Int. J. Adv. Res. Sci. Technol. Volume 2, Issue 2, 2013,pp. 106-109.

IJARST

International Journal of Advanced Research in Science and Technology

journal homepage: www.ijarst.com

ISSN 2319 – 1783 (Print)

ISSN 2320 – 1126 (Online)

Review Of Revolutionary algorithms as an Optimization tool for test cases

Sujit Kumar panda, Smruti Ranjan Swain and Aradhana Sahoo*

Gandhi institute for Technology, Bhubaneswar, Odisha, India

****Corresponding Author's Email: sujit.panda@gift.edu.in***

ARTICLEINFO

Article history:

Received 07 Sep. 2013

Accepted 22 Sep. 2013

Available online 30 Sep. 2013

Keywords:

Revolutionary algorithm,
Optimization tool,
Test cases.

ABSTRACT

The aim of this paper is , we focus on one such non-traditional optimization method which takes the ion's share of all non-traditional optimization methods. This so-called 'evolutionary algorithm (EA)' mimics the natural evolutionary principles on randomly-picked solutions from the search space of the problem and iteratively progresses towards the optimum point. Nature's ruthless selective advantage to interest individuals and creation of new and fit individuals using re-combinative and mutative genetic processing with generations is well- mimicked artificially in a computer algorithm to be played on a search space where good and bad solutions to the underlying problem coexist. The task of an evolutionary optimization algorithm is then to avoid the bad solutions in the search space, take clues from good solutions and eventually reach close to the best solution, similar to the genetic processing in natural systems.

Introduction:

Optimization is an activity which does not belong to any particular discipline and is routinely used in almost all fields of science, engineering and commerce. The Chambers dictionary describes optimization as an act of 'making the most or best of anything'. Theoretically speaking, performing an optimization task in a problem means finding the most or best suitable solution of the problem. Mathematical optimization studies spend a great deal of effort in trying to describe the properties of such an ideal solution. Engineering or practical optimization studies, on the other hand, thrive to look for a solution which is as similar to such an ideal solution as possible. Although the ideal optimal solution is desired, the restrictions on computing power and time often make the practitioners happy with an approximate solution. Serious studies on practical optimization begun as early as the Second World War, when the need for efficient deployment and resource allocation of military personnel and accessories became important. Most development in the so-called 'classical' optimization field was made by developing step-by-step procedures for solving a particular type of an optimization problem. Often fundamental ideas from geometry and calculus were borrowed to reach the optimum in an iterative manner. Such optimization procedures have enjoyed a good 50 years of research and applications and are still going strong. However, around the middle of eighties, completely unorthodox and less-mathematical yet intriguing optimization procedures have been suggested mostly by computer scientists. It is not surprising because these 'non-traditional' optimization methods exploit the fast and distributed computing machines which are for finding suggestions.

An Optimization Problem and its Notations:

Throughout this article, we describe procedures for finding the optimum solution of a problem of the following type Classical Search and Optimization Techniques Most classical point-by-point algorithms use a deterministic procedure for approaching the optimum solution. Such algorithms start from a random guess solution. Thereafter, based on a pre-specified transition rule, the algorithm suggests a search direction, which is often arrived at by considering localized information. A one-dimensional search is then performed along the search direction to find the best solution. This best solution becomes the new solution and the above procedure is continued for a number of times .Figure 1 illustrates this procedure. Algorithms vary mostly in the way the search directions are defined at each intermediate solution. Classical search and optimization methods can be classified into two distinct

groups mostly in the way the directions are chosen: Direct and gradient-based methods [3, 22, 24]. In direct methods, only objective function and constraints re used to guide the search strategy, whereas gradient-based methods use the first and/or second-order derivatives of the objective function and/or constraints to guide the search process. Since derivative information is not used, the direct search methods are usually slow, requiring many function evaluations for convergence. For the same reason, they can be applied to many problems without a major change of the algorithm. On the other hand, gradient-based methods quickly converge to an optimal solution, but are not efficient in non-differentiable or discontinuous problems. In addition, there are some common difficulties with most of the traditional direct and gradient-based techniques:

1. Convergence to an optimal solution depends on the chosen initial solution.

2. Most algorithms are prone to get stuck to a suboptimal solution.

3. An algorithm efficient in solving one problem may not be efficient in solving a different problem.

4. Algorithms are not efficient in handling problems having discrete variables or highly non-linear and many constraints.

5. Algorithms cannot be efficiently used on a parallel computer.

Motivation from Nature:

It is commonly believed that the main driving principle behind the natural evolution is the Darwin's survival-of-the-fittest principle [2, 11]. In most scenarios, the nature ruthlessly follows two simple principles:

1.If by genetic processing an above-average offspring is created, it usually survives longer than an average individual and thus has more opportunities to produce offspring having some of its traits than an average in dividable.

2. If, on the other hand, a below-average offspring is created, it usually does not survive longer and thus gets eliminated soon from the population.

The principle of emphasizing good solutions and eliminating bad solutions seems to dovetail well with desired properties of a good optimization algorithm. But one may wonder about the real connection between an optimization procedure and natural evolution! Has the natural evolutionary process tried to maximize a utility function of some sort? Truly speaking, one can imagine a number of such functions which the nature may have been thriving to maximize: life span of aspecies, quality of life of a species, physical growth, and others .However, any of these functions is non-stationary in nature and largely depends on the evolution of other related species. Thus, in essence, the nature has been really optimizing a much more complicated objective function by means of natural genetics and natural selection than the search and optimization problems we are interested in solving in practice. Thus, it is not surprising that a genetic algorithm is not as complex as the natural genetics and selection procedures; rather it is an abstraction of the complex natural evolutionary process. The simple version of a GA described in the following section aims to solve stationary search and optimization problems. Although a GA is a simple abstraction, it is robust and has been found to solve various search and optimization problems of science, engineering, and commerce. Programs to manipulate. Natural language processing.

Revolutionary Algorithms:

The optimization of dynamic problems is both widespread and difficult. When conducting dynamic optimization, a balance between reinitialization and computational expense has to be found. There are multiple approaches to this. In parallel genetic algorithms, multiple sub-populations concurrently try to optimize a potentially dynamic problem. But as the number of sub-population increases, their efficiency decreases. Cultural algorithms provide a framework that has the potential to make optimizations more efficient. But they adapt slowly to changing environments. We thus suggest a confluence of these approaches: revolutionary algorithms. These algorithms seek to extend the evolutionary and cultural aspects of the former to approaches with a notion of the political. By modeling how belief systems are changed by means of revolution, these algorithms provide a framework to model and optimize dynamic problems in an efficient fashion.

Optimization is an activity which does not belong to any particular discipline and is routinely used in almost all fields of science, engineering and commerce. The Chambers dictionary describes optimization as an act of 'making the most or best of anything'. Theoretically speaking, performing an optimization task in a problem means finding the most or best suitable solution of the problem. Mathematical optimization studies spend a great deal of effort in trying to describe the properties of such an ideal solution. Engineering or practical optimization studies, on the other hand, thrive to look for a solution which is as similar to such an ideal solution as possible. Although the ideal optimal solution is desired, the restrictions on computing power and time often make the practitioners happy with an approximate solution. Serious studies on practical optimization begun as early as the Second World War, when the need for efficient deployment and resource allocation of military personnel and accessories became important.

Classical Search and Optimization Techniques Most classical point-by-point algorithms use a deterministic procedure for approaching the optimum solution. Such algorithms start from a random guess solution. Thereafter, based on a pre-specified transition rule, the algorithm suggests a search direction, which is often arrived at by considering localized information. A one-dimensional search is then performed along the search direction to find the best solution. This best solution becomes the new solution and the above procedure is continued for a number of times.

Classical search and optimization methods can be classified into two distinct groups mostly in the way the directions are chosen: Direct and gradient-based methods [3, 22, 24]. In direct methods, only objective function and constraints are used to guide the search strategy, whereas gradient-based methods use the first and/or second-order derivatives of the objective function and/or constraints to guide the search process. Since derivative information is not used. In the absence of any knowledge of the problem domain, a GA begins its search from a random population of solutions. If a termination criterion is not satisfied, three different operators-reproduction, crossover, and mutation – are applied to update the population of solutions. One iteration of these three operators is known as a generation in the parlance of GAs. Since the representation of a solution in a GA is similar to a natural chromosome and GA operators are similar to genetic operators, the above procedure is called a genetic algorithm.

Fitness Assignment:

In a GA, each string created either in the initial population or in the subsequent generations must be assigned a fitness value which is related to the objective function value of the string. The fitness should reflect the goodness of the solution. For minimization problems, a smaller fitness value is judged good.

Crossover Operator:

Crossover operator is applied next to the strings of the mating pool. In a typical crossover operator, two strings are picked from the mating pool at random and some portions of the strings are exchanged between the strings.It is important to note that the above crossover operator created a solution (having a cost of

units) which is better in cost than both of the parent solutions. One may wonder that if a different cross site was chosen or two other strings were chosen for crossover, whether we would have found a better offspring every time.

Reproduction Operator:

Reproduction (or selection) is usually the first operator applied on a population. Reproduction selects good strings in a population and forms a mating pool. These strings have survived tournaments played with other 12solutions during the earlier reproduction phase. Thus, they are expected to have some good bit combinations in their string representations. Since, a single-point crossover on a pair of parent trings can only create it.

Constraint Handling in Gas:

Constraints are inevitable in real-world problems. The classical penalty approach of penalizing an infeasible solution is sensitive to a penalty parameter associated with the technique and often requires a trial-and-error method. In the recent past, a parameter-less penalty approach is suggested by the author [4] with the following fitness function derived from the objective. In a tournament between two feasible solutions, the first clause ensures that the one with a better function value wins. The quantity is the objective function value of the worst feasible solution in the population. The addition of these quantities to the constraint violation ensures that a feasible solution is always better than any infeasible solution. Moreover, since this quantity is constant in any generation, between two infeasible solutions the one with smaller constraint violation is judged better. Since the objective function value is never compared with a constraint violation amount, there is no need of any penalty parameter with such an approach.

A Case Study: Car Suspension Design Using:

Genetic Algorithms:

Suspension systems are primarily used in a car to isolate road excitations from being transmitted directly to the passengers Indian automobile company. The car motion is simulated over a sinusoidal bump having 500 mm width and 70 mm height. Constraints on the pitching motion and change in acceleration were also recommended by the automobile company. Although there are only four variables in the problem, the constraints are highly nonlinear and produce enough difficulties for a classical optimization method to find the true optimum solution. The following table shows that a Matlab code implementing the sequential quadratic programming (SQP) approach to the same problem is sensitive to the initial guess solution and the obtained results are far from being optimum minimum GA-optimized.

Customized Genetic Algorithms:

In trying to solve a real-world optimization problem, it is often necessary to introduce problem information to genetic operators. In a casting scheduling problem introduced to the author by a foundry from Lucknow.

Multi-Objective Genetic Algorithms:

Most real-world search and optimization problems involve multiple conflicting objectives, of which the

user is unable to establish a relative preference. Such considerations give rise to a set of multiple solutions

Conclusion:

This paper focus on one such non-traditional optimization method which takes the ion's share of all non-traditional optimization methods. This so-called 'evolutionary algorithm (EA)' mimics the natural evolutionary principles on randomly-picked solutions from the search space of the problem and iteratively progresses towards the optimum point.

References:

1. H.-G. Beyer. The theory of evolution strategies. Berlin, Germany: Springer, 2001.
2. R. Dawkins. The Selfish Gene. New York: Oxford University Press, 1976.
3. K. Deb. Optimization for Engineering Design: Algorithms and Examples. New Delhi: Prentice-Hall, 1995.
4. K. Deb. An efficient constraint handling method for genetic algorithms. Computer Methods in Applied Mechanics and Engineering, 186(2-4):311-338, 2000.
5. K. Deb. Multi-objective optimization using evolutionary algorithms. Chichester, UK: Wiley, 2001.
6. K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. Evolutionary Computation Journal, 10(4):371-395, 2002.
7. K. Deb and D. E. Goldberg. An investigation of niche and species formation in genetic function optimization in Proceedings of the Third International Conference on Genetic Algorithms, pages 42-50, 1989.
8. K. Deb and S. Jain. Multi-speed gearbox design using multi-objective evolutionary algorithms. ASME Transactions on Mechanical Design, 125(3):609-619, 2003.
9. K. Deb, A. R. Reddy, and G. Singh. Optimal scheduling of casting sequence using genetic algorithms. Journal of Materials and Manufacturing Processes, 18(3):409-432, 2003.
10. K. Deb and V. Saxena. Car suspension design for comfort using genetic algorithms. In Proceedings of the Seventh International Conference on Genetic Algorithms, pages 553-560, 1997.
11. N. Eldredge. Macro-Evolutionary Dynamics: Species, Niches, and Adaptive Peaks. New York: McGraw-Hill, 1989.
12. D. B. Fogel. Evolutionary Computation. Piscataway, NY: IEEE Press, 1995.
13. L. J. Fogel. Autonomous automata. Industrial Research, 4(1):14-19, 1962.
14. D. E. Goldberg. Genetic Algorithms for Search, Optimization, and Machine Learning. Reading, MA: Addison-Wesley, 1989.
15. N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In Proceedings of the IEEE International Conference on Evolutionary Computation, pages 312-317, 1996.
16. 16 .J. H. Holland. Adaptation in Natural and Artificial Systems. Ann Arbor, MI: MIT Press, 1975.
17. P. Jain and A. M. Agogino. Theory of design: An optimization perspective. Mech. Mach. Theory, 25(3):287-303, 1990.
18. James Kennedy and Russell C. Eberhart. Swarm intelligence. Morgan Kaufmann, 2001.
19. J. R. Koza. Genetic Programming : On the Programming of Computers by Means of Natural Selection. Cambridge, MA: MIT Press, 1992.
20. K. Miettinen. Nonlinear Multi-objective Optimization. Kluwer, Boston, 1999.
21. A. PrÄugel-Bennett and J. L. Shapiro. An analysis of genetic algorithms using statistical mechanics. PhysicsReview Letters, 72(9):1305-1309, 1994.
22. S. S. Rao. Optimization: Theory and Applications. Wiley, New York, 1984.
23. I. Rechenberg. Cybernetic solution path of an experimental problem, 1965. Royal Aircraft Establishment, Library Translation Number 1122, Farnborough, UK.
24. G. V. Reklaitis, A. Ravindran, and K. M. Ragsdell. Engineering Optimization Methods and Applications. New York: Wiley, 1983.
25. A. Rogers and A. PrÄugel-Bennett. Modelling the dynamics of steady-state genetic algorithms. In Foundations of Genetic Algorithms 5 (FOGA-5), pages 57-68, 1998.
26. G. Rudolph. Convergence analysis of canonical genetic algorithms. IEEE Transactions on Neural Network, 5(1):96-101, 1994.
27. H.-P Schwefel. Projekt MHD-Staustrahlrohr: Experimentelle optimierung einerzweiphasendÄuse, teil I. Technical Report 11.034/68, 35, AEG Forschungsinstitut, Berlin, 1968.
28. H.-P. Schwefel. Numerical Optimization of Computer Models. Chichester, UK: Wiley, 1981.
29. R. Storn and K. Price. Deferential evolution {a fast and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization, 11:341-359, 1997.
30. H. A. Taha. Operations Research. New York: Macmillan, 1989.
31. M. D. Vose. Simple Genetic Algorithm: Foundation and Theory. Ann Arbor, MI: MIT Press, 1999.
32. M. D. Vose and J. E. Rowe. Random heuristic search: Applications to gas and functions of unitation. Computer Methods.