\} \\ y_i & i \notin \{i_1, \ldots, i_k\} \end{cases}$$

+ Arithmetic Crossover

Choose a real number aaa such that (0<a<1).

Children are computed as:

$x_i' = a*x_i + (1 - a) * y_i'$;

$y_i' = a*y_i + (1 - a) * x_i'$;

+ Heuristic Crossover

Given parents (X, Y) where X is fitter than Y, generate a single child X′:

$x_i' = r * (x_i - y_i) + x_i$ với $a < r < 1$

+ Blend Crossover (BLX-α)

Given two parents:

$X = (x_1, \ldots, x_k, y_{k+1}, \ldots, y_m)$.

$Y = (y_1, \ldots, y_k, y_{k+1}, \ldots, x_m)$.

Then each gene of the offspring is randomly selected from:

$[\min \min(x_i, y_i) - I * \propto, \max(x_i, y_i) + I * \alpha]$.

Experiments show best performance when α=0.5.

+ UNDX (Unimodal Normal Distributed Crossover)

In UNDX-m, (m+2) parents are randomly selected from the population, in which the first (m+1) individuals are used to compute a mean vector. The final individual is used to determine the direction of the offspring to be generated.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

The algorithm is described as follows:

1. Randomly select (m+1) parent individuals $x^1, x^2, \ldots, x^{m+1}$ from the initial population.
2. Compute the mean vector of the selected individuals: $u = \frac{1}{m+1}\sum_{i=1}^{m+1} x^i$.

   Then compute $d^j = x^i - u$.
3. Randomly select the x^{m+2}-th parent from the population.
4. Compute the vector $d^{m+2} = (x^{m+2} - u)$, which is orthogonal to $d^1, \ldots, d^{m+1}$, and its length.
5. Find an orthogonal basis $e^{m+2}, \ldots, e^n$ for the subspace orthogonal to the subspace spanned by $d^1, \ldots, d^{m+1}$.
6. Generate offspring using the following formulas:

$x^{c1} = u + \sum_{i=1}^{m+1} w_i\, d^i + \sum_{j=m+2}^{n} v_i\, de^j$

$x^{c2} = u + \sum_{i=1}^{m+1} w_i\, d^i - \sum_{j=m+2}^{n} v_i\, de^j$

Where w_i and v_j are random variables sampled from the normal distributions: N (0, $\sigma \frac{2}{\eta}$) and N (0, $\sigma \frac{2}{\xi}$).

Mathematicians **Kita and Yamanura** successfully used the following parameter values:

$$\sigma_\xi = \frac{1}{\sqrt{m}} \text{ và } \sigma_\eta = \frac{0,35}{\sqrt{n-m}}$$

+ CMX (Center of Mass Crossover) Assume the current population is $\{X_1,\ldots,X_N\}$, each $X_i \in R^n$

1. Randomly select *m* individuals:
2. $X^{pi} = (x_1^{pi}, \ldots, x_n^{pi})$ i = 1,…, m
3. For each i = 1,…, m, compute the "virtual parent" vectors X^{vi} that are symmetric to X^{pi} with respect to X_{CM}^p: $X^{vi} = 2X_{CM}^p - X^{pi}$

4. For each pair (X^{vi}, X^{pi}), generate child using BLX-α crossover.

Thus, *m* parents yield *m* offspring.

The simulation of CMX crossover is illustrated in Figure 2.

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

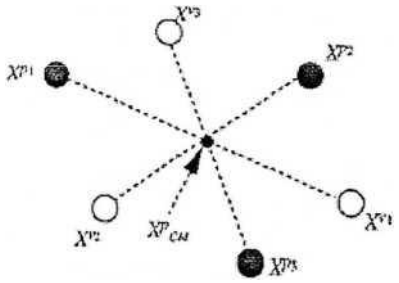

Figure 2: Simulation of CMX Crossover

With this crossover method, the offspring will be distributed close to the "center" of the group of parent individuals.

Multi-parent Feature-wise Crossover (MFX)

This algorithm selects a characteristic feature from each parent individual and then evolves using this feature information.

The algorithm is described as follows:

Randomly select m individuals from the population (1 < m < N):

$X^{pi} = (x_1^{pi} x_n^{pi})$ $i = 1, \ldots, m$

2. The offspring $X^{ci} = (x_1^{pi}, \ldots, x_n^{pi}$ is generated as follows:

For each j, j = l,..., n, randomly select an integer k from (1, m) such that $k \neq$ i. Then, x_j^{ci} is generated randomly in the interval: $(x_x^{pi} - \alpha l, x^{pk} + \alpha l)$. where l is the distance between x_j^{pi} and x_j^{pk}

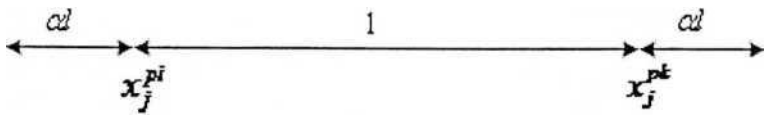

Figure 3: Distribution of x_j^{ci}

By the above method, if a parent X^{pi} is fixed, the offspring X^{ci} will be generated such that each component of the offspring vector inherits a characteristic feature from the parent X^{pi} and another parent X^{pk}

Seed Crossover (SX)

This algorithm performs pairwise crossover between individuals, where the offspring produced will continue to crossover with parents having progressively higher fitness. The algorithm is as follows:

1. Randomly select *m* individuals from the population (1 < m < N):

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

$X^{pi} = (x_1^{pi}, ..., x_n^{pi})$ $i = 1,...,m$

2. Rank the parent individuals in descending order by fitness.

3. $X^c = Blx(X^{pm}, X^{pm-1})$

4. For j running from m-2 down to 1, perform: $X^c = Blx(X^c, X^{pj})$.

where *Blx(X,* Y) is the BLX-α crossover operator applied to two parents.

With this method, at each iteration, the newly created individual is able to inherit characteristics from parents with increasing fitness. This operator generates only one offspring from the *m* selected paren.

Cha mẹ con

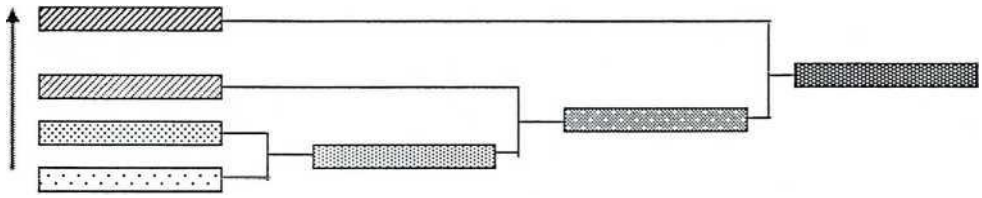

Figure 4: Seed Crossover operator sx

c. Mutation Operators

+ Uniform Mutation

For a randomly chosen gene i to mutate from an $b = (x_1, x_2,..,x_N)$,the component x_i is replaced by a random number within the defined range $[l_i, u_i]$ for x_i

+ Boundary Mutation

Given a selected parent individual x and a mutation position k, the kkk-th component $k(x_k)$ of x is replaced by either the lower bound lkl_klk or the upper bound uku_kuk, where $[l_k, u_k]$ is the defined range for X_k

In problems where the variable boundaries are small and the solution is expected near the boundaries, boundary mutation proves to be very useful.

+ Non-uniform Mutation

Assume t_{max} is a predefined maximum number of generations. The component x_i is replaced by one of the two values calculated as follows:

$x_i' = x_i + \Delta(t, b_i - x_i)$

International Journal of Education and Social Science Research

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

$x_i'' = x_i + \Delta(t, x_i - a_i)$

The choice between these two values is made randomly with a probability of 1/2. The random variable Δ(t,x) determines the mutation step within the interval [0,x] and is given by:

$$\Delta(t,x) = x * (1 - \lambda)^{(1-\frac{1}{tmax})\tau}$$

where:

λ is a random parameter uniformly distributed in the unit interval [0,1].

τ is a parameter controlling the influence of generation t on the mutation distribution within the range [0, x].

5. CONCLUSION

Genetic algorithms, based on the principle of evolution through natural selection, have proven to be important tools for solving complex optimization problems. By simulating genetic processes such as crossover and mutation, these algorithms demonstrate effective search capabilities in large solution spaces.

The effectiveness of genetic algorithms is evident in many application fields such as engineering optimization, finance, artificial intelligence, and computational biology. However, to achieve the best results, it is necessary to select suitable parameters and combine genetic algorithms with other optimization methods to improve convergence speed and accuracy.

Acknowledgement

This research is funded by Tan Trao University in Tuyen Quang, Viet Nam.

REFERENCES

1. Adam Marczyk (2004). *Genetic Algorithms and Evolutionary Computation*.
2. David A. Coley. *An Introduction to Genetic Algorithms*.
3. Eiben, E. et al. (1994). Genetic algorithms with multi-parent recombination. In *PPSN III: Proceedings of the International Conference on Evolutionary Computation* (pp. 78–87).
4. Charbonneau, Paul (1995). Genetic algorithms in astronomy and astrophysics. *The Astrophysical Journal Supplement Series*, *101*.
5. Nguyen Duc Nghia (1999). *Optimization (Linear and Discrete Programming)*, Education Publishing House, Hanoi.
6. Bui The Tam & Tran Vu Thien (1998). *Optimization Methods*, Transport Publishing House, Hanoi.

ISSN 2581-5148

Vol. 8, Issue.3, May-June 2025, p no. 354-373

7. Bui Minh Tri (2001). *Mathematical Programming*, Science and Technology Publishing House, Hanoi.
8. Nguyen Anh Tuan (2011). *Quasi-convex and quasi-concave programming applied to linear programming*, Science and Technology Publishing House, Hanoi.