

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,800

Open access books available

142,000

International authors and editors

180M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Introduction to Evolutionary Algorithms

S. Tamilselvi

Abstract

Real-world has many optimization scenarios with multiple constraints and objective functions that are discontinuous, nonlinear, non-convex, and multi-modal in nature. Also, the optimization problems are multi-dimensional with mixed types of variables like integer, real, discrete, binary, and having a different range of values which demands normalization. Hence, the search space of the problem cannot be smooth. Evolutionary algorithms have started gaining attention and have been employed for computational processes to solve complex engineering problems. Because it has become an instrument for research scientists and engineers who need to apply the supremacy of the theory of evolution to shape any optimization-based research problems and articles. In this chapter, there is a comprehensive introduction to the optimization field with the state-of-the-art in evolutionary computation. Though many books have described such areas of optimization in any form as evolution strategies, genetic programming, genetic algorithms, and evolutionary programming, evolutionary algorithms, that is, evolutionary computation is remarkable for considering it to discuss in detail as a general class.

Keywords: evolutionary algorithms, genetic operators, non-convex, multi-modal, optimization process

1. Introduction

Darwin's principle of evolution says that the existence of any creature is based on the law "strongest creature survives." Before computers have entered the human world, in the 50s, knowledge to apply Darwinian principles for automated problem solving was invented. Darwin also proved that the survival of any organism can be maintained with genetic inheritance, such as reproduction, crossover, and mutation. Thus, Darwin's evolution theory was deployed by computational optimization algorithm to search for a solution to any real-world optimization problem in a natural way [1].

In the 60s, three various interpretations of this idea were introduced at different places. Evolutionary programming was developed by Lawrence J. Fogel in the USA when John Henry at Holland started his methodology as a genetic algorithm,

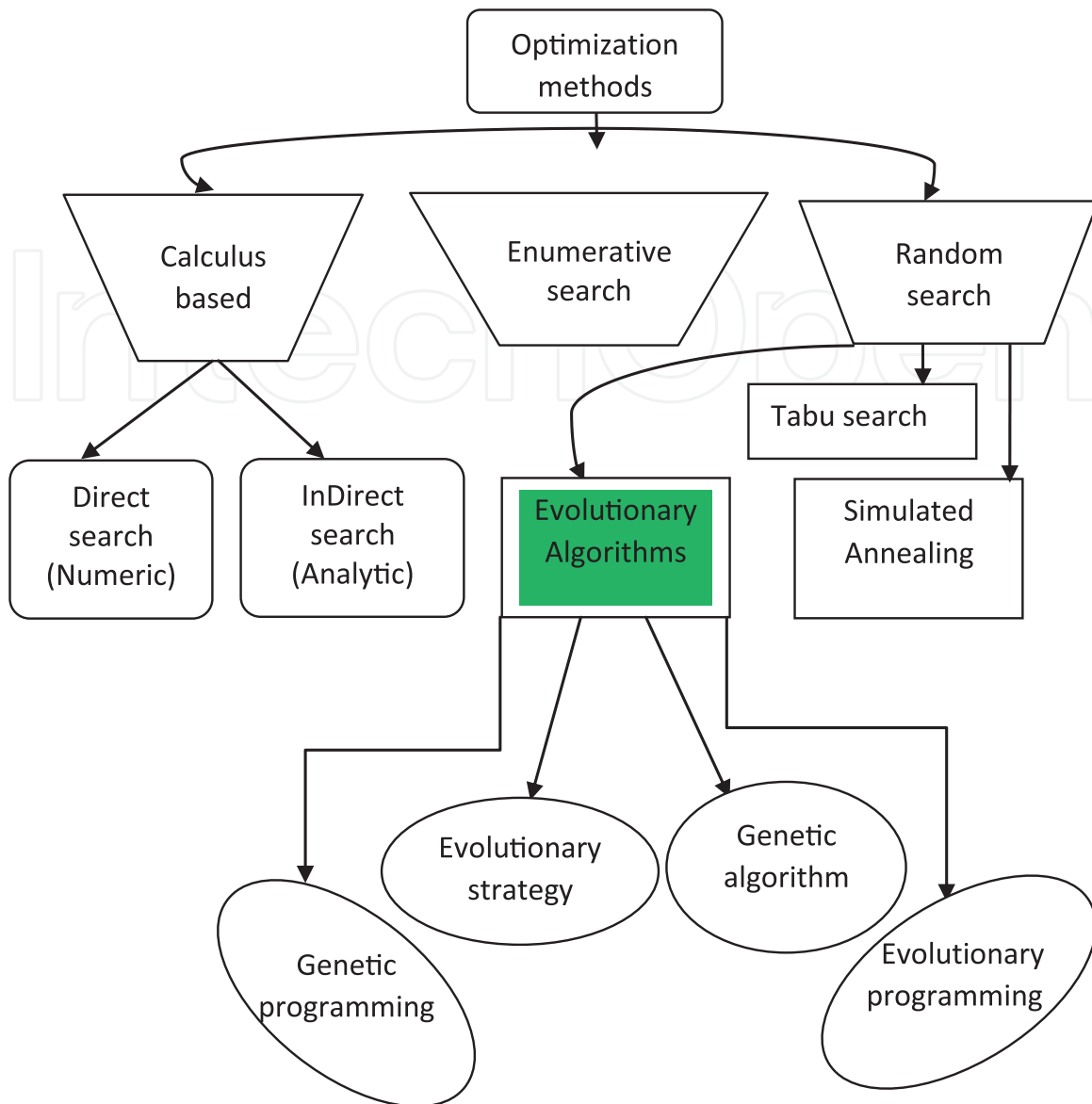


Figure 1.
Evolutionary algorithms and their subtypes.

stimulated by Darwin’s evolutionary concepts. Similarly, Ingo Rechenberg and Hans-Paul Schwefel have invented evolution strategies in Germany. Following this fourth one had emerged as genetic programming, in the early 90s. These four different terminologies are seen as different representatives of one technology called evolutionary algorithms (EAs), which denote the whole field by considering evolutionary programming, evolution strategies, genetic algorithms, and genetic programming as sub-areas and is well depicted in **Figure 1** [1, 2].

2. Need for evolutionary algorithms

Real-world has many optimization scenarios. Optimization, by definition, is a methodology of making the decision as fully perfect as possible to achieve the maximum possible goal, in an engineering system. Nature is a very good optimizer. An optimization problem can be stated as follows [2, 3].

Find $x = \{x_1, x_2, \dots, x_n\}$, which

Minimize/Maximize $f(x)$

Subject to

$$g_j(x) \leq 0, j = 1, 2, \dots, m$$

$$h_j(x) = 0, j = 1, 2, \dots, p$$

Any engineering system can be represented with a set of quantities. Certain quantities are usually fixed called as pre-assigned constants. Remaining quantities can be treated as decision variables in the optimization process, $x_i = \{x_1, x_2, \dots, x_n\}$. ' $f(x)$ ' is the objective function or goal to be attained, ' $g_j(x)$ ' represent ' m ' the count of inequality constraints and ' $h_j(x)$ ' represent ' p ' count of equality constraints to be satisfied for attaining feasibility [3].

In real-world engineering problems, the objective function is discontinuous, nonlinear, non-convex, and multi-modal. Also, the problems are multi-dimensional as the number of design variables are more and they are mixed in type like integer, real, discrete, binary. Hence, the search space is not smooth. It may require accessing look-up table data for objective function evaluation. The constraint functions are very complex and the amount of violation of each constraint does not cover the same range, which requires normalization [4].

In general, the optimization problems are categorized based on the existence of constraints, nature of the decision variables, permissible values of the design variables, nature of the equations involved, deterministic nature of the variables, separability of the functions, number of objective functions, etc. Some of them are static optimization problem, dynamic optimization problem, linear programming problem, convex programming problem, nonlinear programming problem, geometric programming, quadratic programming problem, separable programming problem, multi-objective optimization problem, single-variate optimization problem, multi-variate optimization problem [5].

Two different techniques to find the solution for optimization problems are mathematical programming techniques and meta-heuristic techniques. When derivative-based mathematical programming methods are applied in solving nonlinear programming problems, there are several shortcomings. Traditional optimization methods:

- Yields results that are caught at premature convergence, that is, local optima often, due to the search space with multi-modality.
- Requires mathematically well-defined objective and constraint functions.
- Requires existence of derivatives for objective function and constraint functions.
- Find difficulty to handle mixed variables.

For a real-world optimization problem, the surface plot obtained even for optimizing two design variables gives greater number of local minima/maxima. **Figure 2** depicts the surface plot obtained for the design optimization of the distribution transformer problem with two decision variables, width of the core leg, height of the core window, by minimizing the transformer lifetime cost (TLTC) of the transformer. It is

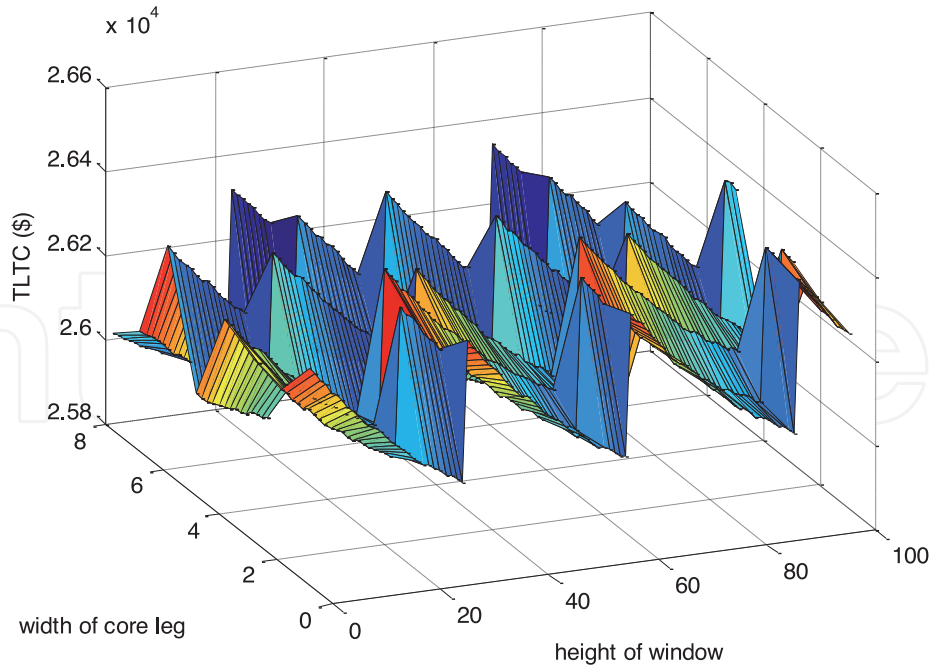


Figure 2.
Search space for minimization of TLTC objective optimizing two decision variables.

very clear from **Figure 2** that the real-world problem is very complex with its multi-modal search space [3].

Of course, when it is a multi-dimensional engineering problem, which requires optimization of more decision variables, the multi-modality cannot be imagined. So, the conventional derivative-based mathematical programming technique cannot handle such complex nature of the optimization problem, accurately. It may yield a feasible design; however, it will not be an optimal solution. If there is no knowledge or little knowledge about the behavior of the objective function related to the presence of local minima, location of feasible region, infeasible region in the multi-dimensional parameter space, it is advisable to start the meta-heuristic technique, which is a stochastic strategy [2, 3].

Of all the meta-heuristic techniques, evolutionary algorithms (EAs) are especially effective in the solution of high-complexity, non-convex, nonlinear, multi-dimensional, mixed variable, multi-objective, constrained optimization problems, for which a traditional mathematical model is difficult to build, where the nature of the input/output relationship is neither well defined nor easily computable. The stages of EAs have not yet been investigated in detail steps with illustration, despite their performances are better in terms of convergence, consistency in obtaining the solution, and computational speed in solving any multi-modal problems.

Hence, this chapter discusses in detail the step-by-step evolutionary process that happens behind the optimization algorithm. It highly helps to find solution for any multi-modal real-world engineering optimization problem, by optimizing design objective, while satisfying simultaneously various constraints imposed by international standards and user specifications.

3. Known optimization problems

Evolutionary optimization algorithms minimize or maximize an objective function and they are search algorithms. The algorithm checks all the way through a large

search space of possible solution set for the optimal best solution. In day-to-day practical life as well as professional life, there are numerous activities that seek optimization. Some of the common well-known real-world optimization problems are the traveling salesman problem, classification problem, economic power dispatch, base station location problem, antenna design, scheduling problem, etc.

In traveling salesman problem, a salesman wants to visit all the towns, with information of list of towns and distances between all the towns. The constraint is that each town has to be visited only once. The optimization problem statement is searching for the optimum shortest distance/route that the salesman travel and visits each town exactly only once and returns to the place where he started [6].

Base station location problem is setting radio and optimizing maximum coverage. Given a set of spots for installing base stations, feasible configurations for every base station, antenna tilt, maximum power, antenna height, sectors orientation, etc. along with the information of traffic and strength of the signal propagation, the optimization algorithm is to choose the right location and appropriate configuration of the base station such that the installation cost is minimum while meeting the traffic demand simultaneously [7].

The job of optimal generator maintenance scheduling problem is to find out the optimum period for which, the generator units must be taken offline for maintenance over the stipulated time horizon, so that the operating costs involved are minimized, meeting the maintenance constraints during the considered time period such as load demand, maintenance window, maintenance duration and manpower availability [8].

Previous research works have applied only machine learning techniques for the prediction and classification of any disease/tumor. However, nowadays due to the capability of evolutionary algorithms, such classification problems have been stated as an optimization problem and solved using the integrated machine learning-optimization technique. Thus, optimal classification problem aims to select optimum elite features from intelligent liver and kidney cancer diagnostic systems of huge data sets, by filtering the redundant features, minimizing the error rate, in order to improve the quality of heart disease classification [9].

Economic power dispatch is a vital optimization problem in power system planning. The aim of the economic dispatch is to schedule the optimum power output for the available generator units of the power system such that the production cost is minimum and power demand is met [10].

4. Optimization process of simple evolutionary algorithm

EA handles a population of possible random solutions. Each solution is represented through a chromosome. The fitness of each chromosome is calculated to call for a competition among the chromosomes. Competition results in the selection of those better chromosomes/solutions (with high fitness value) that are suited for the environment. The process of the first level of filtration based on the fitness value is called parent selection [3]. The selected individuals, that is, parents act as seeds for creating children through genetic inheritance, that is, recombination and mutation. These genetic operators aid the necessary diversity. Few pairs of chromosomes from the parent pool are chosen based on the random probability to undergo crossover for forming offspring. The resulted offspring individuals obtained after crossover are allowed to take up mutation randomly. Different regions of the search space are explored for identifying possible optimal individuals through “recombination and mutation” operation known as “exploration.” The

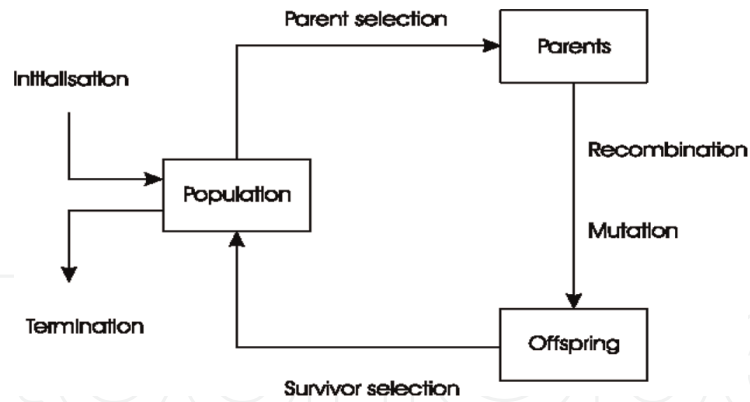


Figure 3.
Working of evolutionary algorithms.

new individuals/children thus formed have their fitness evaluated to compete for survival in the next generation. As an end to an iteration, in a replacement stage, 80% worst solutions of the initial random population are substituted by the best offspring children filtered after survival selection process based on the evaluated fitness value. Over time and several iterations, “natural selection” operation, which is called exploitation leads to the identification of an individual in the population as global optimum. The complete working of the evolutionary algorithm is pictured in **Figure 3** [11]. The major steps involved in the process of optimization in an evolutionary algorithm are as follows [1, 3].

- Solution representation
- Random population generation
- Fitness function evaluation
- Parent selection
- Reproduction—(crossover, mutation)
- Survival selection
- Replacement
- Stopping criteria

5. Iterative process behind evolutionary algorithms

To define problem statement:

Consider an equality function, $x + 2y + 3z + 4u = 30$. We shall apply the evolutionary algorithm to find the appropriate values for x, y, z, u , such that the equity equation gets satisfied [12].

5.1 Formulation of optimization problem

- a. Formulate objective function: $f(k)$

The objective/aim is to **Minimize** $f(k) = [(x + 2y + 3z + 4u) - 30]$.

b. Identify decision variables/type: In this equity problem, there are four decision variables $[x, y, z, u]$. Variables that possess a larger influence on the objective function and constraint functions are appropriate ones to be chosen as decision variables.

c. Find problem dimension:

Total number of decision variables is the problem dimension = 4.

d. Representation:

A solution generated by an evolutionary algorithm is called a chromosome/individual, which is made up of genes [1]. After selecting the decision variables, and problem dimension, choice of suitable type for these variables is another important task. The nature of the decision variables is completely problem dependent and thus in this example, $[x, y, z, u]$ —they are integers. However, the genes can be mixed like binary, real, integer, discrete variables, etc., depending upon the need of the problem under consideration. Chromosome/solution is thus represented as an integer variable as under:

x	y	z	U
-----	-----	-----	-----

e. Impose boundary constraint:

This range selection for setting the search space is more often done on a trial basis, in case the problem dimension is high. On contrary, if the objective function is very simple, clear, and possesses straight relationship (mathematical equation) with lesser number of decision variables, then the search space can be decided by inspection [2]. For this example, it is very clear that the integer values of decision variables $[x, y, z, u]$ can be restricted to vary between 1 and 30, in order to speed up the computational search.

5.2 Different stages in optimization process

To illustrate solving a minimization type optimization problem using EA, integer type for decision variables, six for population size, single-point method for crossover, and roulette wheel for selection are assumed. The various stages involved in the process of optimization are given for one iteration in this section [13–15].

Stage 1: Population generation: Initial solution set

Four genes $[x, y, z, u]$ are generated randomly satisfying the lower and upper limits of the boundary constraint. A chromosome thus generated is a vector comprising of four genes. Chromosome refers to the solution/individual of the formulated optimization problem, while the collection of such chromosomes is referred to as a population. For example, Solution [1] = $[x; y; z; u] = [12, 05, 23, 08]$. Then, the initial population will have an array of sizes [population size, problem dimension]. That is, $[(6, 4)]$ as shown in **Table 1**.

Stage 2: Function evaluation: Feval

All the chromosomes of random population will then go through a process known as fitness evaluation to measure the quality of the solution created by EA. Evaluation

of fitness value of chromosome/solution is carried out by calculating the objective function value as $Feval = \text{Modulus}[f(k)]$.

$$f(k) = [(x + 2y + 3z + 4u) - 30].$$

$$\begin{aligned} \text{For Solution [1], Feval [1]} &= \text{Modulus [f(Solution [1])]} \\ &= \text{mod} [(12 + 2 \times 5 + 3 \times 23 + 4 \times 8) - 30] \end{aligned}$$

$Feval [1] = \text{mod} [(12 + 10 + 69 + 32) - 30] = 93$. Similarly, the solutions of entire population can be calculated and tabulated as under in **Table 1**. It is found that Chromosome 4 has the least objective function value 46.

Stage 3: Parent selection

The chromosomes are selected from the initial population to act as parent for reproduction, based on the fitness of the solution/individual. The selection procedure tells how to choose individuals in the population that will create offspring for the next generation. The fittest solution will have a higher probability to be selected as a parent. Two-step selection process is discussed as follows [13, 15].

A. To compute the probability of selection: $\text{Prob}[i]$

$$\text{Prob}[i] = \frac{\text{Fit}[i]}{\sum_{i=1}^6 \text{Fit}[i]}$$

where,

$$\text{Fit}[i] = \frac{1}{(1 + \text{Feval}[i])}$$

(to avoid undefined divide by zero error, which may encounter for the optimal solution, it is advisable to add 1 with Feval).

$$\text{Fit [1]} = 1/(1 + \text{Feval [1]}) = 1/94 = 0.0106 \text{ and so on, till } i = 6.$$

$$\text{Total fitness} = 0.0845 \text{ (refer Table 2).}$$

$$\text{Prob [1]} = 0.0106/0.0845 = 0.1254$$

B. To select the parent pool: Roulette-wheel (RW) selection process:

Parent selection is vital for the convergence of optimization algorithm as efficient parents force solutions/individuals to optimality. There are different methods in the

	Initial random population				Feval [k]	Remarks
Solution [1]	12	05	23	08	93	
Solution [2]	02	21	18	03	80	
Solution [3]	10	04	13	14	83	
Solution [4]	20	01	10	06	46	Best Solution
Solution [5]	01	04	13	19	94	
Solution [6]	20	05	17	01	55	

Table 1.
Functional evaluation of initial population.

	Initial population			Feval [i]	Fit [i]	Prob [i]	Cum [i]
12	05	23	08	93	0.0106	0.1254	0.1254
02	21	18	03	80	0.0123	0.1456	0.2710
10	04	13	14	83	0.0119	0.1408	0.4118
20	01	10	06	46	0.0213	0.2521	0.6639
01	04	13	19	94	0.0105	0.1243	0.7882
20	05	17	01	55	0.0179	0.2118	1
Total fitness					0.0845		

Table 2.
 Selection probability computation.

process of selecting parents such as stochastic universal sampling, fitness proportionate selection, tournament selection, rank selection, and random selection. In this chapter, roulette wheel selection has been implemented for identifying the right parent pool.

Consider a wheel that is split into ‘6’ pies. Pie refers to the individual in the population. Each solution occupies a portion of the wheel, proportional to its fitness value. It is clear that a fitter solution takes a larger pie on the wheel and has larger probability chance of being selected as a parent when the wheel is made to spin ‘6’ times. Hence, the probability of choosing a chromosome depends on its fitness only. The steps involved in the roulette wheel selection process are:

- Compute cumulative probability values for all the solutions—Cum[i].
- Allot pie in sequence for all the ‘6’ individuals, based on the cumulative probability. That is, Chromosome 1 has occupied light blue pie with cumulative probability ranging between [0–0.1254]. Chromosome 4 which has the highest fitness value ‘0.0213’ has the highest probability ‘0.2521’ among all the solutions of the population. Naturally, it will take larger sized pie, which is yellow in color on the wheel. It is clearly explained in **Figure 4**, and **Tables 3** and **4**.
- To arrange the order of chromosomes (6) in the parent pool, equivalent to spinning the wheel ‘6’ times, generate random number ‘6’ times, ‘rand[i]’ < 1, six.

rand [place 1] = 0.2; rand [place 2] = 0.285; rand [place 3] = 0.098;

rand [place 4] = 0.812; rand [place 5] = 0.397; rand [place 6] = 0.50.

- Fit the random number of each place in the respective range of cumulative probabilities and fetch the color of the pie. For example, rand [place 2] = 0.285, which is between Cum [2] and Cum [3]. So, the gray pie which is individual/ chromosome [3] will occur in place2 of the parent/mating pool.

Stage 4: Crossover

The crossover operation involves three steps: (A) selecting mating chromosomes, (B) determining cut point for crossover, and (C) updating the population.

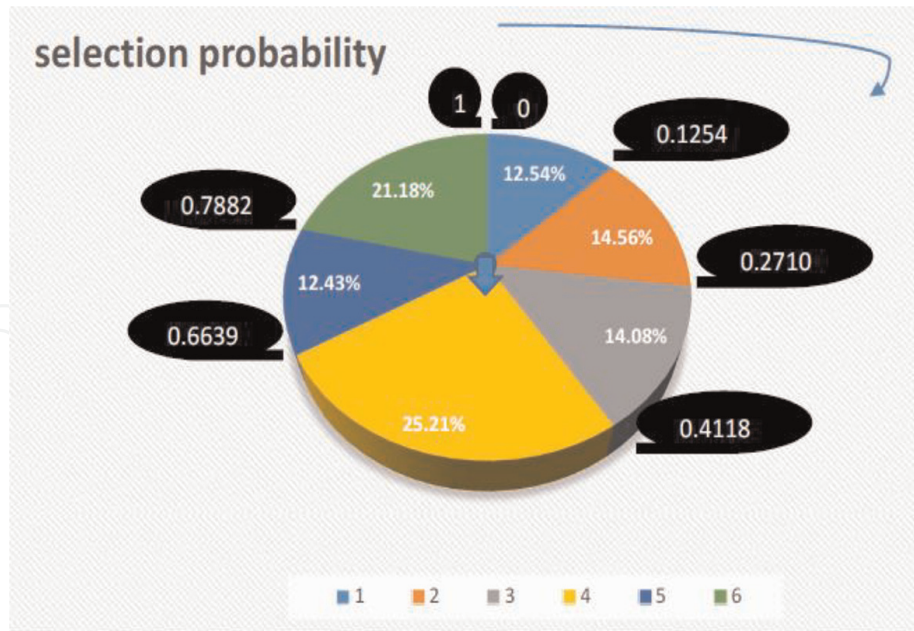


Figure 4. Roulette wheel—Parent selection process.

Position	Chromosome			Initial population		
Place 1	I	Solution [1]	12	05	23	08
Place 2	II	Solution [2]	02	21	18	03
Place 3	III	Solution [3]	10	04	13	14
Place 4	IV	Solution [4]	20	01	10	06
Place 5	V	Solution [5]	01	04	13	19
Place 6	VI	Solution [6]	20	05	17	01

Table 3. Place and position of solution—Before RW selection process.

In this operation pairs of parents are chosen and many children/off-springs are generated using the information available in the gene of the parents. Usually, crossover operation is deployed in EA with high probability (p_c). Some of the commonly used crossover operators are whole arithmetic recombination, one point crossover, uniform crossover, multi-point crossover, Davis’ order crossover, etc. In the equity problem chosen for illustration, one-point crossover has been used for offspring creation. In one-point crossover, a randomly point of crossover has been chosen and the tail ends of the parent pairs are swapped to produce new children. The process is evident in **Table 5**.

A. To select chromosome:

Parent chromosome from parent pool that undergoes the mating process is randomly selected and the number of mate solutions is decided using crossover rate, p_c . Solution ‘ i ’ will become a parent, if random number, $\text{rand}[i]$ falls below the crossover rate. Let us assume the $p_c = 25\%$ for solving the problem. Generate number randomly ‘6’ times (population size) below 1.

Position	Chromosome	Population after selection			
Place 1	II	02	21	18	03
Place 2	III	10	04	13	14
Place 3	I	12	05	23	08
Place 4	VI	20	05	17	01
Place 5	III	10	04	13	14
Place 6	IV	20	01	10	06

Table 4.
 Chromosomes in the MATING POOL after RW selection process.

Population after selection				Population after crossover			
02	21	18	03	02	05	17	01
10	04	13	14	10	04	13	14
12	05	23	08	12	05	23	08
20	05	17	01	20	04	13	14
10	04	13	14	10	04	18	03
20	01	10	06	20	01	10	06

Table 5.
 Population after and before crossover operation.

rand [1] = 0.19; rand [2] = 0.249; rand [3] = 0.750; rand [4] = 0.005
 rand [5] = 0.149; rand [6] = 0.320

Thus, for the generated random numbers, three chromosomes/solutions [1, 4, 5] are selected for crossover operation. Hence, the number of crossovers becomes 3, that is, 3 pairs.

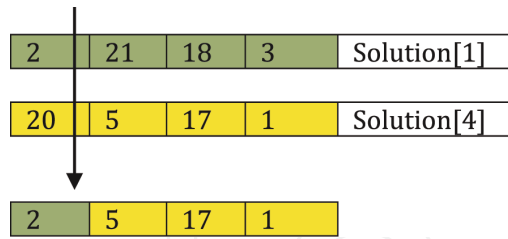
Solution [1] >< Solution [4] — First Crossover
 Solution [4] >< Solution [5] — Second Crossover
 Solution [5] >< Solution [1] — Third Crossover

B. To determine cut point:

Followed by mating chromosome selection, the next phase is to determine the position of the crossover point. The steps involved are:

- Generate random numbers between 1 to (Problem dimension—1) in order to get the crossover point. That is, between 1 and 3. Assume, Cut [1] = 1; Cut [2] = 1; Cut [3] = 2
- Parent individuals get cut at the crossover point and their genes are interchanged. For first, second, and third crossovers, parents' genes are cut at positions 1, 1, and 2, respectively.

First Crossover: New Chromosome [1] = Solution [1] >< Solution [4]



Second Crossover: New Chromosome [4] = Solution [4] >< Solution [5]



Third Crossover: New Chromosome [5] = Solution [5] >< Solution [1]



C. To update the population after crossover:

Stage 5: Mutation

Mutation operation is a small random sharp change in the chromosome necessary to obtain a new solution. Mutation is used to sustain population diversity and it generally has a low probability, p_m . Mutation in EA refers “exploration” of search space. It has been proven that mutation operation is crucial for the convergence of the algorithm, however, crossover operation is not so. Some of the commonly used mutation operators are bit flip mutation, swap mutation, scramble mutation, random resetting, inversion mutation, etc. Like the crossover operators, this is not an extensive list since EA designer may deploy a hybrid approach as a combination of these operators or prefer problem-specific mutation operators as more practical.

- Calculate the total number of genes in the population = 24 genes.
- Calculate number of mutable genes.

Number of solutions that undergo mutations in a population is decided by mutation rate p_m . Let, $\rho_m = 10\%$; Number of mutations = $0.1 \times 24 = 2.4 = 2$.

- Calculate gene positions.

Generate two random numbers below 24, say 12 and 18. Mutable genes and chromosomes are Chromosome [3]-gene 4 and Chromosome [5]-gene 2. This process is seen in **Table 6**.

- The value of mutable genes at the mutation point is substituted with random number, satisfying the boundary constraint of decision variables/genes. That is, between 0 and 30; Say 02, 05.

Stage 6: Survival selection and replacement mechanism

After mutation operation one iteration/generation of EA is over. Functional evaluation is again performed on the offspring for survival selection. From the functional evaluation of population after mutation, it is evident that the objective function value

Population after selection			Population after crossover				Population after mutation				
02	21	18	03	02	05	17	01	02	05	17	01
10	04	13	14	10	04	13	14	10	04	13	14
12	05	23	08	12	05	23	08	12	05	23	02
20	05	17	01	20	04	13	14	20	04	13	14
10	04	13	14	10	04	18	03	10	05	18	03
20	01	10	06	20	01	10	06	20	01	10	06

Table 6.
 Population after and before mutation operation.

of the best solution is reducing—37 in comparison with the minimum objective value —47 of initial random population, as shown in the table. Hence the minimization objective, $f(k) = [(x + 2y + 3z + 4u) - 30]$ is met. This means that the solutions obtained by EA at the end of the first iteration is better than the solutions of random population.

To execute the iteration process continuously, population is to be revised at the end of each iteration, as a final process, which is referred to as replacement mechanism. In each iteration end, 80–90% of best solutions from offspring population (4–5 best children) and 20–10% best solutions from the initial population (2–1 random solution) are selected to form new population for next generation [15]. Chromosomes of the next generation will then become as shown in **Tables 7 and 8**.

Population after mutation				Feval	Remarks
02	05	17	01	37	Best Solution and survive in the next generation
10	04	13	14	77	Survive in next generation
12	05	23	02	47	Survive in next generation
20	04	13	14	93	Rejected solution
10	05	18	03	56	Survive in next generation
20	01	10	06	46	Survive in next generation

Table 7.
 Survival selection.

Population after mutation				Next generation initial population				Feval	Remarks
02	05	17	01	02	05	17	01	37	
10	04	13	14	10	04	13	14	77	
12	05	23	02	12	05	23	02	47	
20	04	13	14	20	01	10	06	47	Replaced solution
10	05	18	03	10	05	18	03	56	
20	01	10	06	20	01	10	06	46	

Table 8.
 Replacement-population for the next iteration.

Stage 7: Stopping the iteration

The optimization process is repeated until when objective function value or decision variables values become stagnant, that is, have no/very little change for a greater number of iterations. Thus, over period, the solution will get converge to the final best minimum optimal solution and the optimization process will be stopped, based on any stopping criteria such as the maximum number of iterations, or maximum number of functional evaluations, etc.

6. Conclusion


The basic processes that occur behind an evolutionary algorithm have been explained and illustrated in this chapter with steps covering solution representation, population generation, functional evaluation, parent selection, genetic operations, offspring evaluations, survival selection, and stopping criteria for a simple optimization problem. This knowledge can be extended very well by researchers across any discipline, working in the field of optimization and for applying evolutionary algorithms to solve any complex engineering problem using computers. Although the process behind EA may appear to be simple, the details of the optimization process form the base and are very much necessary in applying the learned concepts for modifying the existing evolutionary concepts and evolving into better optimization methods in the research level.

Author details

S. Tamilselvi
Sri Sivasubramaniya Nadar College of Engineering, Chennai, India

*Address all correspondence to: tamilselvis@ssn.edu.in

IntechOpen

© 2022 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Michalewicz Z. *Genetic Algorithms + Data Structures = Evolution Programs*. 3rd ed. Berlin, Heidelberg: Springer; 1996. p. 387. DOI: 10.1007/978-3-662-03315-9
- [2] Gen M, Cheng R. *GA & Engineering Design*. Hoboken, New Jersey, United States: John Wiley & Sons, Inc.; 1997
- [3] Tamilselvi S, Baskar S, Anandapadmanaban L, Kadhar K, Varshini PR. Chaos-assisted multiobjective evolutionary algorithm to the design of transformer. *Soft Computing*. 2017;**21**(19):5675-5692. DOI: 10.1007/s00500-016-2145-7
- [4] Tamilselvi S, Baskar S, Anandapadmanaban L, Karthikeyan V, Rajasekar S. Multi objective evolutionary algorithm for designing energy efficient distribution transformers. *Swarm and Evolutionary Computation*. 2018;**1**(42): 109-124
- [5] Tamilselvi S, Baskar S, Sivakumar T, Anandapadmanaban L. Evolutionary algorithm-based design optimization for right choice of transformer conductor material and stepped core. *Electrical Engineering*. 2019;**101**(1):259-277
- [6] Agatz N, Bouman P, Schmidt M. Optimization approaches for the traveling salesman problem with drone. *Transportation Science*. 2018;**4**(52): 965-981
- [7] Lakshminarasimman N, Baskar S, Alphones A, Willjuice Iruthayarajan M. Evolutionary multiobjective optimization of cellular base station locations using modified NSGA-II. *Wireless Networks*. 2011;**17**(3):597-609
- [8] Tamil Selvi S, Baskar S, Rajasekar S. An Intelligent Approach Based on Metaheuristic for Generator Maintenance Scheduling—Classical and Recent Aspects of Power System Optimization. 1st ed. Cambridge, Massachusetts, United States: Academic Press; 2018. pp. 99-136
- [9] Gunasundari S, Janakiraman S, Meenambal S. Multiswarm heterogeneous binary PSO using Win-Win approach for improved feature selection in liver and kidney disease diagnosis. *Computerized Medical Imaging and Graphics*. 2018;**70**:135-154
- [10] Bhattacharya A, Chattopadhyay PK. Solving complex economic load dispatch problems using biogeography-based optimization. *Expert Systems with Applications*. 2010;**37**(5):3605-3615
- [11] De Jong KA. *Evolutionary Computation—A Unified Approach*. 1st ed. Berlin/Heidelberg, Germany: Springer; 2017. p. 268
- [12] Hermawanto D. Genetic Algorithm for Solving Simple Mathematical Equality Problem. 2013 [arXiv preprint arXiv:1308.4675]
- [13] Available from: https://www.tutoriaispoin.com/genetic_algorithms/genetic_algorithms_parent_selection.htm
- [14] Goldberg DE. *Genetic Algorithms in Search, Optimization, and Machine Learning*. 13th ed. Boston, Massachusetts, United States: Addison Wesley; 1989. p. 432
- [15] Hancock PJ. An empirical comparison of selection methods in evolutionary algorithms. In: *AISB Workshop on Evolutionary Computing 1994 Apr 11*. Berlin, Heidelberg: Springer; 1994. pp. 80-94