International Journal of Scientific Research & Engineering Trends
Volume 10, Issue 5, Sept-Oct-2024, ISSN (Online): 2395-566X

Review of Optimization Algorithms

Vivek Syal [1], Raman kumar sofat[2]

Department of Electrical Engineering
1st Swami Devi Dayal Institute of Engineering & Technology, Barwala , Panchkula (Haryana)
2nd *Department of Electrical Engineering,Lamrin Tech Skills University Punjab*

Abstract- Over the years, several optimization techniques has been developed for the real life applications. The traditional methods do not solve the nonlinear objectives. This paper presents the review of four popular optimization techniques: genetic algorithm (GA), differential evolution (DE. They can be applied to the linear, non-linear, differential and non-differential problems. The related description for each procedure of optimization is presented.

Index Terms- Evolutionary algorithms Optimization Algorithms; Metaheuristic Optimization Methods

I. INTRODUCTION

Optimization is the method to make decisions that help to achieve the goals set in the given problem. Use of optimization methods have become topic of research in all the aspects of life from engineering, markets to our daily activities. We always try to enlarge or reduce something that is called an objective. To find the optimized results for the objective, several optimization methods can be applied. For every problem, we try to find the best and more feasible solution which is not always possible.

As the complexity of problem rises, it becomes difficult to find the solution. We aim to find the best feasible solution in minimum time. Several methods of optimization had been introduced like Quadratic programing, linear programming, dynamic programing, simplex methods and many more.

This paper provides the introduction to algorithm for population based metaheuristic optimization methods: Genetic Algorithm and Differential Evolution algorithm. In fact, a number of new algorithms in optimization are metaheuristics.

II. OPTIMIZATION ALGORITHMS

Optimization is commonly referred to as mathematical programming that consists of a group of mathematical ideas and strategies for resolving quantitative issues in a variety of fields, such as market, engineering and economics. It developed a result of the discovery that quantitative issues in ostensibly unrelated fields share significant mathematical components. Because of this similarity, optimization unified principles and techniques can be used to construct and solve a wide variety of problems. Making choices that contribute to achieving the objectives outlined in the specified problem is called Optimization. The attempt is always made in order to increase or decrease what is referred to as an objective function. Multiple Optimization techniques can be used to find the results that are most suited to the purpose [23]. The achievement of best and most practical solution is desired for every issue, but this is not always attainable.

Finding a solution gets more challenging as a problem complexity increases. The aim is to find the best optimal solution in the shortest amount of time. Numerous techniques for optimization have been developed over the years to solve the single and multiple objective problems.

1. Genetic Algorithm (GA)

Genetic programming is a domain-independent problem-solving approach in which computer programs are evolved to solve, or approximately solve, problems. Genetic programming is based on the Darwinian principle of reproduction and survival of the fittest and analogs of naturally occurring genetic operations such as *crossover* and mutation]. John Holland's pioneering *Adaptation in Natural and Artificial Systems* (1975) described how an analog of the evolutionary process can be applied to solving mathematical problems and engineering optimization problems using what is now called the *genetic algorithm* (GA). The genetic algorithm attempts to find a good (or best) solution to the problem by genetically breeding a population of individuals over a series of generations. In the genetic algorithm, each *individual* in the population represents a candidate solution to the given problem. The *genetic algorithm* (GA) transforms a *population* (set) of individuals, each with an associated *fitness* value, into a new *generation* of the population using reproduction, crossover, and mutation [14].

Crossover is the primary genetic operator, which promotes the exploration of new regions in the search space. *Mutation* is a secondary operator and prevents the premature stopping of the algorithm in a local solution. The mutation operator is defined by a random bit value change in a chosen string with a low probability of such change. The mutation adds a random

© 2024 IJSRET

search character to the genetic algorithm, and it is necessary to avoid that, after some generations. Reproduction is based on the principle of survival of the better fitness. It is an operator that obtains a fixed number of copies of solutions according to their fitness value. If the score increases, then the number of copies increases too. A score value is of associated to a given solution according to its distance of the optimal solution (closer distances to the optimal solution mean higher scores) [16- 22].

A simple Genetic Algorithm is an iterative procedure, which maintains a constant size population P of candidate solutions. During each iteration step (generation) three genetic operators (reproduction, crossover, and mutation) are performing to generate new populations (offspring), and the chromosomes of the new populations are evaluated via the value of the fitness which is related to cost function. Based on these genetic operators and the evaluations, the better new populations of candidate solution are formed [6-9]. Parameter encoding is the first step in applying a genetic algorithm to a problem.

Coding

The coding of optimization parameters must be properly planned to take use of the capabilities of genetic algorithms to effectively exchange details between chromosomal strings and optimization issue objective functions. The decision variables are represented as binary coded substrings of 1s and 0s as:

$$y_j = \sum_{i=1}^{l} 2^{i-1}\, b_{ij} \quad ; (j = 1, 2, \ldots L) \qquad (1)$$

Where, y_j is the decimal-coded value of the j^{th} binary substring; b_{ij} is the i^{th} bit of the j^{th} substring; l represent stretch of substring; L represent length of string.

In the search space, the continuous variable x_j is obtained to show a point according to the fixed mapping rule, given as:

$$x_j = x_j^{min} + \frac{x_j^{max} - x_j^{min}}{2^l - 1} y_j \qquad (j=1,2,\ldots N_V) \qquad (2)$$

Where, x^{min} and x^{max} are the minimal and maximal values of variable x_j. N_V represents no. of decision variables.

The number of binary digits required to indicate a continuous fluctuation in Δx_j accuracy may be calculated using the formula below:

$$2^l \geq \frac{x^{max} - x^{min}}{\Delta x} + 1 \qquad (2.3)$$

or

$$l \geq log_2 \left(\frac{x^{max} - x^{min}}{\Delta x} + 1 \right) \qquad (2.4)$$

Fitness Function

To keep a search process going, Genetic Algorithms use the survival of the fittest concept. As a result, genetic algorithms are well-suited processes for solving maximizing issues. As a result, maximizing issues are often transformed into minimization problems using appropriate objective function transformations. The objective function is used to formulate the fitness function, which is then applied to additional genomic operations. Similarly, the fitness function may be utilized for maximizing issues with the same objective function as the fitness function.:

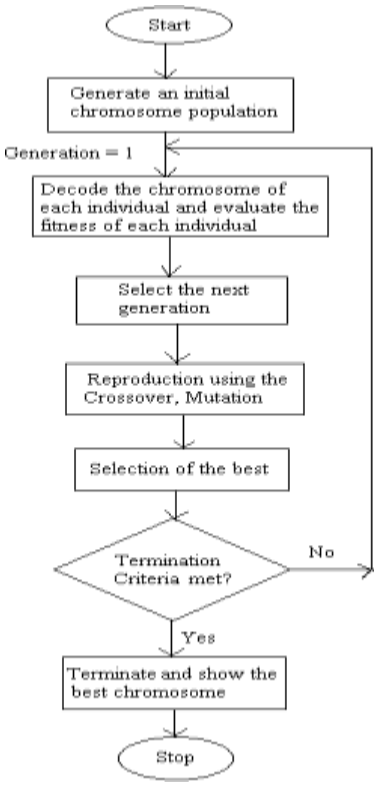
$$f(x) = F(x) \qquad (3)$$

The answer to the minimization issue may be found using the fitness function equation presented below.:

$$f(x) = \frac{1}{1+F(x)} \qquad (4)$$

The search for and location of minimal fitness value has not changed as a result of this modification. However, the conversion is only done in the case of a minimization issue with an equal maximization problem. The string's fitness value is known as string's fitness.

A simple genetic algorithm is given as follow:



Genetic Algorithm Flowchart

- Generate randomly a population of binary strings (chromosomes).
- Calculate the fitness for each string in the population
- Create offspring strings through reproduction, crossover and mutation operation.
- Evaluate the new strings and calculate the fitness for each string (chromosome).
- If the search goal is achieved, or an allowable generation is attained, return the best chromosome as the solution;
- Otherwise go to step 3.

Genetic Algorithm is a multi-objective optimization method that gives an approximate solution to the problems. It is effective method of optimization applicable in various fields. GA searches various solutions in search space from the random population and does not stuck in local minimum.

2. Differential Evolution (DE)

Differential Evolution (Storn and Price, 1997) is an iterative optimization method that tries to improve the results within the given quality measure. It is a metaheuristic approach that optimizes by searching a large space of solutions but it does not guarantee the optima l results are found. As in classical search methods a differentiable problems are required, DE uses the multi-dimensional real value function. So it can be used for the optimization problems that are not a continuous function or change with time. This method optimizes a function by creating the new solutions from the existing solution combinations and stores the best solution according to the best fitness value from the solution set based on the problem. So there is no need of gradient function [1],[11-13].

In basic DE algorithm, it starts by creating a population of random candidates in solution. The candidates then search around the search space to combine with existing solution candidates. If this combination presents the improved solution then this will be accepted and become a part of candidate solutions in population else the combination can be denied. On repetition of the combination process, it is thought that an optimal and satisfactory solution would be achieved [15].

Differential evolution generates new candidates in solution by adding the weighted difference between two population candidates to a third candidate vector, known as mutation. The mutated candidate vectors are then mixed with the candidates of other predetermined candidate vector, the target vector, to get the trial vector. This combination is known as “crossover". If the value of trial vector results a lower function value than the target vector, the trial vector is then replaced by the target vector in the upcoming generation, this is called selection. This process of mutation, crossover and selection continues for a given number of generations or till the given termination condition is satisfies.

Initialization

The evolutionary optimization methods are related to the differential evolution procedure. DE has a thriving population P^J of size Npop consisting of potential technique at every generation J.

$$P^J = [Y^J_1, \ldots Y^J_k, \ldots, Y^J_{Npop}] \quad (5)$$

Each individual solution Y^J, is a M dimensional vector that consists of as many parameters as decision variables, M, in a problem to be optimized.

$$Y^J_i = [Y^J_{1,k}, \ldots, Y^J_{i,k}, \ldots, Y^J_{M,k}]; \quad (6)$$

i= 1,2,, M ; k= 1,2, ..., Npop

Where, J is number of iterations or iterations.

To produce a population of viable solutions, it starts by assigning a random value for every deciding variable of every member of a population. These values should be inside the decision variable's limits.

$$Y^{J=0}_{i,k} = Y^{mn}_i + \lambda_i (Y^{mx}_i - Y^{min}_i); \quad (7)$$

i= 1,2,…,M; k= 1,2,.., Npop

Where, Y^{mn}_i and Y^{mx}_i are the i^{th} decision variable's lower and upper limits respectively; λ_i -for each value, a uniformly distributed random number is produced. of i; $Y^{J=0}_i$ is the initial value of the ith parameter of the kth individual vector. Once the population's vectors have been initialized, the associated fitness value is computed and saved for future reference.

Mutation

Following the initialization of the population, the three operators mutation, crossover, and selection use the current population, P^J, to construct the population of the following generation, P^{J+1}. Each vector in the population must play the role of the target vector at every iteration. Y^J_i For each target vector, a mutant vector is created simply altering one of the vectors with the comparison of the two other vectors.

Mutant Vector, $U^J_k = [U^J_{1,k}, \ldots, U^J_{D,k}]$ (8)

$$U^J_k = [Y^J_A + F(Y^J_B - Y^J_C)] \quad (9)$$

Where, Y^J_A, Y^J_B and Y^J_C are randomly selected vectors from the set { 1,2, . . ., Npop} and subjected to the condition a ≠ b ≠ c≠ k. For each parent vector, a new vectors Y^J_A, Y^J_B and Y^J_C are

© 2024 IJSRET

selected. F is a scaling mutation factor that can be chosen from a range of [0, 2].

Crossover

The crossover procedure generates trial vectors (V_i^J) by mixing the parameters of mutant vectors (U_i^J) and target vectors (Y_i^J) based on the probability distribution chosen.

$$V_{i,k}^J = \begin{cases} V_{i,k}^J & if\ \lambda_i \le C\ or\ i = s \\ Y_{i,k}^J & ot\ erwise \end{cases} \quad (10)$$

Where λ_i is a uniformly distributed random number in the range [0, 1] that is generated for each new value of i; C is a user-defined constant for crossover rate in the range [0, 1]; and s is a randomly chosen parameter from the interval [1,2,..., M] to ascertain that the mutant vector sends at least one parameter to the trial vector.

Selection

The selection mechanism selects the vectors which will make up the population of the following generation in the final step. The trail vector's fitness and the matching target vector's fitness are compared, and the vector that provides the best answer is chosen. The best fitness vector is then allowed to go to another stage

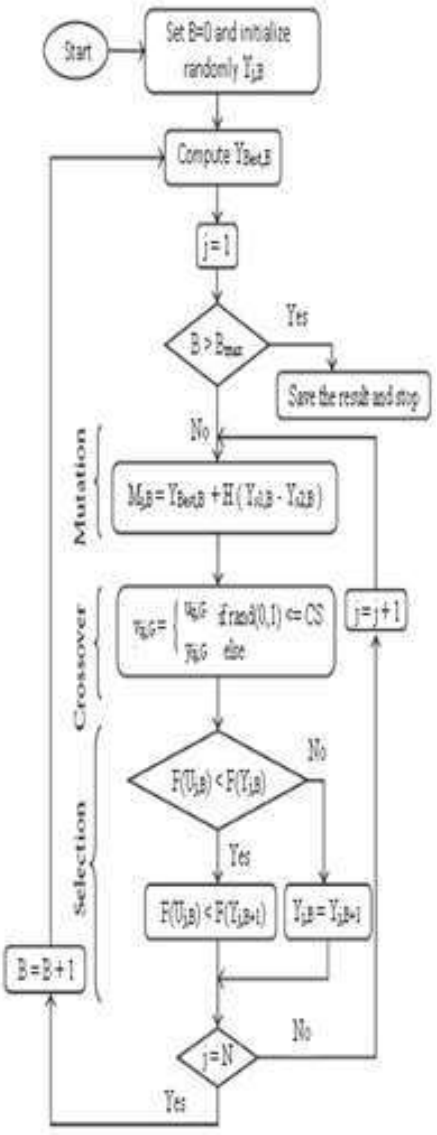
$$Y_{i,k}^{J+1} = \begin{cases} V_{i,k}^J & if\ F(V_{i,k}^J) < F(Y_{i,k}^J) \\ Y_{i,k}^J & ot\ erwise \end{cases} \quad (11)$$

This procedure is repeated for many iterations, improving the fitness of individuals in search of optimal value of solution. The procedure of differential evolution algorithm terminated when iterations reach the maximum no. or till the given termination condition is satisfied.

An algorithm for differential evolution method is given as follow:

- Generate randomly an initial population vector.
- Calculate the fitness of each candidate in the population vector.
- Run iterations, t = 1 to n
- Make t-th vector in candidate population as target
- Generate a trial vector
- Choose the best by comparing the target and trial vector.
- If the search goal is achieved, repeat from step 2; otherwise go to step 8.
- Give the best candidate solution vector in population.

Differential Evolution method is an optimization method that includes the elite, that means previous values of the search space are saved during the procedure and are not lost until a better solution is reached.



Differential Evolution Algorithm Flowchart

V. CONCLUSION

In present paper an attempt has been made to review the methods used to solve optimization problems. The methods include: Genetic Algorithm (GA) and Differential Evolution (DE). A details procedure and description of each method with its basic algorithm is presented for the better understanding. Modern optimization techniques are applicable to any kind of problem that exists in day-to-day that may be linear, non-linear, differential or multi-dimensional.

REFERENCES

1. V. Kachitvichynaukul, "Comparison of three Evolutionary Algorithms: GA, PSO and DE," Industrial

Engineering & Management Systems, Vol. 11, No 3, pp.215-223, September 2012.
2. R. Klempka, B. Filipowicz, "Comparison of using genetic algorithm and cuckoo search for multicriteria optimization with limitation," Turk J Elec Eng & Comp Sci,pp.1300-1310, vol. 25, April 2017
3. A. Dutta, S. Das, B. Tudu, K.K. Mandal, "A Novel Improved Algorithm using Cuckoo Search for Economic Load Dispatch", 1st IEEE International Conference on Power Electronics, Intelligent Control and Energy Systems, 2016.
4. Y. Xiaodong, C. Zefan, "Particle Swarm Optimization and Cuckoo Search Paralleled Algorithm," 3rd IEEE International Conference on Computer and Communication, 2017.
5. C.L. Chiang, "Power Economic/ Environmental Dispatch Problem Using Cuckoo Search Algorithm", 4th International Conference on Systems and informatics, 2017..
6. M. Sasson, "Non linear Programming Solutions for Load Flow, Minimum Loss, and Economic Dispatching Problems", IEEE Trans. on Power Apparatus and Systems, vol. 4, Apr. 1969.
7. T. Bouktir, M. Belkacemi, K. Zehar, "Optimal power flow using modified gradient method," Proceeding ICEL'2000, U.S.T.Oran, Algeria, vol. 2, pp. 436-442, 13-15 Nov. 2000.
8. R. Fletcher, "Practical Methods of Optimization," John Willey & Sons, 1986.
9. D.E.Goldberg, "Genetic Algorithms in Search, Optimization and Machine Learning," Addison Wesley Publishing Company, Ind. USA, 1989.
10. C. Zefan, Y. Xiaodong , "Cuckoo Search Algorithm with Deep Search," 3rd IEEE International Conference on Computer and Communication, 2017.
11. D. Niu, F. Wang, Y. Chang, D. He, D.Gu, "An Improved Multi-objective Differential Evolution Algorithm," 24th Chinese Control and Decision Conference, 978-1-4577-2074-1/12/$26.00,2012 IEEE.
12. H. Zhao, J. Xiao, "A New Many-Objective Evolutionary Algorithm Based on Self-Adaptive Differential Evolution, 9th International Conference on Natural Computation, 978-1-4673-4714-3/13/$31.00, 2013 IEEE.
13. R. Lei, Y Cheng, "A Pareto-Based Differential Evolution Algorithm for Multi-objective Optimization Problems," Chinese Control and Decision Conference, 978-1-4244-5182-1/10/$26.00, 2010 IEEE.
14. S. Katoch, S. S.Chauhan, V. Kumar, "A review on genetic algorithm: past, present and future," Multimedia Tools and Applications, Springer Science+Business Media, LLC, part of Springer Nature, October 2020.
15. S.A. Pandya, V.V. Mehta, N.K. Meghapara,"A Review of Moderm OptimizationTechniques," Journal of Emerging Technologies and Innovative Research, Vol. 4, Issue 07,July 2017.
16. D. E Goldberg, "Sizing Populations for Serial and Parallel Genetic Algorithms", in J.D. Schaffer, editor, Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann, 1989.
17. John R. Koza, "Genetic Programming" for Encyclopedia of Computer Science and Technology," edited by Allen Kent and G. Williams James, vol 2, Submitted August 18, 1997.
18. Galesia, "Genetic Algorithms in Engineering Systems," Proceedings of the First International Conference, Innovations and Applications (London: Institution of Electrical Engineers. IEE. 1995.
19. Kinnear, E. Kenneth Jr. (editor), "Advances in Genetic Programming," Cambridge, MA:MIT Press,1994.
20. Koza, R. John and Andre David, "Evolution of Iteration in Genetic Programming in Evolutionary Programming", V: Proceedings of the Fifth Annual Conference on Evolutionary Programming. Cambridge, MA: MIT Press, 1996.
21. Y. J. CAO and Q. H. WU, "Teaching Genetic Algorithm using MATLAB", Intelligent Computer Systems Centre, University of the West of England, Bristol, UK Manchester U.P. Printed in Great Britain, 1999.
22. G. Bakirtzis Anastasios, Senior Member, I, Pandel N. Biskas, Student Member,IEEE, Christoforos E. Zoumas, Student Member, IEEE, and Vasilios Petridis, Member, "Optimal Power Flow by Enhanced Genetic Algorithm" IEEE transactions on power systems, vol. 17, May 2002.
23. [7] R. Fletcher, "Practical Methods of Optimization," John Willey & Sons, 1986.

© 2024 IJSRET