International Journal of Advance Research in Science and Engineering
Vol. No.5, Special Issue No. (01), February 2016
www.ijarse.com
IJARSE
ISSN 2319 - 8354

PERFORMANCE COMPARISON BETWEEN GENETIC ALGORITHM AND POPULATION-BASED INCREMENTAL LEARNING ALGORITHM

Archana Kumari[1], U. Sai Priyanka[2],
Y. Sri Hasita[3], P.K.S Sandeep Kumar[4]

[1,2,3,4]*Computer Science and Engineering, Vignan's Institute of Information Technology, (India)*

ABSTRACT

This paper compares the performance of Genetic Algorithm and Population-based Incremental Learning algorithm on a simple mathematical function. GA and PBIL are evolutionary search based optimization algorithm which can arrive at near-optimal solution faster by using randomized techniques. These are stochastic search based algorithms that include randomness to escape local optima. We compare the two algorithms based on the accuracy of optimal value, number of generations and time taken to reach optimal value.

Keywords: Evolutionary computing, Genetic algorithm, optimization, performance comparison, population-based incremental learning.

I. INTRODUCTION

Lot of work has been done on GA and PBIL individually but there are very few resources that discusses about the two algorithms together. This work analyses and compares two evolutionary optimization algorithms - genetic algorithm (GA) and population based incremental learning (PBIL) and determine which of them is better.

GA is derived from Darwinian's theory of "survival of the fittest". It is efficient, adaptive and robust search process. It is guided by principles of evolution and natural genetic to perform randomized search and optimization.

The population based incremental learning algorithm (PBIL) is an Estimation of Distributed Algorithms, as proposed in. PBIL supposes that all the variables are independent. At each step of the algorithm a probability vector is maintained. Next generation is generated based on this probability vector.

Genetic algorithms, which generate solutions to optimization problems using techniques inspired by natural evolution, such as mutation, selection, and crossover whereas PBIL is an algorithm where the genotype of an entire population (probability vector) is evolved rather than individual members.

The performance analysis and comparison of the two algorithms is done by taking different parameters into consideration like number of iterations, inputs given, time complexity. Thus the algorithm which excels with

respect to the above mentioned parameters is considered as better for optimizing a problem with huge solution space.

The remaining part of the study is divided as follows. Section 2 discusses about optimization problem. Section 3 reviews genetic algorithm and is further subdivided into 3 subsections. Section 3.1 discusses selection operator, section 3.2 explains working of crossover operator and section 3.3 explains working of mutation operator. Section 4 is a brief discussion on PBIL. Section 5 explains the working of population-based incremental learning with the help of a small example. In section 6 we compare the performance of the two algorithms and finally section 6 concludes the paper.

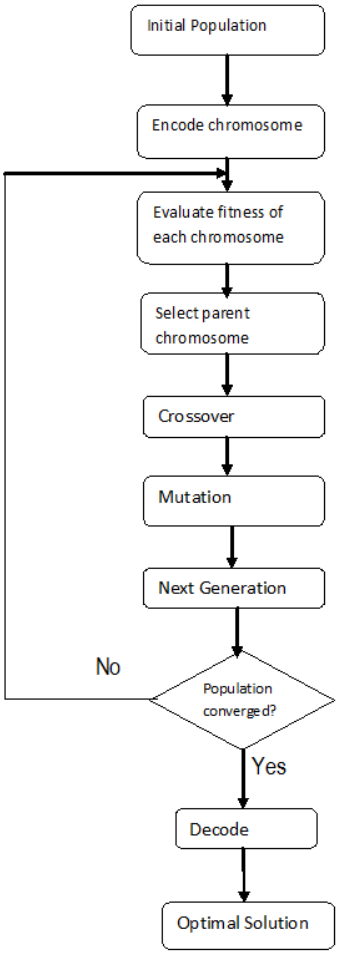
II. OPTIMIZATION

Optimization is a subject that deals with the problem of minimizing or maximizing a certain function in a finite dimensional Euclidean space over a subset of that space. In general it is a technique to select the best solution (with regard to some criteria) from a set of feasible solutions. Optimization has broad range of application in almost all fields. Most real world optimization problems involve complexities like multiple objectives, multimodal function, non-linearity and discontinuity. The search space (state space) may be so large that global optimum cannot be found in a reasonable time. The existing linear or nonlinear methods may not be efficient or computationally inexpensive for solving such problems.

Evolutionary algorithms can easily deal with such complexities. They can handle large search spaces, multi objective functions, multimodal functions. They use randomized techniques to arrive at solution faster which may otherwise take lifetime. They perform well even when there is little or no domain specific knowledge about the search space. They are preferred for problems with huge solution space, little or no domain specific knowledge and where little bit of inaccuracy, imprecision and uncertainty is an acceptable tradeoff for faster, low cost, practical and robust algorithm. Though randomized these algorithms use information from previous iterations and evolves the solution space to guide it towards the optimal solution. Evolutionary algorithms are stochastic search based technique which deliberately adds randomness to search process to avoid being caught at local optima.

III. GENETIC ALGORITHM

Genetic algorithms are often reviewed as function optimizer, although the range of problems to which genetic algorithms have been applied are quite board [1]. The GA begins with a population of chromosomes which are typically random. The population in GA is operated by three main operators: selection, crossover and mutation. These three operators are iteratively operated till the termination criterion met. The operators are as below in the GA.



3.1. Selection

Selection in GA is implemented by choosing parent chromosomes from the population according to their fitness. This operator makes more copies of better strings in the new population. These better strings of the current population and their multiple copies are inserted in a mating pool with the probability proportionality to its fitness. In simple GA, the population size is usually kept constant, thus the probability summation of each better string should be one. Strings are arranged into adjacent slots during the selection.

3.2. Crossover

Cross over operator recombines two strings to produce a better string, this is generally done at the string level. By combining material of two individuals of previous generation, the cross over implements this recombining process to create individuals which are different. From the mating pool, new strings are created by exchanging information among the strings. The two string exchanging the information to implement the crossover operation are the parent strings which results the new strings called as the children strings. Combining the subs strings of the parent strings to produce a better child string is beneficial.

3.3. Mutation

In GA, mutation operator facilitates exploration of search space and prevents getting trapped by local minima while crossover operator facilitates the population to converge on one good solution found so far. Mutation

introduces heterogeneity in the population when the population tends to produce the same outcomes due to iterative use of the selection and the crossover. Both crossover and mutation allow GA to explore and exploit the solution space. The objective of the mutation is to maintain diversity in the population, which may produce true optimum solutions.

In GA, mutation operator facilitates exploration of search space and prevents getting trapped by local minima while crossover operator facilitates the population to converge on one good solution found so far. Together they allow GA to explore and exploit the solution space. Different types of encoding techniques are available [2]. We used binary encoding which is a mapping from integers to binary system.

We use 5 bit chromosome. Accuracy obtained by 5 gene chromosome is 1/32 of solution space.

IV. POPULATION-BASED INCREMENTAL LEARNING

Population-Based Incremental Learning combines genetic algorithm with competitive learning [3]. It is a variant of GA which is simpler than GA and often outperforms GA. Unlike GA it doesn't involves application of complex operators (selection and recombination) instead it generates the next generation by updating a prototype vector (probability vector) based on its learning from present population. The probability of finding a 1 at i^{th} of solution vector is proportional to the i^{th} value in probability vector.

PBIL algorithm works as follows:

Step 1: A population of solution and a probability vector is generated randomly.

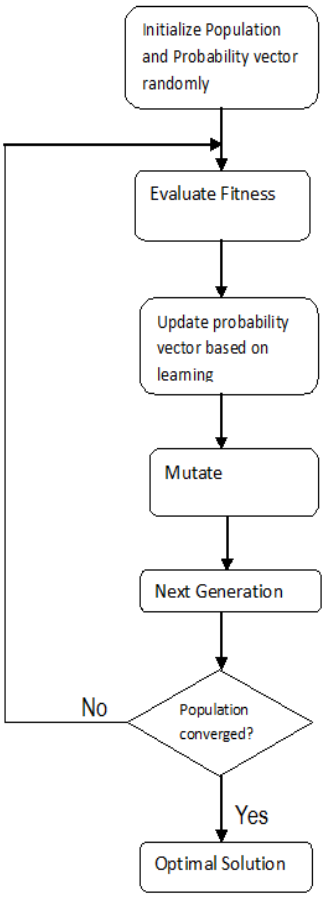
Step 2: The fitness of each individual is evaluated based on the function to be optimized and the individuals are ranked.

Step 3: The probability vector is then updated based on learning (either positive or negative learning) from current population.

Step 4: Mutation is performed with a very low probability.

Step 5: Create next generation using probability vector.

Steps 2 to 5 are repeated until the population converges to an optimal solution.



It can use either positive learning or negative learning to update probability vector. In positive learning the probability vector is moved towards the best solution vector and in negative learning the probability vector moves away from the worst solution vector. Typical value of learning rate lies between 0.1 and 0.4. In this study we took learning rate as 0.2, mutation probability as 0.05 and mutation shift as 0.05. We used positive learning. The mutation probability is low as too high mutation rate prevents population to converge to any optimum solution.

V. POPULATION-BASED INCREMENTAL LEARNING EXAMPLE

This section describes the working of PBIL with the help of a small example. The function to be optimized is $f(x) = x^2$. Our goal is to maximize f(x) within the interval [0 , 31]. Parameters are:

Population size: 4,

Learning rate: 0.2,

Mutation shift=0.05

We use 5 bits chromosome to encode numbers from 0 to 31.

Step 1:

Initial probability vector: [0.5, 0.5, 0.5, 0.5, 0.5]

Initial population:

Chromosome 1	**1**	**1**	**0**	**0**	**0**
Chromosome 2	**1**	**0**	**0**	**0**	**1**
Chromosome 3	**0**	**1**	**1**	**1**	**0**
Chromosome 4	**0**	**0**	**1**	**1**	**1**
Probability Vector (PV)	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**

Step 2: Evaluate fitness and choose best solution vector.

Chromosome 1	**1**	**1**	**0**	**0**	**0**	**576**
Chromosome 2	**1**	**0**	**0**	**0**	**1**	**289**
Chromosome 3	**0**	**1**	**1**	**1**	**0**	**196**
Chromosome 4	**0**	**0**	**1**	**1**	**1**	**49**
Probability Vector (PV)	**0.5**	**0.5**	**0.5**	**0.5**	**0.5**	

Step 3: Probability vector is updated based on learning. Here we use positive learning with learning rate 0.2. Probability vector is modified as [4]

$PV[i] = PV[i]*(1-LR) + SV_B[i]*LR$

Where PV[i] is i^{th} bit of probability vector in this generation,

LR is learning rate,

$SV_B[i]$ is i^{th} bit of best solution vector of this generation.

Updated PV is [0.6, 0.6, 0.4, 0.4, 0.4]

Step 4: Mutate probability vector as follows [4]

PV[i]=PV[i]*(1-MS)+Random(0 or 1)*(1-MS)

Where MS is mutation shift which describes the magnitude to which the PV will be affected by mutation.

Step 5: Create next generation of chromosome as follows.

The probability that a chromosome will have a 1 in i^{th} position is proportional to the value of probability vector's i^{th} bit. For this we generate a random number in the interval [0, 1]. If it is less than PV[i] then the chromosome will have a 0 for that bit else that bit is set to 1. This is repeated for each gene of each chromosome.

Above steps are repeated for specified number of generations or till the population converges to an optimal or near-optimal solution.

VI. COMPARISON OF GA AND PBIL

Population-based Incremental Learning performs better than Genetic Algorithm for the problem compared in this paper. PBIL produces more accurate results and in lesser time than GA. It is fast both in terms of number of generations evaluated and time taken to perform those evaluation. At the same time PBIL implementation is much more concise and simple than that of GA.

For this study we took population size as 4 and each chromosome is 5 bits long. The objective is to maximize the mathematical function $f(x) = x^2$ in the interval [0,31]. For GA we took crossover rate as 0.5 and mutation rate as 0.05. For PBIL we took learning rate as 0.2, mutation probability as 0.05 and mutation shift as 0.05.

Below table shows the optimal value selected by GA and PBIL in different iterations – run 1

	Iteration 1	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**	**Iteration 6**	**Iteration 7**	**Iteration 8**	**Iteration 9**
GA	**24**	**28**	**30**	**29**	**29**	**29**	**29**	**30**	**30**
PBIL	**14**	**29**	**23**	**31**	**31**	**31**	**30**	**30**	**30**

Below table shows the optimal value selected by GA and PBIL in different iterations – run 2

	Iteration 1	**Iteration 2**	**Iteration 3**	**Iteration 4**	**Iteration 5**	**Iteration 6**	**Iteration 7**	**Iteration 8**	**Iteration 9**
GA	**24**	**27**	**29**	**28**	**28**	**29**	**29**	**28**	**30**
PBIL	**14**	**30**	**26**	**31**	**29**	**30**	**31**	**31**	**31**

Thus it is evident from the above results is that PBIL produces more accurate results than GA.

VII. CONCLUSIONS

Population-based Incremental learning algorithm's better performance owes to the fact that it eliminates the need of application of operators like crossover and mutation which take up significant amount of time in GA evaluations. Another reason is that in PBIL the entire population (probability vector) is evolved rather than individual members thus it converges faster than GA. GA explores the solution space far too much and takes more time before zeroing down on a particular solution.

The main focus of this study is PBIL working as there is not much literature on it. In this study step by step process to solve PBIL has been given. This study can be further extended by examining affect of changing various parameters of GA and PBIL on their performance. But lot of work has already been done in this area [5][6].

REFERENCES

[1]. Tom V. Mathew, Genetic algorithm, Indian Institute of technology Bombay

[2]. Practical Genetic Algorithms by Randy L. Haupt, (2004), John Wiley & Sons Inc, Chapter 1 and 2, page 1-47.

[3]. Shumeet Baluja, Population Based Incremental Learning – A Method for Integrating Genetic Search Based Function Optimisation and Competitive Learning, *(Tech. Rep. No. CMUCS-94-163). Pittsburgh, PA: Carnegie Mellon University (1994)*

[4]. J. Bekker and Y. Olivier, Using the Population-based Incremental Learning Algorithm with Computer Simulation: Some Applications, (*South African Journal of Industrial Engineering May 2008 Vol 19(1): 53-71)* University of Stellenbosch, South Africa.

[5]. Shumeet Baluja & Rich Caruana**,** Removing the Genetics from the Standard Genetic Algorithm, (1995), CMU-CS-95-141, Pittsburgh, Carnegie Mellon University.

[6]. Avni Rexhepi , Adnan Maxhuni , Agni Dika, Analysis of the impact of parameters values on the Genetic Algorithm for TSP, IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No (2013).