A Study of Genetic Algorithm with Optimization

Mr. Madhumay Sen Asst. Professor, RCERT, Sitapura, Jaipur

Dr. Gaurav Kumar Jain, Asst. Professor, RCERT, Sitapura, Jaipur

Mr. Ravi Ranjan, Asst. Professor, RCERT, Sitapura, Jaipur

Abstract –

The Genetic Algorithm (GA) and its many versions have been popular in the academy and the industry mainly because of their intuitiveness, ease of implementation, and the ability to effectively solve highly nonlinear, mixed integer optimization problems that are typical of complex engineering systems. The drawback of the GA is its expensive computational cost. Particle Swarm Optimization (PSO) is a relatively recent heuristic search method whose mechanics are inspired by the swarming or collaborative behavior of biological populations. PSO is similar to the GA as these two evolutionary heuristies are populations-based search methods. In other words, PSO and the GA move from a set of points (population) to another set of points in a single integration with likely improvement using a combination of deterministic and probabilistic rules. This paper attempts to examine the claim that PSO has the same effectiveness (finding the true global optimal solution) as the GA but with significantly better computational efficiency (less function evaluations) by implementing statistical analysis and formal hypothesis testing. The major objective of this paper is to compare the computational effectiveness and efficiency of the GA and PSO using a formal hypothesis testing approach.

Keywords: Genetic algorithm. Candid solution, Hybrid PSO, Met heuristics, Numerical optimization, Stochastic Swarm.

I. Introduction

THE Genetic Algorithm (GA) was introduced in the mid 1970s by John Holland and his colleagues and students at the University of Michigan. The GA is inspired by the principles of genetics and evolution, and mimics the reproduction behavior observed in biological Populations.

The GA employs the principal of "survival of the fittest" in its search process to select and generate individuals (design solutions) that are adapted to their environment (design objectives/constraints). Therefore, over a number of generations (iterations), desirable traits (design characteristics) will evolve and remain in the genome composition of the population (set of design solutions Generated each iteration) over traits with weaker undesirable characteristics.

The GA is applied to solve complex design optimization problems because it can handle both discrete and continuous variables and nonlinear objective and constrain functions without requiring gradient information.

Particle Swarm Optimization (PSO) was invented by Kennedy and Eberhart in the mid 1990s while attempting to simulate the choreographed, graceful motion of Swarms of birds as part of a socio cognitive study investigating the notion of "collective intelligence" in biological populations. In PSO, a set of randomly generated solutions (initial swarm) propagates in the design space towards the optimal the optimal solution over a number of iterations (moves) based on large amount of information about the design space that is assimilated and shared by all members of the swarm. PSO is inspired by the ability of flocks of birds, schools of fish, and heads of animals to adapt to their environment, find rich sources of food, and avoid predators by implementing an "information sharing" approaches, hence, developing an evolutionary advantage.

II. GENETIC ALGORITHM

In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called phenotypes) to an optimization problem, evolves toward better solutions. Solutions are represented in binary as strings of 0s and 1s, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected

from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is them used in the next iteration of the algorithm. The algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

A typical genetic algorithm requires :

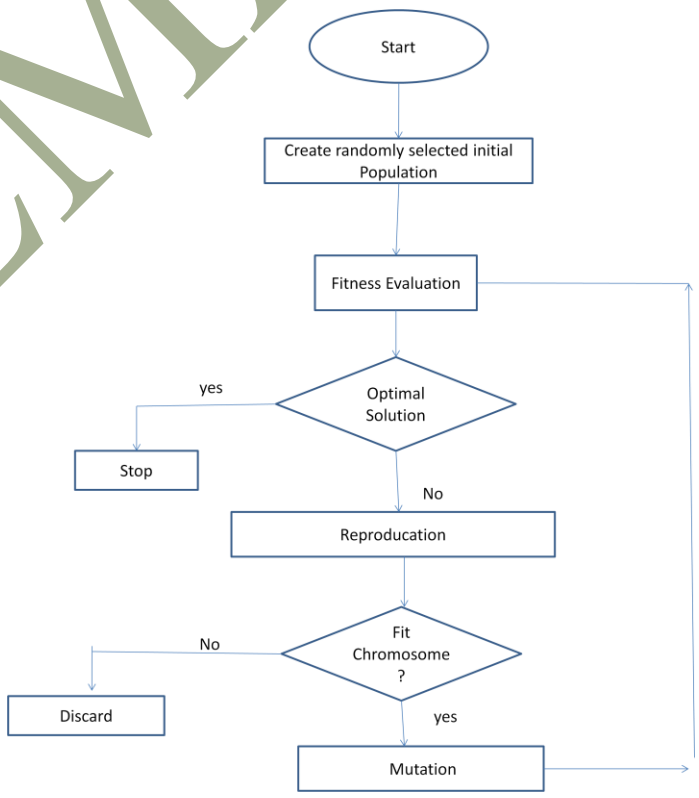
- A genetic representation of the solution domain
- A fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations.

Variable length representation may also be used, but crossover implementation is more complex in this case.

Tree-like representations are explored in genetic programming and graph-form representations are explored in evolutionary programming.

Objective Function of Genetic Algorithm : The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem, one wants to



Flow Chart of the Generic Algorithm

maximize the total value of objects that can be put in a knapsack of some fixed capacity.

A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack.

The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise, in some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

Once we have the genetic representation and the fitness function defined, GA proceeds to initialize a population of solutions randomly, and then improve it through repetitive application of mutation, crossover, and inversion and selection operators.

Implementation algorithm : The genetic algorithm uses the chromosomes fitness value to create a new population consisting of there fittest members. The flow chart of the GA is given in Fig. 1.

Applications : Genetic algorithms find application in bioinformatics, phylogenetics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields.

Learning Robot behavior using Genetic Algorithms : Robot has become such a prominent tool that it has increasingly taken a more important role in many different industries. As such, it has to operate with great efficiency and accuracy. This may not sound very difficult if the environment in which the robot operates remain unchanged, since the behavior of the robot could be pre-programmed.

However, if the environment is ever-changing, it gets extremely difficult, if not impossible, for programmers to figure out every possible behavior of the robot. Applying robot in a changing environment is not only inevitable in modern technology, but is also becoming more frequent. This led to the development of a learning robot.

III. PARTICLE SWARM OPTIMIZATION

The PSO was first designed to simulate birds seeking food which is defined as a "cornfield vector." The bird would find food through social cooperation with other birds around it (within its neighborhood). It was then expanded to multidimensional search.

Particle swarm optimization (PSO) is a computational method that candidate solution with regard to a given measure of quality. Such methods are commonly known as metaheuristics as they make few or no assumptions about the problem being optimized and can search very large spaces of candidate solutions. However, metaheuristics such as PSO do not guarantee an optimal solution is ever found.

PSO does not use the gradient of the problem being optimized, which means PSO does not require for the optimization problem to be differentiable as is required by classic optimization methods such as gradient descent and quasi-Nnewton methods. PSO can therefore also be used on optimization problems that are partially irregular, noisy, cange over time, etc (2).

PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae. The movements of the particles are guided by the best found positions in the search-space which are updated as better positions are found by the particles.

PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles are moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the serach-space as well as the entire swarm's best known position. When improved positions are being discovered these will then it will guide the movements of the swarm. The process is repeated and satisfactory solution will be discovered.

PSO variants : Various variants of a basic PSO algorithm are possible. New and some more sophisticated PSO variants are continually being introduced in an attempt to improve optimization performance. There is a

trend in that research; one can make a hybrid optimization method using PSO combined with other optimization techniques.

- Discrete PSO
- Constriction Coefficient
- Bare-bones PSO
- Fully informed PSO.

Application : The first practical application of PSO was in the field of neural network training and was reported together with the algorithm itself (kennedy and Eberhart 1995). Many more areas of application have been explored ever since, including telecommunications, control, data mining, design, combinatorial optimization, power systems, signal processing, and many others. PSO algorithms have been developed to solve :

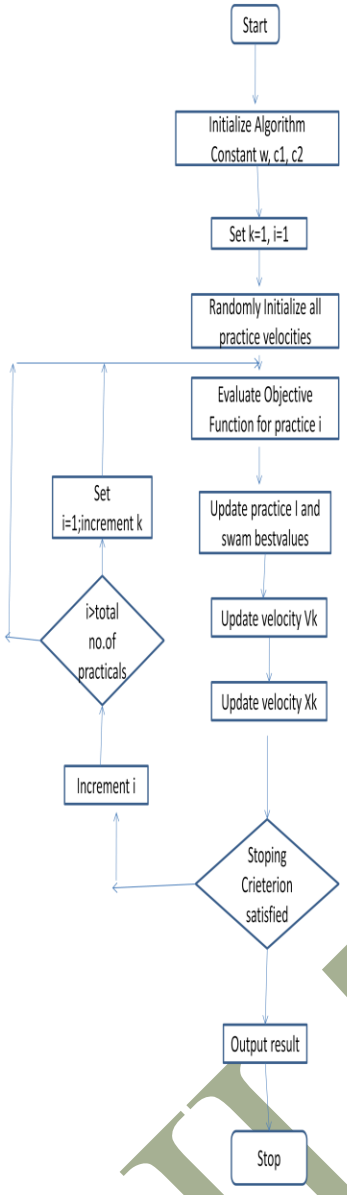
- Constrained optimization problems
- Min-max problems
- Multi objective optimization problems
- Dynamic tracking.

Implementation Algorithm : The PSO algorithm is simple in concept, easy to implement and computational efficient. Original PSO was implemented in a synchronous manner (Fig 2) but improved convergence rate is achieved by asynchronous PSO algorithm Figure.

IV GENETIC ALGORITHM VERSUS PARTICLE IN SWARM OPTIMIZATION

GA is inherently discrete, i.e. it encodes the design variables into bits of 0's and 1's, therefore it easily handles discrete design variables, and PSO is inherently continuous and must be modified to handle discrete design variables.

Unlike the GA with its binary encoding, in PSO, the design variables can take any values, even outside their side constraints, based on their current position in the design space and the calculated velocity vector.



Synchronous PSO Algorithm

V. CONCLUSION

Particle Swarm Optimization (PSO) is a relatively recent heuristic search method that is based on the idea of collaborative behavior and swarming in biological populations. SO is similar to the Genetic Algorithm (GA) in the sense that they are both population-based search approaches and that they both depend on information sharing among their population members to enhance their search processes using a combination of deterministic and probabilistic rules. Conversely, the GA is well established algorithm with many versions and application.

GA is very helpful when the developer does not have precise domain expertise, because Gas possesses the ability to explore and learn from their domain. PSO can be applied to multi objective problems, in which the fitness comparison takes pareto dominance into account when moving the PSO particles and non-dominated solutions are stored so as to approximate the pareto front.

The objective of this research paper is to test the hypothesis that states that although PSO and the GA on average yield the same effectiveness (solution quality), PSO is more computationally efficient (uses less number of function evaluations) than the GA.

VI. REFERENCES

1) Goldberg. D.E. (1989). Genetic Algorithms in search, Optimization and Machine Learning, Addison-Wesley.

2) Kennedy, J. and Eberhart, R., "Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia 1995, pp. 1942-1945.

3) Kennedy, J. and Eberhart, R., Swarm Intelligence, Academic Press, 1st ed., San Diego, CA, 2001.

4) Yuhui Shi., Particle Swarm Optimization. Electronic Data Systems, Inc. IEEE Neural Network Society Magazine, 2001.

5) R. Poli. Analysis of the publications on the applications of particle swarm optimization. Journal of Artificial Evolution and Applications, Article ID 685175, 10 pages, 2008.

6) R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. An overview. Swarm Intelligence, 1(1): pp 33-57, 2007.