International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Impact Factor (2012): 3.358

A Survey: Swarm Intelligence *vs.* Genetic Algorithm

Keisam Thoiba Meetei

Galgotias University, School of Computing Science and Engineering, Plot No.2, Sector 17-A
Yamuna Expressway, Greater Noida, GautamBuddh Nagar, Uttar Pradesh, India

Abstract: ***This paper gives simple introduction of Genetic Algorithms and Swarm Intelligent Algorithms. The theoretical concept and application of these two intelligent techniques in solving complex problem are also discussed. Comparison between these two intelligent techniques is also shown. Genetic Algorithms and Swarm Intelligent Algorithms are meta-heuristic search algorithms that emulate the living systems so to make computer programs the ability to learn. Both these two algorithms own biological plausibility. These algorithms are generally used to find out useful solutions to problems like optimization and searching.***

Keywords: Meta-Heuristic, Biological, Genetic, Swarm.

1. Introduction

This paper mainly discusses the theoretical concept and practical implementations of Genetic Algorithm and Swarm Intelligence Algorithms. The concept of these two algorithms is mainly based on the biological behaviors of natural objects. Both these algorithms are inspired by evolution. These two Artificial Intelligence techniques are mainly used as a search technique in order to find optimal solutions to optimization and search problems. Swarm Intelligence (SI) is a study of the collective behavior emerged from social insects, animals working under very few rules. It is a decentralized technique with self-organized properties [1]. Swarm Intelligence Algorithm is mainly used to find optimal solutions of complex problems. Genetic Algorithms (GAs) are adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics. It is a sub-part of or belongs to Evolutionary Algorithms and a rapidly growing area of Artificial Intelligence. Genetic Algorithm is applied to solve complex design optimization problems.

2. Swarm Intelligence

In Computer Science and Artificial Intelligence (AI), Swarm Intelligence is a field which is considered as an important concept with emergent properties that designs and studies efficient computational methods for solving complex problems. It is inspired especially by the behavior of biological systems like real swarms or insect colonies. SI is the collective behavior of decentralized nature of self-organized systems. It is natural or artificial. This SI concept is applied in many works on AI. Gerardo Beni and Jing Wang introduced this concept in 1989, in the context of cellular robotic systems. SI systems consist of a population of simple agents or boids that follow very simple rules by interacting locally with one another and with their environment. As there is no centralized control structure dictating how individual agents should behave, the agents work randomly and interactions between such agents lead to the emergence of "intelligent" global behavior, unknown to the individual agents. Ant Colonies, Bird Flocking, Animal herding, Bacterial Growth and Fish Schooling are some natural system example of SI. This paper will discuss two algorithms of Swarm Intelligence: Ant Colony Optimization and Particle Swarm Optimization.

2.1 Ant Colony Optimization

In computer science and operations research, the Ant Colony Optimization Algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. This algorithm was proposed by Marco Dorigo in 1992. The most recognized example of swarm intelligence in real world is the ants. Ants usually wander randomly through space searching for food source. Ants will start searching from their colony moving randomly in all directions. If an ant finds food, then the ant returns to colony. While returning back towards colony the ant will leave a trail of chemical substances called pheromone along the path. By doing so other ants can detect the pheromone left by the previous ant and they follow the same path. The path is determined by the amount of concentration of pheromone along the path. It is by nature that the pheromone will start evaporating over time reducing the concentration of pheromone thus leading to poor strength of the path. Considering this condition, a shorter path will be suited for ants [4]. Every ant will follow the shorter path thus keep adding pheromone which makes the concentration strong enough to against evaporation. This makes the emergence of shortest path from colony to food [2] [3]. Seeking a path between the colony and a food source is the natural behavior of ants. Based on this biological behavior, the algorithm was developed aiming to search for an optimal path in a graph. The fundamental approach underlying ACO is an iterative process. ACO is a paradigm for designing Meta heuristic algorithms for combinatorial optimization problem [5] [6].

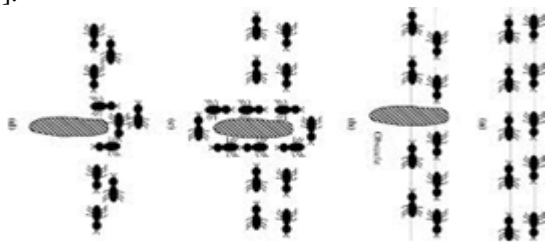

Figure 1: Selection of shortest path by ants

ACO Algorithm

Suppose a graph *Gr* with vertices V and edges E be a connected graph with *n* numbers of nodes. Let s_v be the source and d_v be the destination node. Applying ACO meta-heuristic, the shortest optimum path between s_v and d_v on the graph *Gr* can be calculated. The path length in the graph *Gr* is defined by the number of nodes in the path. Every edges*ed*(x, y) which belongs to E connects vertices v_x and v_y. These edges *ed* contains a pheromone variable $\phi_{x,y}$. Every time the ants visit the nodes, the variable $\phi_{x,y}$ gets a modification in its quantity where every ants deposit their pheromones. The deposition of the pheromone $\phi_{x,y}$ is giving a clear message indicating that the edge was used by other ants. The ant which on the node v_x will use the pheromone variable $\phi_{x,y}$ of the node $v_y \in N_x$ to calculate the probability of finding the next node. Using this probability the ants moves to the next node v_y. N_x contains all the possible neighbours of v_x[1].

$$P_{x,y} = \begin{cases} \frac{\varphi x,y}{y \in Nx\ \varphi x,y} & if\ y \in Nx \\ 0 & if\ y \notin Nx \end{cases} \quad (1)$$

$P_{x,y}$ is the transition probabilities of node v_x. This transition probability satisfies the constraint:

$$\sum_{y \in Nx} Px, y = 1, \quad x \in [1, N] \quad (2)$$

When searching for the next route, every ants deposit the pheromone on the edges*ed* of the connected graph *Gr*. A constant amount $\Delta\phi$ of pheromone is added to the existing pheromone. The overall quantity of the pheromone existing on the edge $ed(v_x, v_y)$ changes whenever an ant moves from v_x to v_y:

$$\varphi_{x,y} := \varphi_{x,y} + \Delta\varphi \quad (3)$$

The same way natural pheromone evaporates over time, the artificial pheromone concentration also reduces with time to inhibit a fast convergence of pheromone on the edges[1]. In ACO meta-heuristic, this happens exponentially:

$$\varphi_{x,y} := (1-q)\ .\ \varphi_{x,y}, q \in (0,1] \quad (4)$$

Basic Pseudo-code

Initialization
while(termination condition not met) do
Construct Ant Solutions
Apply Local Search(optional)
Global Update Pheromones
End

2.1.1 Application

- water irrigation grid's multiple objective design
- GPS geodetic grids optimization
- protein folding prediction by its amino-acid chains
- allocation of data in supercomputer's memory

2.1.2 Advantages

- Inherent parallelism
- Rapidly discover good solutions with positive feedback
- Efficient for small number of nodes Traveling Salesman Problem and other similar optimization problems
- Can be applied in dynamic applications
- Guaranteed convergence

2.1.3 Disadvantages

- Difficult in theoretical analysis
- Random decisions (dependent)
- Change in probability distribution by iteration
- Convergence time is uncertain

2.2 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a heuristic optimization method. It was developed by Doctor Kennedy and Eberhart in the year 1995. PSO gets the inspiration from the sociological movement behavior associated with birds and fish flocking [1][7]. It is also a population based stochastic approach that was developed to solve non-linear continuous and discrete complex search or optimization problems[1]. In PSO, the particles are generally a group of software agents. These software agents are then used to search for good solutions to a given discrete or continuous optimization problems. While searching for good solutions each particle is considered a solution of the given problem. In order to give the best solutions, while searching, the particles use collective experiences, one is their own best experiences and the other is neighbor particle's[8]. The collective experiences are evaluated and then after evaluation, the particles choose the next move in the search space. PSO works in the following ways-

- First, an initialization phase is created and each particle is given a randomly selected initial position, an initial velocity.
- During its movement from one state to another, the best position is stored in the particle's memory. At each iteration of this algorithm, each particle moves with a velocity that is a sum of weights of three components: the old velocity, a velocity component that lead the particle towards a location in the search space where the particle's best solution is found so far, and a velocity component that lead the particle towards the location in the search space where the best solution of the neighbor particles found so far.

Figure 2: Bird Flocks

Algorithm of PSO

$v_a \leftarrow v_a + €_1 * (p_a - x_a) + €_2 * (p_g - x_a)$ (5)

$x_i \leftarrow x_i + v_i$ (6)

where

x_a is the current position of the a-th particle in the swarm;
v_a is the velocity of the a-th particle;
p_a is the personal best position found by the a-th particle;
p_g is the global best position taken from the experience of the neighborhood particle, i.e.,;
The symbol * is a point-wise vector multiplication;
$€_1 = c_1 r_1$ and $€_2 = c_2 r_2$;
$r_1$and r_2 are two vectors of random numbers uniformly chosen from [0, 1];
c_1 and c_2 acceleration coefficients.

According to equation (5):

$v_i \rightarrow$ momentum

$€_1 * (p_i - x_i) \rightarrow$ cognitive component

$€_2 * (p_g - x_i) \rightarrow$ social component

Velocity v_i(i-th particle velocity) is determined with the help of three components:

- **momentum-** the previous velocity of the particle,
- **cognitive** component- the best position of the particle,
- **social** component- the best global position.

Basic Pseudo-code

Generate an initial population of agents randomly
Repeat
for i= 1 to population_size*do*
if $f(x_i) < f(p_i)$ then $p_i = x_i$;
$p_g = \min(p_{neighbours})$;
*for*d= 1 to dimensions *do*
velocity_update() ;
position_update() ;
end
end
until termination.

2.2.1 Application

- Human tumor analysis
- Pressure Vessel
- Real-time training of neural networks
- Reactive power and voltage control

2.2.2 Advantages

- Calculation is easy to perform
- Small number of parameters to adjust
- High efficiency in global searching

2.2.3 Disadvantages

- At the stage of redefined search, convergence is slow
- Weak in local search
- Cannot work with non- coordinate system

3. Genetic Algorithms

Genetic algorithms (GA) are part of AI. These algorithms also belong to the family of evolutionary algorithms[9][10]. Inspired by evolution natural behavior like selection, inheritance, crossover, mutation at the level of cells, Genetic Algorithm was developed by Goldberg aiming to find the best solution in solving optimization problems by mimicking such natural behavior. This algorithm works by creating set (population) of candidate solutions which are also called individuals, at the initial stage. Every candidate solution has its own set of properties (chromosomes) and these properties alterable and mutated. Generally binary strings i.e., 0s and 1s are used to represent these solutions but are not limited to this; other way of encodings can also be done. The evolution starts from a set of individuals that are randomly generated. This process is iterative and each iteration is called a generation. Every individual has a fitness which is the objective function's value at each generation and this fitness is to be evaluated. Based on the fitness, individuals are selected from the current set and a new generation is formed by modifying (mutated)the individuals properties. Now, in the next iteration the new generated individual is used. The algorithm is terminated after it reaches the highest number of generations or after reaching a fitness satisfaction level.

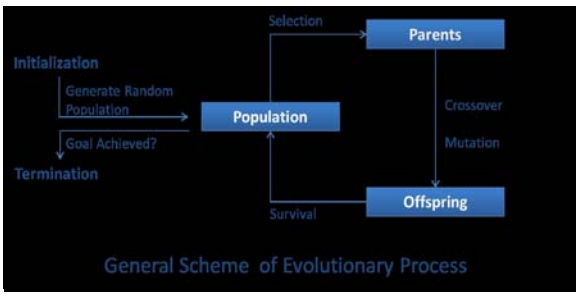

Figure 3: Evolutionary process of Genetic Algorithms

GA Algorithm

- **[Start]** Start with randomly generating population of *n* Chromosomes
- **[Repeat]** Until the satisfied condition is reached.
- **[Find Fitness]** For each chromosome *i* find the fitness *f(i)*
- **[Generate New p opulation]** Follow the below steps to generate a new population. Repeat it until a new and fully complete population is created.

a) **[Chromosomes Selection]** Get the evaluated fitness value of chromosomes and according to it select the best two parent chromosomes (chromosomes with better fitness gets the higher chance of being selected).

b) **[Crossover]** Create new offspring(children) by using crossover probability crossing over the parents.

c) **[Mutation]** Mutate new offspring using mutation probability, at each position.

d) **[Accepting]** New population gets new offspring.

- **[Replace]** Run the algorithm further using the newly generated chromosome population
- **[Test]** If the satisfied condition is reached, stop and return the best chromosome that is in the current population.
- **[End Repeat]**
- **[Solution]** Best Chromosomes.

Basic Pseudo-code

Begin;
Generate population of *n* chromosomes randomly;
For each individual$i \in n$: calculate fitness *f(i)*;
For *i* = 1 to total number of generations;
Select an operation randomly (crossover or mutation);
If crossover;
Select two parents at random i_a and i_b;
Generate on offspring i_c= crossover (i_a and i_b);
Else If mutation;
Select one chromosome *i* at random;
Generate an offspring i_c = mutate (*i*);
End if;
Calculate the fitness of the offspring i_c;
If i_c is better than the worst chromosome then
replace the worst chromosome by i_c;
Next *i*;
Check if termination = true;
End;

3.1 Application

- Automatic Programming
- Machine and Robot Learning
- Economic Models
- Immune System Model

3.2 Advantages

- Optimization problems are solved with multiple solutions
- Can solve problems with non-parametrical problems, multi-dimensional, non-continuous and non-differential
- Optimization problem that can be explained with chromosome encoding can be solved using GA
- Large number of solution set can be quickly scanned

3.3 Disadvantages

- No guarantee that GA will always give a global optimum
- Cannot assure that GA will give constant optimization response time
- Applications of GA in real time system controls are limited due to convergence and random solutions
- Cannot use GA in dynamic problems

4. Comparisons

The three algorithms defined above have their own way of solving complex optimization problems. Each algorithm shows their performance in finding the best solution, depending on the problems. Due to the differences in their workings, they can be compared among each other.

ACO* vs *PSO

- Mechanism of communication among ants in ACO is indirect means interaction is done through environment (adaptable). Communication among PSO particles is direct (not adaptable)
- Originally ACO was designed to solve discrete optimization problem and later modified to accommodate continuous problems also. PSO is opposite to ACO. It was designed to solve continuous problems but modified later to accommodate discrete or binary optimization problems.
- Typically the solution space of ACO is a construction graph (weighted graph). The solution space of PSO is set of n-dimensional points.

GA* vs *ACO

- GA basically solves problems where there is no pre-determined shape, size and complexity. This means that analytical knowledge is not required but still gives accurate result. In ACO, source and destination are pre-defined and specific.
- GA works locally with stable system (static). GA suffers in case of unstable system. So here comes ACO for dynamic or unstable problems. ACO uses the idea of self-organizing principles.

GA* vs *PSO

- GA was designed basically for discrete optimization problem where bit of 0's and 1's are used to encode discrete design variables. But PSO was designed for continuous problems and can choose any value to encode design variables. In PSO, the previous and the next position of a particle at each point are defined uniquely and clear.
- Unlike GA, PSO has no any calculation method that can be considered systematical, and there is no any mathematical foundation that is definite.

5. Conclusion

Genetic Algorithm and Swarm Intelligence are the two intelligent techniques currently using as a heuristic method for solving complex problems (like NP-Hard) that are hard to find solutions using normal existing technique. Every algorithm discussed has merits and demerits. They worked with different structures in different environments. These algorithms do not guarantee that it will always give optimal solution. In order to enhance the capability of solving problems and improve their performance, hybrid algorithms are being developed by combining these algorithms among each other. Their applications are still in exploration. Some of their well-known applications are network management, robotics, neural networks, machine learning, traffic control etc.

Among these algorithms, PSO is the newest and research on it is at the beginning stage. PSO has no any systematical calculation method or any definite mathematical foundation but still widely used in various area using its simple calculation technique. PSO is still weak in theoretical foundation so most of its optimal solution cannot guarantee in theory. Further research in PSO's theoretical foundation should be done in order to bring it in perfection. Its application areas should be explored further.

GA works only in stable environment with discrete variables. It should have the ability to work in dynamic environment with continuous variable and it should guarantee convergence so further improvement is required. Global search technique should also be implemented. In ACO starting and destination node is defined earlier. ACO's problem areas should be extended further by making it capable to solve problems that do not define its size, shape and complexity in advance.

References

[1] Thakare, S. A. "Comparison of Swarm Intelligence Techniques." International Journal of Computer Science and Business Informatics 1.1 (2013).
[2] RakaJovanovic and Milan Tuba, "An ant colony optimization algorithm with improved pheromone correction strategy for the minimum weight vertex cover problem", Elsvier, Applied Soft Computing, PP 5360-5366,2011.
[3] Laalaoui Y, Drias H, Bouridah A and Ahmed R B, " Ant Colony system with stagnation avoidance for the scheduling of real time tasks", Computational Intelligence in scheduling, IEEE symposium, 2009, pp 1-6.
[4] David C Mathew, "Improved Lower Limits for Pheromone Trails in ACO", G Rudolf et al(Eds), LNCS 5199, pp 508-517, Springer Verlag, 2008.
[5] Yaseen, SaadGhaleb, and Nada MA AL-Slamy. "Ant colony optimization." IJCSNS 8.6 (2008): 351.

[6] Al Salami, Nada MA. "Ant colony optimization algorithm." UbiCC Journal 4.3 (2009): 823-826.
[7] Bai, Qinghai. "Analysis of Particle Swarm Optimization Algorithm." Computer & Information Science 3.1 (2010).
[8] Poli, Riccardo, James Kennedy, and Tim Blackwell. "Particle swarm optimization." Swarm intelligence 1.1 (2007): 33-57.
[9] Mathew, Tom V. "Genetic algorithm." Department of Civil Engineering, Indian Institute of Technology, Bombay (2005).
[10] Joshi, Krutika D., and Amit A. Pandya. "GENETIC ALGORITHMS AND THEIR APPLICATIONS–TRAVELING SALES PERSON AND ANTENNA DESIGN."
[11] Hermawanto, Denny. "Genetic Algorithm for Solving Simple Mathematical Equality Problem." arXiv preprint arXiv: 1308.4675 (2013).

External links

[1] Ant colony optimization algorithms- Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Ant_colony_optimization)
[2] Swarm intelligence- Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Swarm_intelligence)
[3] Swarm intelligence- Marco Dorigo and Mauro Birattari (2007), Scholarpedia, 2(9):1462 (http://www.scholarpedia.org/article/Swarm_intelligence)
[4] Genetic algorithm - Wikipedia, the free encyclopedia (http://en.wikipedia.org/wiki/Genetic_algorithm)

Author Profile

Keisam Thoiba Meetei received the Bachelor degree in Computer Science and Engineering in 2011 from Shiv Shankar Institute of Engineering and Technology (SSIET), Punjab. He is currently pursuing the Master degree in Computer Science and Engineering from Galgotias University, Greater Noida, UP. His area of Interest is Artificial Intelligence