Global Journal of researches in engineering
Numerical Methods
Volume 12 Issue 1 Version 1.0 March 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

Comparative Analysis of Threshold Acceptance Algorithm, Simulated Annealing Algorithm and Genetic Algorithm for Function Optimization

By Tejas P. Patalia & Dr. G.R. Kulkarni
Singhania University, Rajasthan, India

Abstract - The goal of this study of threshold acceptance algorithm (TA), simulated annealing algorithm (SA) and genetic algorithm (GA) is to determine strength of Genetic Algorithm over other algorithm. It gives a clear idea of how genetic algorithm works. It gives the idea of various sub methods used in genetic algorithm to improve the results and outcome. Basically genetic algorithm and all traditional heuristic methods are used for optimization. Optimization problems are class NP complete problems. Genetic algorithm can be viewed as an optimization technique which exploits random search within a defined search space to solve a problem by some intelligence ideas of nature. In this work we have done Comparative analysis of Threshold Acceptance Algorithm, Simulated Annealing Algorithm and Genetic Algorithm by considering different test functions and its constraints to minimize the test functions.

Keywords : Heuristic methods, Genetic Algorithm, Chromosomes, Mutation, threshold acceptance algorithm, simulated annealing algorithm, function optimization.

GJRE-I Classification : FOR Code: 010399

Strictly as per the compliance and regulations of:

Comparative Analysis of Threshold Acceptance Algorithm, Simulated Annealing Algorithm and Genetic Algorithm for Function Optimization

Tejas P. Patalia [α] & Dr. G.R. Kulkarni [σ]

Abstract - The goal of this study of threshold acceptance algorithm (TA), simulated annealing algorithm (SA) and genetic algorithm (GA) is to determine strength of Genetic Algorithm over other algorithm. It gives a clear idea of how genetic algorithm works. It gives the idea of various sub methods used in genetic algorithm to improve the results and outcome. Basically genetic algorithm and all traditional heuristic methods are used for optimization. Optimization problems are class NP complete problems. Genetic algorithm can be viewed as an optimization technique which exploits random search within a defined search space to solve a problem by some intelligence ideas of nature. In this work we have done Comparative analysis of Threshold Acceptance Algorithm, Simulated Annealing Algorithm and Genetic Algorithm by considering different test functions and its constraints to minimize the test functions.

Keywords : Heuristic methods, Genetic Algorithm, Chromosomes, Mutation, threshold acceptance algorithm, simulated annealing algorithm, function optimization.

I. Introduction

Optimization is the process of finding absolutely best values of the variables so that value of an objective function becomes optimal. Optimization problems are a class of NP-Complete problems. This work contains overview of threshold acceptance algorithm, simulated annealing algorithm and brief introduction to Genetic algorithm. Genetic algorithm is probabilistic, heuristic, robust search algorithm premised on the evolutionary ideas of natural selection and genetic. Main idea behind the design of genetic algorithm is to achieve robustness and adaptiveness in real world complex problems. Genetic algorithm can be viewed as an Optimization technique, which exploits random search within a defined search space to solve a problem, by some intelligence ideas of nature.

II. Threshold Acceptance Algorithm

Threshold Accepting (TA) is a local search method and was first described by Dueck and Scheuer and Moscato and Fontanari.

Author α : PhD Student, Singhania University – Rajasthan & Sr. Lecturer, V.V.P. Engineering College, Rajkot – Gujarat, India.
E-mail : pataliatejas@rediffmail.com
Author σ : Principal, C.U. Shah College of Engineering & Technology, Wadhwan City - 363030 – Gujarat, India.

A classical local search starts with a random feasible solution and then explores its neighbourhood in the solution space by moving (usually randomly) from its current position, accepting a new solution if and only if it improves the objective function. TA overcomes the problem of stopping in local minima by also allowing uphill-moves that is TA also accepts new solutions which lead to higher objective function values.

To implement TA, three points need to be specified:

1. The objective function f: This function is generally given by the problem at hand.
2. The neighborhood definition (the function N): Given a candidate solution x^c, one needs to define how to move from this solution to an alternative, but 'close' solution x^n.
3. The thresholds: Given a neigbourhood definition, one needs to determine the magnitude of the deterioration in the objective function that the algorithm should still accept for a new solution.

The pseudo-code of TA can be given as follows:

1: Initialize ηRounds and ηSteps
2: Compute threshold sequence Tr
3: Randomly generate current solution $x^c \epsilon X$
4: for r = 1: ηRounds do
5: for i = 1: ηSteps do
6: Generate $xn \epsilon N(xc)$ and compute $\Delta = f(x^n) - f(x^c)$
7: if $\Delta < Tr$ then $x^c = x^n$
8: end for
9: end for
10: $x^{sol} = x^c$

Here, *f* is the objective function to be minimized. x^c denotes the current solution, x^n is the 'new' (or neighbor) solution, and X is the set of feasible solutions.

TA starts with a (random) feasible solution. Given a threshold sequence T of length ηRounds, one can see that TA always accepts a solution that improves the objective function f, but deteriorations are only accepted if they are not worse than a particular threshold, Tr. Over time, the threshold decreases to zero, thus TA turns into a classical local search.

III. Simulated Annealing Algorithm

Simulated Annealing (SA) was introduced by Kirkpatrick. Like other trajectory methods, it evolves a single solution over time. By changing this solution

gradually, the algorithm follows some path ('trajectory') through the search space.

SA starts with a random solution x^c and creates a new solution x^n by adding a small perturbation to x^c. If the new solution is better ($\Delta < 0$), it is accepted. In case it is worse, though, SA applies a stochastic acceptance criterion, thus there is still a chance that the new solution is accepted, albeit only with a certain probability. This probability is a decreasing function of both the order of magnitude of the deterioration and the time the algorithm has already run. The latter feature is controlled by the temperature parameter T which is reduced over time; hence impairments in the objective function become less likely to be accepted and eventually SA turns into classical local search. Here, the algorithm stops after a predefined number of iterations Rmax; of course, alternative stopping criteria are possible.

The pseudo-code of SA can be given as follows:

1: Generate initial solution x^c, initialize Rmax and T
2: for r = 1 to Rmax do
3: while stopping criteria not met do
4: Compute $x^n \epsilon N(x^c)$ (neighbour to current solution)
5: Compute $\Delta = f(x^n) - f(x^c)$ and generate u (uniform random variable)
6: if ($\Delta < 0$) or ($e^{-\Delta/T} > u$) then $x^c = x^n$
7: end while
8: Reduce T
9: end for

IV. Genetic Algorithm

What is Genetic Algorithm?

Genetic algorithms are probabilistic, robust and heuristic search algorithms premised on the evolutionary ideas of natural selection and genetic.

Darwin's Principle of Natural Selection

IF there are organisms that reproduce, and

IF offspring's inherit traits from their progenitors, and

IF there is variability of traits, and

IF the environment cannot support all members of a growing population,

THEN those members of the population with less adaptive traits will die out, and

THEN those members with more-adaptive traits will thrive.

Concept

The basic concept of genetic algorithms is designed to simulate the processes in natural system necessary in for evolution, specifically for those that follow the principle of *survival of the fittest*. They represent the intelligent exploitation of a random search within a defined search space to solve a problem. Genetic Algorithm is developed by John Holland and his students at Michigan University during 1965-1975.

Encoding

Encoding is the first step towards genetic algorithm. First the data is encoded with the help of some encoding technique. Then it is given to genetic algorithm. Selection of encoding technique depends upon the problem. Different types of encoding techniques are available. They are as follows

Binary Encoding

Most common method of encoding. Chromosomes are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem

Chromosome A 110011001110110

Chromosome B 100011001100111

Permutation Encoding

Useful in ordering problems such as the Traveling Salesman problem (TSP). In TSP every chromosome is a string of numbers, each of which represents a city to be visited.

Chromosome A 1 5 3 2 6 4 7 9 8

Chromosome B 8 5 6 7 2 3 1 4 9

Value Encoding

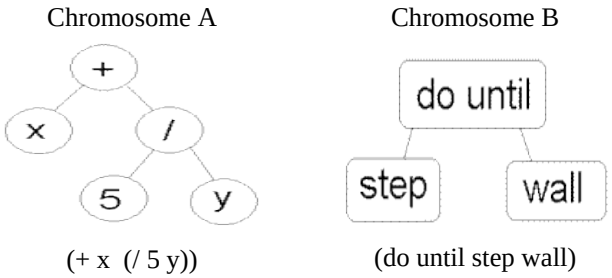
Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice. Good for some problems, but often necessary to develop some specific crossover and mutation techniques for these chromosomes.

Chromosome A 1.235 5.323 0.454 2.321 2.454

Chromosome B (left), (back), (left), (right), (forward)

Tree Encoding

Tree encoding is used mainly for evolving programs or expressions. In the tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. Tree encoding is useful for evolving programs or any other structures that can be encoded in trees. Programming language LISP is often used for this purpose, since programs in LISP are represented directly in the form of tree and can be easily parsed as a tree, so the crossover and mutation can be relatively easily.



a) Components of Genetic Algorithm

i. *Chromosomes*

All living organisms are different then other

organisms of the same species as well as different species. Even twins have at least some minor differences. These differences are due to genetic structure, which is called chromosomes. Chromosomes or individuals are consisting of genes. Genes may contain different possible values depending on the environment, constraints and struggle to survive. Each gene is responsible for some part of solution but we cannot identify the role of each gene individually because they work collectively and their inter-relations are complex. The encoding process of solution as a chromosome is most difficult aspect of solving any problem using genetic algorithm. Encoding of solution as a chromosome is known as genotype and its equivalent physical representation is known as phenotype.

ii. *Fitness Function*

In the nature the organism's "fitness" can be measured by its ability to reproduce, to adapt and to survive. In genetic algorithm chromosomes should be measured by some technique to decide which chromosomes are good compared to other chromosomes. Fitness function is a objective or evaluation function which is used to measure how good a chromosome is. Fitness function assigns fitness value to each chromosome using genetic structure and relevant information of the chromosome. Fitness function plays a big role because subsequent genetic operators use fitness values to select chromosomes. Different fitness functions are used depending on type and solution vector of problem. For function optimization problems, fitness function may be the value of objective function.

iii. *Reproduction*

Reproduction or selection is based on the concept natural selection and it is one of the main three operators used in genetic algorithm. The main objective of reproduction operator is to emphasize good chromosomes in a population. Reproduction makes multiple copies of relatively good chromosomes at the cost of relatively bad chromosomes while keeping population size constant. The essential idea is that chromosomes having a higher fitness value have a higher probability of selection. The identification of good or bad chromosomes is done using fitness value of the chromosomes. Many selection methods are available, some of them make multiple copies of the chromosomes on the basis of probability, where as some make multiple copies deterministically.

1) Roulette Wheel Selection Method

In this method each chromosome in the population occupies an area of the roulette wheel proportional to its fitness value. Chromosomes with better fitness occupies large fraction of roulette wheel where as chromosomes with bad fitness occupies small fraction of roulette wheel. Then roulette wheel is spun as many times as population size. Each time roulette wheel pointer points one chromosome and that chromosome is placed in mating pool. A chromosome with a higher fitness is likely to receive more copies than a chromosome with a lower fitness. Roulette Wheel Selection method is widely used for the maximization problems but it has two main drawbacks:

i. It can handle only maximization problem so minimization problem must be converted into an equivalent maximization problem.
ii. If a population contains a chromosome having exceptionally better fitness compared to the rest of the chromosomes in the population then this chromosome occupies most of the roulette wheel area. Thus, almost all the spinning of the roulette wheel is likely to choose the same chromosome, this may result in the loss of genes diversity and population may coverage to local optima.

2) Rank Based Selection Method

Rank based selection uses fitness value of the chromosomes to sort chromosomes in to ascending or descending order depending on the minimization or maximization problem. Then it assigns reproduction probability and ranked fitness to each chromosome on the basis of only rank order of the chromosome in the current population. Rank based selection also assigns some probability to the worst chromosome so that it has some chance for getting selected.

3) Steady State Selection

In the steady state selection in every generation a few good chromosomes are selected for creating new offspring. Then some bad chromosomes are removed and the new offspring is placed. The rest of population survives to new generation.

4) Elitism

After crossover and mutation new population is generated. With the help of elitism we can store the best found chromosomes. The remaining chromosomes are delivered for the next generation. In this way we cannot lose best chromosomes.

5) Tournament Selection

Runs a “tournament” among a few individuals chosen at random from the population and selects the winner (the one with the best fitness) for crossover. Two entities are picked out of the pool, their fitness is compared, and the better is permitted to reproduce. Advantage is decreases computing time.

b) Operators of Genetic Algorithm

i. *Crossover*

Two parents chromosomes are selected randomly from the mating pool, few genes of the chromosomes are exchanged between these two parents and offspring are produced. In general crossover operator recombines two chromosomes so it is also known as recombination. Crossover is intelligent search operator that exploits the information acquired by

the parent chromosomes to generate new offspring. If both the parent has same genetic structure then offspring are just copies of the parent irrespective of cutting point but if parent have different genetic structure then offspring are different then parent. Thus, crossover is sampling process, which samples new points in search space. Generally crossover probability is very high like 1.00, 0.95, 0.90 etc.. Different types of Crossover methods can be used i.e. 1-point crossover, n-point crossover and uniform crossover.

ii. *Mutation*

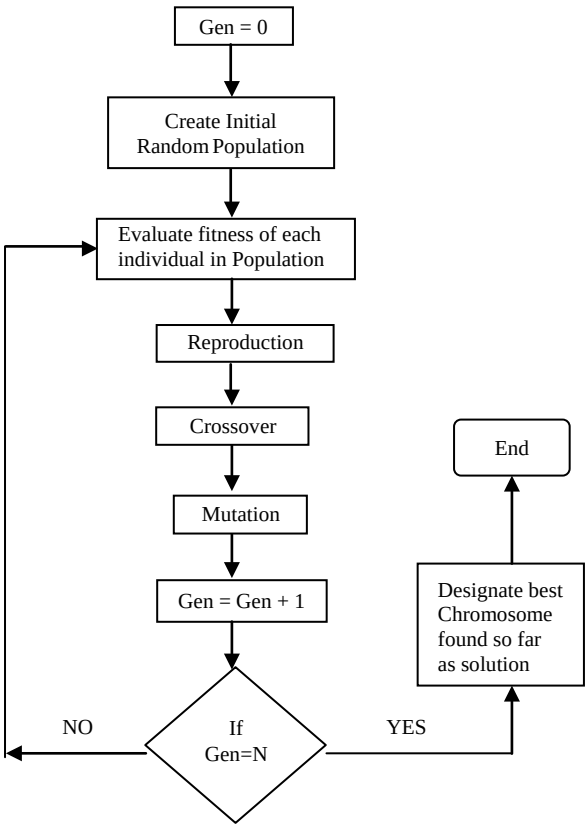
Mutation is secondary operator used in genetic algorithm to explore new points in the search space. In the latter stages of a run, the population may converge in wrong direction and stuck to local optima. The effect of mutation is to reintroduce divergence into a converging population. Mutation operator selects one chromosome randomly from the population, then selects some genes using mutation probability and flips that bit. So mutation is a random operator that randomly alters some value. Mutation either explores some new points in the search space and leads population to global optima direction or alters value of the best chromosome and losses knowledge acquired till now. So mutation should be used rarely. Generally per gene probability of mutation is 0.001, 0.01, 0.02 etc..

Flip a bit

Parent 110-011-010-001-101

Child 110-111-010-001-101

iii. *GA Flowchart*



The pseudo-code of GA can be given as follows:

1. Set the values of the parameters regarding population size, probability of crossover, probability of mutation, number of generations, and all the other parameters.
2. Generate random initial population of chromosomes.
3. Select two of the chromosomes as parents, with probability proportional to their fitness.
4. If crossover is used, combine the genes of these chromosomes using the crossover operator to form two children chromosomes. In the case no crossover is applied, the children chromosomes will be initially, just copies of the parent chromosomes.
5. Then apply the mutation operator to the children chromosomes, so that some (if any) random bits of the children chromosomes are inverted.
6. Repeat steps 4-6, until children chromosomes have been formed.
7. Repeat steps 3-7 until the specified number of generations have passed.

V. Test Functions

Different types of Nine Test Functions are considered as given below in Table 1. Threshold Acceptance Algorithm (TA), Simulated Annealing Algorithm (SA) and Genetic Algorithm (GA) are applied to these test functions.

Table I : List of 9 Test Functions

Function Name	9 Test Functions	Cons-traints	Function Value
F1	$f(x_1, x_2)=x_1 + x_2$	[0,1]	Minimize
F2	$f(x_1, x_2)=x_1^2 + x_2^2$	[0,1]	Minimize
F3	$f(x_1, x_2)=20 + x_1^2 + x_2^2$	[0,1]	Minimize
F4	$f(x_1, x_2)=x_1^4 + x_2^4 + (2 \times x_1 \times x_2)$	[0,1]	Minimize
F5	$f(x_1, x_2)=x_1^2 + x_2^2 + (20 \times x_1 \times x_2)$	[0,1]	Minimize
F6	$f(x_1, x_2, x_3, x_4)=x_1^2 + x_2^2 + x_3^2 + x_4^2$	[0,1]	Minimize
F7	$f(x_1, x_2)=(x_1 \times \sin\sqrt{x_1}) + (x_2 \times \sin\sqrt{x_2})$	[0,1]	Minimize
F8	$f(x_1, x_2)=x_1^2 - (10 \times \cos(2 \times \pi \times x_1)) + 10 + (10 \times \cos(2 \times \pi \times x_2)) + 10$	[0,1]	Minimize
F9	$f(x_1, x_2)=(4 \times x_1^2) + (2.1 \times x_1^4) + (x_1^6 / 3) + (4 \times x_1 \times x_2) + (4 \times x_2^2) + (4 \times x_2^4)$	[0,1]	Minimize

VI. Results by Comparative Analysis of TA, SA and GA

Function Name	No. of Iterations			Time in Seconds		
	TA	SA	GA	TA	SA	GA
F1	1506	5924	100	15	30	5
F2	1000	1637	51	9	12	2
F3	1000	1462	52	7	8	2
F4	1000	2620	51	7	13	2
F5	5918	5940	100	25	25	4
F6	2000	3483	64	17	19	3
F7	2220	1160	51	18	7	2
F8	1196	3079	51	9	12	2
F9	1000	1168	51	5	7	2

VII. Conclusion

Genetic algorithm is probabilistic, heuristic, robust search algorithm premised on the evolutionary ideas of natural selection and genetic. Main idea behind the design of genetic algorithm is to achieve robustness and adaptiveness in real world complex problems. From the above results we have concluded that genetic algorithm is more reliable, strong and robust than threshold acceptance algorithm and simulated annealing algorithm.

VIII. Acknowledgement

I take this opportunity to express my immense gratitude to my Research Guide Dr. G. R. Kulkarni. I am hearty grateful to him for his prolonged interest, excellent guidance and constant inspirations. The success of this work would not be possible without his uncompromising demand for quality, and his reviews of my work have helped me lot to complete this research paper

References Références Referencias

1. Goldberg. D.E. (1989): "Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley".
2. M.F.Bramlette. Initialization, mutation and selection methods in genetic algorithm for function optimization. In R.K. Belew and L.B. Booker, editors, Proceedings of the Fourth International Conference on Genetic Algorithms, pages 100-107. Morgan Kaufman, 1991.
3. L. Davis. Adapting operator probabilities in genetic algorithms. In Proceedings of the 3rd International Conference on Genetic Algorithms, pages 61-69. Morgan Kaufmann, 1989.
4. K.A. De Jong and W. Spears. An analysis of interacting roles of population size and crossover in genetic algorithms. In H.P, Schwefel and R Manner, editors, Parallel problem solving from nature-Proceedings of 1st workshop, PPSN 1, Volume 496, pages 38-47, Dortmund, Germany, 1-3 1991. Springer-Verlag, Berlin, Germany.
5. K.A. De Jong and J. Sharma. Generation gaps revisited. In Darrell Whitley, editor, Foundations of Genetic Algorithms 2, pages 19-28. Morgan Kaufmann, 1992.
6. K.A. De Jong and W.M. Spears. A formal analysis of the role of multi-point crossover in genetic algorithms. Annals of mathematics and Artificial intelligence, 5:1-26,1992.
7. D.B. Fogel. Evolutionary Computation: Toward a New Philosophy of Machine Intelli¬gence. IEEE Press, 1995.
8. D.E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In Proceedings of the Second International Conference on Genetic Algorithms, pages 41-49, Massachusetts, 1987. Lawrence Erlbaum Associates.
9. M.E. thesis of Mr. Nilesh Ghambhava
10. Fundamentals of Algorithmics by Gilles Brassard and Paul Bratley
11. Artificial Intelligence (Second Edition) by Elaine Rich and Kevin Knight
12. www.iitk.ac.in/kangal/
13. comisef.wikidot.com/concept:thresholdaccepting
14. www.sciencedirect.com