CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

Comparative study on Particle Swarm Optimization and Genetic Algorithm

Kundan Kumar Jha – 22MSM40052

22SDT-652: Evolutionary Algorithm and Numerical Optimization

Branch - MSc Data Science
Department of Mathematics

Chandigarh University, Mohali Punjab

May 2022

Abstract

Researchers are paying a lot of attention to evolutionary algorithms as viable solutions to various efficient operation issues. The Genetic Algorithm (GA) is widely used in many domains due to its logic, ease of use, and capacity to resolve difficult issues frequently encountered in engineering systems. The GA's disadvantage is that it has a high implementation cost and typically calls for more iteration. A relatively new predictive technique called Particle Swarm Optimization (PSO) is based on the swarming behavior of real things. PSO and the GA are both evolutionary search methods, meaning that they switch from one set of points to another inside iteration while clearly improving upon the prior values making use of a combination of probabilistic and deterministic principles. The implementation, characteristics, and efficacy of these two evolutionary algorithms are examined in this research.

Keywords: Genetic algorithm (GA), Numerical optimization, Particle Swarm Optimization (PSO), Stochastic, Swarm.

1. Introduction

Any evolutionary algorithm can be used to tackle the majority of optimization problems. Genetic algorithms (GA) are one of the most significant classes of evolutionary algorithms. At the University of Michigan in the 1970s, John Holland introduced the idea of GA. Iterative processes are used by genetic algorithms, which are categorized as global search heuristics, to arrive at desired solutions. GA typically offers rough solutions to the various issues. GA makes use of a number of biological processes, including reproduction, crossover or recombination, mutation, and inheritance. Complex optimization issues can be resolved using GA because it can handle both discrete and continuous variables. In many different problems, including optimization, design and scheduling, power systems, data handling, etc., GA has proven to be quite effective. Swarm Intelligence (SI), a novel distributed paradigm, can also successfully address the optimization issues. In 1989, Gerardo Beni and Jing Wang invented the idea of SI. They were first motivated by biological examples such bacterial proliferation, ant colonies, animal herding, fish schooling, and bird flocking. On the basis of biological phenomena or systems, attempts were made to build various algorithms or distributed problem-solving tools. Kennedy and Eberhart created Particle Swarm Optimization (PSO) in the middle of the 1990s. Each particle in PSO represents a potential solution, which it updates based on two crucial types of information that are available during the decision-making process. The first (cognitive behavior) is learned

from one's own experience, while the second (social behavior) is learned from one's neighbors. In other words, both have made choices for themselves and have learned which ones their neighbors have made that stand out so far and how beneficial the best pattern of choices was. PSO has grown in popularity as a result of its many benefits, including its reliability, effectiveness, and simplicity. The PSO algorithm has been discovered to need less processing effort when compared to other stochastic algorithms. Even though PSO has demonstrated its potential in many areas for solving various optimization problems, it still takes a long time to run in order to find solutions for complex engineering issues.

2. Genetic Algorithm

One significant type of evolutionary algorithms is the genetic algorithm (GA). Prof. John Holland employed the genetic algorithm for the first time in 1975 (Holland, 1975). GA typically offers rough solutions to the various issues. GA makes use of a number of biological processes, including reproduction, crossover or recombination, mutation, and inheritance. Because real-coded GA does not require binary encoding or decoding, it is typically faster than binary GA. This algorithm's different steps include the following:

1) Create an initial population unintentionally or intentionally.
2) Determine each person's fitness level within the population.
3) Assign each member's selection probability so that it is inversely correlated with its fitness value.
4) Create the following generation from the current one by choosing the desired persons to have children.
5) Continue until an appropriate answer is discovered.
6) According to GA, a population is a group of particles, and a chromosome is a single particle. The cost function, also known as the fitness function, is then used to assess these chromosomes. The objective function of the task at hand is typically the cost function. Several of the procedures connected with GA include:
    a) Selection - Using the fitness criterion, this procedure often selects the chromosome that will continue to replicate.
    b) Reproduction: From the present generation, the following generation is created by this phase.
    c) Crossover: This procedure allows genetic material to be transferred between chromosomes.
    d) You can utilize a single or several crossing points.
    e) Mutation: In this process, the chromosomes of a single person alter. The algorithm is kept from getting stuck at a particular moment by mutation.
    f) The last step in GA is the stopping criteria. When the desired outcome is reached or the maximum number of cycles is reached, the iteration comes to an end.

Implementation Algorithm: By utilizing a predetermined fitness function, GA causes the formation of the fittest members after each iteration. The Genetic Algorithm's fundamental flowchart is shown in Figure 1.

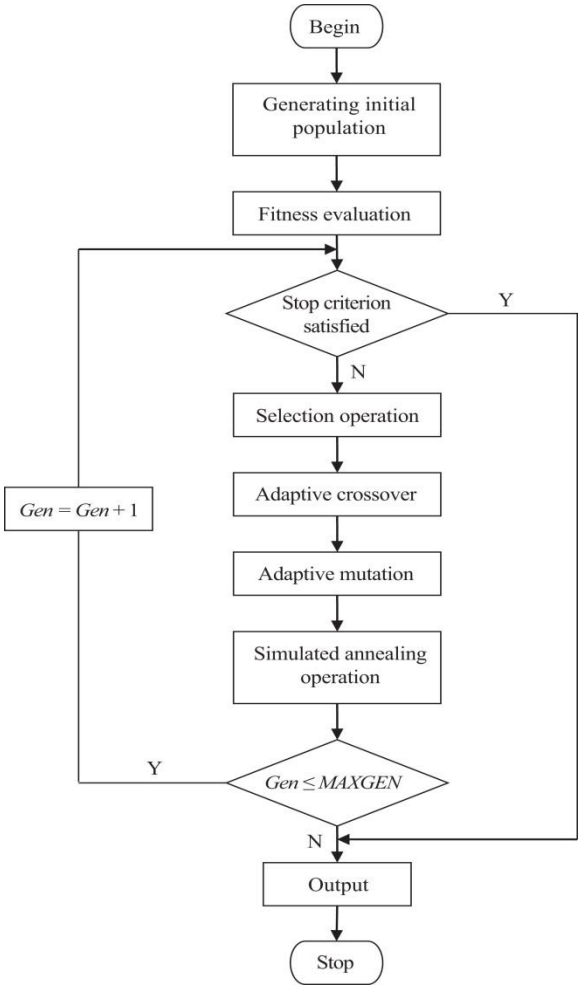

Figure 1: Basic implementation of GA

Applications: A wide range of disciplines can make use of genetic algorithms. It is mostly employed to address optimization issues. Bioinformatics, computational science, electrical engineering, manufacturing, and phylogenetics are a few of the industries that employ GA.

3. Particle Swarm Optimisation

A computational technique called particle swarm optimization (PSO) seeks to solve a problem more effectively by repeatedly attempting to make a candidate solution better in terms of a specified criterion for quality. Since they can search very large spaces of potential solutions and

make few or no presumptions about the problem being optimized, such techniques are frequently referred to as metaheuristics. Metaheuristics like PSO, though, do not ensure that an ideal solution will ever be identified.

Since PSO does not use the problem's gradient, it does not require that the optimization problem be differentiable, as do more traditional optimization techniques like gradient descent and quasi-Newton methods. As a result, PSO can be applied to optimization issues that are noisy, partially irregular, change over time, etc. By using a population of potential solutions, referred to as particles, and moving them across the search area in accordance with straightforward mathematical rules, PSO optimizes a problem. The best positions that the particles have found in the search area, which are modified as improved positions are discovered by the particles, serve as guidance for their travels. The PSO method relies on a population of potential solutions, or "particles," collectively referred to as "swarms."

A few basic equations are used to move these particles around in the area of search. The positions of the particles in the swarm, as well as their best known positions within the search space, serve as guidance for their travels. The swarm will be guided in its moves after better places have been identified.

Once the process is repeated, a workable solution will be found.

PSO Variants: There are numerous variations of the fundamental PSO algorithm. In an effort to boost optimization performance, new and occasionally more complicated PSO variations are constantly being introduced. There is a pattern in that research; PSO can be integrated with other optimization techniques to create a hybrid optimization strategy.

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

Applications: PSO's first real-world application, which was published alongside the method originally (Kennedy and Eberhart 1995), was in the area of neural network training. Since then, numerous additional application fields have been investigated, including internet access, management, data mining, design, combinatorial optimization, power systems, signal processing, and many others. PSO algorithms were created to address:

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

Implementation Algorithm: The PSO algorithm is straightforward conceptually, simple to apply, and effective computationally. The initial PSO was done synchronously, whereas asynchronous PSO leads to a higher convergence rate. The PSO algorithm is shown in Figure 2.

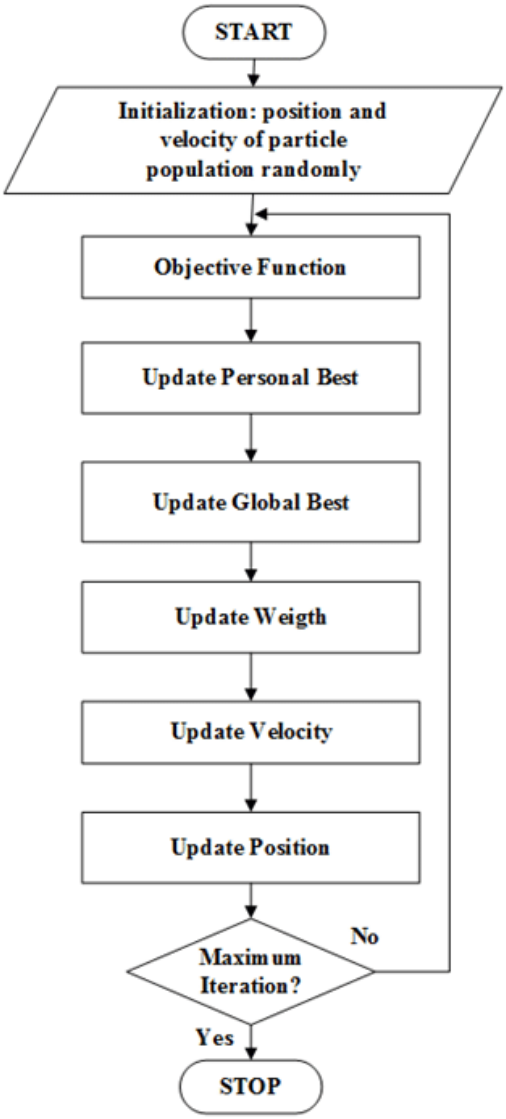

Figure 2: Basic implementation of PSO

The various steps used in the PSO algorithm are given below:

1) Place the particles at random initial speeds and locations in a search space.

2) Begin figuring out the swarm particle's fitness function's matching value.

3) Compare the fitness level evaluation to the particle's current pbest value. If current value is superior to pbest, set it as the new pbest value and change the pbest location in the n-dimensional space to the current location;

4) Compare the fitness value to the prior all-time high. gbest is reset to the array index and value of the current particle if the current value is superior to it;

5) Finally, change these numbers to match the swarm particle's position and speed.

6) The particle adjusts its velocity and locations using equations (i) and (ii) after determining the two optimum values.

v[] = v[] + c1 * rand() * (pbest - present) + c2 * rand( ) * (gbest - present).....(i)

Present[] = Present[] +v[.....] is the definition of present[]..... (ii)

Present [] is the current particle (solution), and v [] is the particle velocity. The definitions of pbest and gbest are the same as before. A random number between 0 and 1 is called a rand (). The learning factors c1 and c2 are. Commonly, c1 = c2 = 2.

The standard process is as follows:

For each particle.

- Initialize particle
- Do
  - For each particle
  - Calculate fitness value.
  - If the fitness value is better than the best fitness value
  - (pbest) in history.
  - Set current value as the new pbest
- End.
- Choose the particle with the best fitness value of all the particles as the gbest for each particle
- Calculate particle velocity according equation (a)
- Update particle position according equation (b)

End (while maximum iterations or a minimum error criterion is not attained)

Advantages of PSO:

- Over a single population, there is more diversity and exploration.
- The influence of momentum on particle motion may enable faster convergence.
- PSO is a form of parallel optimization technique that provides more diverse and interesting trajectory options.

Both methods begin with a collection of a population that is formed at random; they both use fitness values to assess the population. Using random approaches, both the population is updated and the optimum is sought. Success is not guaranteed by any system. Crossover and mutation are two examples of genetic operators that are absent from PSO. With respect to internal velocity, particles update themselves. Additionally, they have memory, which is crucial to the algorithm. The information exchange technique of PSO differs greatly from genetic algorithms (GAs). Chromosomes communicate with one another in GAs. So, the entire population moves as a single unit in the direction of the ideal location. Only gbest (or lbest) can divulge knowledge to others in PSO. It is a one-way technique for exchanging information. Only the best solution is sought after by evolution. In contrast to GA, all particles typically converge to the ideal outcome rapidly, despite the local form.

Limitations of PSO:

- PSO is a continuous method that does not work well for combinatorial issues.
- PSO is an a single direction mechanism for transferring information.
- The algorithm's comprehensiveness is negatively impacted by PSO's lack of support for GA operators, and there is no assurance of an optimal outcome.
- Tool under stressful condition may collapse
- Occasionally, algorithms may catch local minima or local maxima.

4. Comparison of Particle Swarm Optimization & Genetic Algorithm

Both GA and PSO are population-based optimization algorithms that have their strengths and weaknesses. Table summarizes the comparison of GA and PSO based on various criteria.

Criteria	Genetic Algorithm (GA)	Particle Swarm Optimization (PSO)
Working principle	Evolution	Swarm behavior
Search space	Continuous/discrete	Continuous
Convergence rate	Slow	Fast
Handling non-linear functions	Good	Good
Handling discrete variables	Good	Poor

Criteria	Genetic Algorithm (GA)	Particle Swarm Optimization (PSO)
Handling multi-modal functions	Good	Poor
Computational complexity	High	Low
Robustness to noise	Good	Poor
Diversity in solutions	Good	Poor
Local optima	May get stuck	May get stuck
Parameter tuning	Time-consuming	Relatively easy

Working Principle

The working principle of GA is based on the concept of evolution, where a population of solutions evolves over generations to find the optimal solution. On the other hand, PSO is based on the behavior of particles in a swarm, where each particle moves towards the best position found by itself and other particles in the swarm.

Search Space

Both algorithms can handle continuous variables, but GA can also handle discrete variables. This makes GA more suitable for problems that involve discrete variables.

Convergence Rate

PSO has a faster convergence rate compared to GA, which makes it more suitable for problems that require quick solutions. However, the faster convergence rate of PSO may lead to premature convergence to local optima.

Handling Non-linear Functions

Both algorithms are good at handling non-linear functions.

Handling Multi-modal Functions

GA is better at handling multi-modal functions compared to PSO, as it can find multiple solutions in a single run.

Computational Complexity

GA is computationally more intensive compared to PSO, as it requires a larger population size and more generations to converge.

Robustness to Noise

GA is more robust to noise compared to PSO, as it can maintain diversity in the population and avoid getting stuck in local optima.

Diversity in Solutions

GA is better at maintaining diversity in the population and finding multiple solutions, while PSO may converge to a single solution.

Local Optima

Both algorithms may get stuck in local optima, which is a common problem in optimization algorithms.

Parameter Tuning

Parameter tuning is time-consuming in GA, as it requires adjusting the population size, crossover rate, and mutation rate. PSO has fewer parameters to tune, which makes it relatively easier to use.

5. Conclusion

A relatively new heuristic search technique called particle swarm optimization (PSO) is based on the notion of cooperative behavior and flocking in biological populations. In that both PSO and the Genetic Algorithm (GA) are population-based search algorithms and both rely on information exchange among the members of their populations to improve their search processes by using the combination of determined and probabilistic criteria, they are comparable to each other. Contrarily, the GA is a well-known method with several iterations and applications. While both GA and PSO are significant components of evolutionary optimization algorithms, they also have drawbacks that restrict their applicability to a small number of situations. . This study's goal is to evaluate the idea that, even though PSO and GA often produce similar efficacy (solution quality), PSO is more computationally effective than GA. PSO is a population-based optimization method, and both methods cannot ensure success. Both systems refresh the population and use random procedures to search for the optimum. PSO lacks evolution operators like crossover and mutation, in contrast to GA. Particles in PSO update themselves based on their internal velocities. They possess memory as well, which is crucial to the algorithm. Additionally, by emulating the current optimum particles, the potential solutions, or "particles," are "flown" through the problem space.

References:

[1] J. H. Holland, ―Genetic algorithms and the optimal allocation of trials, SIAM Journal on Computing, vol.2, no. 2, pp. 88–105, 1973.

[2] R. Kennedy, J. Eberhart, ―Particle swarm optimization, Neural Networks, 1995. Proceedings., IEEE International Conference on Neural Networks, Perth, WA, Australia, vol. 4, pp. 1942– 1948, 1995.

[3] A. Engelbrecht, ―Fundamentals of computational swarm intelligence, 2006, Hoboken: John Wiley & Sons, Ltd

[4] O. Maimon and D. Braha, ―A genetic algorithm approach to scheduling pcbs on a single machine,‖ International Journal of Production Research, vol. 36, no. 3, pp. 761–784, 1998.

[5] J. Zhang, H. Chung, and W.-L. Lo, ―Pseudocoevolutionary genetic algorithms for power electronic circuits optimization,‖ Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, vol. 36, no. 4, pp. 590– 598, 2006.

[6] J. Kennedy and R. Mendes, ―Population structure and particle swarm performance,‖ 2002.

[7]J. Kennedy, J. F. Kennedy, and R. C. Eberhart, Swarm intelligence. Morgan Kaufmann, 2001.

[8] Almufti, S., Marqas, R., & Asaad, R. (2019). Comparative study between elephant herding optimization (EHO) and U-turning ant colony optimization (U-TACO) in solving symmetric traveling salesman problem (STSP). Journal Of Advanced Computer Science & Technology, 8(2), 32. https://doi.org/10.14419/jacst.v8i2.29403.

[9] Asaad, R., Abdulnabi, N. (2018). Using Local Searches Algorithms with Ant Colony Optimization for the Solution of TSP Problems. Academic Journal of Nawroz University, 7(3), 1-6. https://doi.org/10.25007/ajnu.v7n3a193.

[10] Coello, Carlos. "An updated survey of GA-based multiobjective optimization techniques." ACM Computing Surveys, vol.32, no.2, p.109-143 (June 2000). https://doi.org/10.1145/358923.358929.