Introduction to Swarm Intelligence - COS314

1 Overview

We have studied one area of evolutionary computation, namely, evolutionary algorithms. Swarm Intelligence is another area we will study. Swarm Intelligence (SI) is a field of research that focuses on the study of the collective behavior of social animals and insects, with the aim of developing intelligent algorithms for solving complex problems. The term "swarm" refers to a group of agents that interact with each other and their environment, using simple rules to achieve complex goals.

The concept of swarm intelligence is inspired by the behavior of social animals, such as ants, bees, and termites. These animals work together to achieve common goals, such as finding food, building nests, or defending their colony. They do this by following simple rules of behavior, such as pheromone trails, communication, and cooperation.

Swarm intelligence algorithms are designed to mimic the behavior of these social animals, by using simple rules to guide the behavior of a group of agents. These algorithms are often used to solve optimization problems, such as finding the shortest path between two points, or optimizing a complex function.

There are several types of swarm intelligence algorithms, including Ant Colony Optimization, Particle Swarm Optimization, and Artificial Bee Colony Optimization. These algorithms have been applied to a wide range of applications, including data clustering, image processing, and network routing.ial ant scenario which ACO employs, based on the real world scenario, is described next.

1.1 Real World Ants

One of the key benefits of swarm intelligence algorithms is their ability to adapt to changing environments. Unlike traditional optimization algorithms, which require a fixed problem definition, swarm intelligence algorithms can adjust their behavior based on the changing conditions of the problem. This makes them particularly useful for solving dynamic problems, such as those encountered in real-world applications.

Swarm Intelligence algorithms can solve complex problems using simple rules and adapt to changing environments. Thus far we have studied three Evolutionary Algorithms, namely, a Genetic Algorithm, Genetic Programming and Grammatical Evolution. In this section of the module we will study Ant Colony Optimization (ACO) ,Particle Swarm Optimization(PSO) and the Artificial Bee Colony Optimsation (ABC).

2 Ant Colony Optimization

Ant Colony Optimization is a swarm intelligence algorithm. Like evolutionary algorithms, ACO is also a metaheuristics. It mimics the behaviour of ants to solve optimization problems [1]. Like evolutionary algorithms, ACO, is a population-based method and hence a multipoint search. The population in this case is the colony of ants. The following section describes the real world

ant scenario on which ACO is based. The artific Ants find their way to a food source from the nest, collect the food source and return to the nest. In this process the ants aim to minimize the distance travelled to obtain the food source and return. In going to and from a food source the ants deposit pheromone on the ground. The purpose of this is for other ants to find the trail to the food source and back. In this way pheromone trails are created and the stronger the pheromone on the trail, the more likely other ants are to follow the trail. Hence, paths that have more pheromone attract more ants. Over time the pheromone signal on a trail gets weaker if not reinforced by more ants following the trail until eventually the pheromone evaporates.

2.1 Artificial Ants

ACO mimics the process followed by ants to collect food to solve a problem. At each point ants in the real world decide on which path to follow to lead to food. In ACO the artificial ants decide on which solution component to choose in the process of determining a solution to the problem. The set C represents the set of solution components, and each C_{ij} represents a solution component. Each trail leading to a solution has a pheromone parameter T_{ij} associated with it. The value of the pheromone parameter is denoted by τ_{ij} . ACO finds a solution by ants traversing a construction graph. The vertices or edges of the graph represent solution components. Each ant begins at a different node in the graph and incrementally builds a partial solution by moving from one vertex to the next. Once the ants have completed constructing a solution the pheromone is updated at each vertex or edge. Each edge has a pheromone value and heuristic value associated with it. The ACO algorithm from [1] is depicted in Algorithm 1. The algorithm begins by setting parameters and pheromone values. Parameters are

Algorithm 1 ACO Metaheuristic Algorithm

```
1: Set parameters and initialize pheromone values
2: while termination criterion is not met do
3:     Construction ant solutions
4:     Perform local search
5:     Update pheromones
6: end while
```

Figure 1: ACO Algorithm

associated with probabilistic rules for selecting solution components in creating solutions and rules for updating pheromone. These differ depending on the ACO algorithm. The number of ants is also a parameter. The ACO process is an iterative process which ter- minates when the termination criterion is met. Two commonly used termination criterion are the number of iterations and CPU time. The process begins by each ant constructing a solution. Each

ant begins at a randomly selected vertex. Each partial solution is initially empty and as the ants select components, these are added to the solutions.The ants continue selecting solution components until the solution is complete.The solution component is selected using a probability rule. The probability rule used is dependent on the ACO algorithm. An example of a probability rule is depicted in equation 1.

$$p(c_{ij}|s^p) = \frac{\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} * \eta_{il}^{\beta}}, \forall c_{il} \in N(s^p) \quad (1)$$

Figure 2: Probability Rule

C_{ij} represents the solution component. τ_{ij} represents the pheromone value associated with the solution component and η_{ij} the heuristic value. N (s^p) is the set of feasible solution component neighbours and s^p the partial solution. α and β are parameter values of the algorithm. The most appropriate values are problem dependent. An optional step is applying local search to these solutions. The step in the algorithm is also referred to as daemon actions. These actions are usually general actions, not specific to the actions achieved by the individual ants, applied to the complete solutions. The final step involves updating the pheromone values. The aim is to increase the pheromone values in cases where the solution quality is good and decrease pheromone on trails where the solution quality is not good. This process begins by remove pheromone, according to a pheromone evaporation rate, from all trails. This is equivalent to the pheromone evaporation process with real ant trails. The rule used to update the pheromone depends on the ACO algorithm used. Given the set of solutions **S** produced by the ants, a pheromone update rule commonly used is:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \rho \sum_{s|c_{ij}} F(s) \quad (2)$$

Figure 3: Probability Rule

τ_{ij} is the pheromone value to be updated. ρ represents the pheromone evaporation rate and is in the range (0,1]. F (S) is the fitness function and usually is the objective value of the problem being solved.

2.1.1 ACO Parameters

The algorithm involves constructing a graph representation of the problem and simulating the behavior of ants on this graph to find the optimal solution. Here are some of the key parameters associated with ACO:

1. **Number of Ants:** The number of ants determines the size of the ant colony used in the algorithm. A larger number of ants can lead to a more

diverse search of the solution space, but may also increase the computational cost of the algorithm.

2. **Pheromone Evaporation Rate:** The pheromone evaporation rate controls the rate at which the pheromone trails on the graph evaporate over time. A higher evaporation rate means that the pheromone trails dissipate more quickly, while a lower rate allows the pheromone to accumulate over time.

3. **Pheromone Intensity:** The pheromone intensity controls the strength of the pheromone trails left by the ants on the graph. A higher intensity means that the pheromone trails have a stronger effect on the decision-making process of the ants.

4. **Heuristic Information:** Heuristic information is used by the ants to guide their search towards promising solutions. This information can be based on domain knowledge, such as the distance between nodes in a graph, or on previous experience of the algorithm.

5. **Ant Decision Rule:** The ant decision rule determines how the ants choose their next move on the graph. This rule can be based on the pheromone trails, the heuristic information, or a combination of both.

6. **Local Search Strategy:** ACO can be combined with local search strategies to improve the quality of the solutions. Local search strategies involve making small changes to the current solution to explore nearby regions of the solution space.

7. **Termination Criteria:** Termination criteria determine when the algorithm should stop searching for a better solution. This can be based on a maximum number of iterations, a time limit, or a minimum improvement in the best solution found.

These parameters can be adjusted to optimize the performance of the ACO algorithm for a particular problem.

2.2 Variations of ACO

ACO algorithms generally differ the probability rules and pheromone update rules used. ACO has several variations, each of which modifies the basic ACO algorithm in different ways to improve its performance or adapt to specific problem domains.

1. **Max-Min Ant System (MMAS):** MMAS is a variation of ACO that uses a dynamic update rule for pheromone trail intensities. MMAS limits the amount of pheromone that can be deposited on each trail, which prevents premature convergence and ensures exploration of the search space.

2. **Ant System (AS):** AS is the original version of ACO, which uses a simple pheromone update rule and a static probability function for ant movement. AS is suitable for problems with a small search space and a small number of parameters.

3. **Rank-Based Ant System (RAS):** RAS is a variation of ACO that uses a rank-based selection strategy for ant movement. RAS ranks the potential solutions based on their fitness and selects the best ones for the next iteration, which ensures convergence to a high-quality solution.

4. **Ant Colony System (ACS):** ACS is a variation of ACO that uses multiple ant colonies to search for solutions. ACS introduces a global pheromone update rule that combines the pheromone levels of all ants in the system, which facilitates information sharing and improves the convergence rate.

5. **Elitist Ant System (EAS):** EAS is a variation of ACO that uses a selective pheromone update rule that favors the best ants in each iteration. EAS gives more importance to the best solutions found so far and prevents the loss of good solutions due to pheromone evaporation.

Overall, the variations of ACO are designed to enhance the performance of the algorithm in different problem domains and to adapt to different search space characteristics. The selection of the appropriate variation depends on the nature of the problem and the available resources.

2.3 Applications of ACO

Ant Colony Optimization (ACO) has been successfully applied to a wide range of optimization problems in various application domains. Some of the popular application domains of ACO are:

1. **Routing and Transportation:** ACO has been used to optimize routing and transportation problems, such as vehicle routing, supply chain management, and logistics planning. ACO algorithms can find the shortest path between multiple points and allocate resources efficiently.

2. **Telecommunications:** ACO has been applied to the optimization of telecommunication networks, such as routing in wireless sensor networks, resource allocation in mobile networks, and scheduling in satellite communication. ACO can help to reduce network congestion, improve signal quality, and minimize the cost of communication.

3. **Manufacturing and Production:** ACO has been used to optimize production and manufacturing processes, such as scheduling, layout design, and quality control. ACO can help to minimize production time, reduce waste, and improve product quality.

4. **Bioinformatics:** ACO has been applied to bioinformatics problems, such as protein folding, sequence alignment, and gene expression analysis. ACO algorithms can help to predict protein structure and function, identify genetic mutations, and classify biological data.

5. **Financial Planning:** ACO has been used to optimize financial planning problems, such as portfolio optimization, asset allocation, and risk management. ACO algorithms can help to minimize risk, maximize returns, and balance the portfolio.

6. **Energy Systems:** ACO has been applied to energy optimization problems, such as power grid management, renewable energy allocation, and energy-efficient building design. ACO algorithms can help to reduce energy consumption, improve energy efficiency, and lower carbon emissions.

Overall, ACO has proven to be a versatile optimization algorithm that can be applied to a wide range of problems in various domains. Its ability to find near-optimal solutions and adapt to changing environments makes it a popular choice for many real-world optimization problems.

3 Particle Swarm Optimization

Particle swarm optimization (PSO) is based on the behaviour of swarms or flocks, e.g. school of fish and flocks of birds [2]. Like evolutionary algorithms PSO performs optimization and is a population based method. PSO is a meta-heuristic that iteratively improves a population of candidate solutions. The candidate solutions are particles. Each particle has a location and velocity. The particles move through the search space in in the direction of the optimal particles at that particular point in the search. The movement is determined by a function of their position and velocity. The movement is based on both local and global best positions at the particular point in the search. Mathematical formulae are used to move the particles. A fitness value is also associated with each particle. A problem dependent fitness function is used to calculate the fitness value.

Algorithm 1 PSO Algorithm

```
1: Initialize all x_i, v_i and pbest_i values
2: while termination criterion is not met do
3:     for i ← 1, N do
4:         Calculate F(x_i)
5:         if F(x_i) < F(pbest_i) then
6:             pbest_i = x_i
7:         end if
8:         if F(x_i) < F(g_best) then
9:             g_best = x_i
10:        end if
11:        Update all v_i and x_i values using equation 1 and 2
12:    end for
13: end while
```

Figure 4: PSO Algorithm

The standard PSO algorithm is depicted in Figure 4. For a particle i, x_i represents the position of the particle, textitv$_i$ the velocity of the particle, F(x_i) is the fitness value for the particle and $pbest_i$ the best known position for the

particle, i.e. the local best position. Similarly, *gbest* represents the global best position and N is the number of particles. PSO is a global optimization algorithm, meaning that it is able to find the global optimum, or the best possible solution, of a problem, rather than a local optimum. PSO has been applied successfully to a wide range of optimization problems in various fields, including engineering, finance, and computer science. One of the formulae used to update the velocity of a particle is defined in the following equation:

$$v_i(t+1) = w * v_i(t) + l_1 * r_1[pbest_i(t) - x_i(t)] + l_2 * r_2[g_{best} - x_i(t)] \quad (1)$$

where: w is the inertia weight which determines to what extent the previous velocity $v_i(t)$ contributes to the current velocity $v_i(t+1)$
l_1 and l_2 are learning factors
r_1 and r_2 are randomly generated numbers in the range [0,1]

If the updated velocity values are too high this will result in fast movement of the particles which may prevent the particles from converging to optimal solutions. Hence, a limit VMax is set on the value of the velocity. If the updated velocity $v_i(t + 1)$ exceeds this V_{max} it is set to V_{max}. A larger value of V_{max} promotes exploration of a larger space by the particles while a lower value promotes exploitation of a particular area of the search space. The formula to update the position of the particle is defined in the following equation A topology defines the mode for communication between particles.

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (2)$$

The global topology is the standard topology used by the PSO algorithm. In this topology all the particles communicate with each other. An alternative is the local topology in only a subset of particles can communicate.

3.1 PSO Parameters

As in the case of genetic algorithms various parameter values need to be set for the PSO algorithm. The best values for these parameters is problem dependent. Parameter tuning has to be performed to determine what the best values to use are. Parameters for the PSO algorithm include:

1. **Swarm Size:** The swarm size is the number of particles used in the algorithm. A larger swarm size can lead to a more diverse search of the solution space, but may also increase the computational cost of the algorithm.

2. **Maximum Velocity:** The maximum velocity is a parameter that limits the maximum change in velocity of each particle. This helps to prevent the particles from overshooting the global best position and increases the convergence rate of the algorithm.

3. **Inertia Weight:** The inertia weight is a parameter that balances the exploration and exploitation of the search space. This weight factor controls

the balance between the current velocity and the difference between the current position and the global best position.

4. **Acceleration Coefficients:** The acceleration coefficients determine the influence of the personal best and global best positions on the movement of the particles. These coefficients control the rate at which the particles move towards the best solutions found by themselves and by the swarm as a whole.

5. **Neighborhood Topology:** The neighborhood topology determines how the particles communicate and share information with each other. The topology can be defined as a global, local, or hybrid network.

6. **Local Search Strategy:** PSO can be combined with local search strategies to improve the quality of the solutions. Local search strategies involve making small changes to the current solution to explore nearby regions of the solution space.

7. **Termination Criteria:** Termination criteria determine when the algorithm should stop searching for a better solution. This can be based on a maximum number of iterations, a time limit, or a minimum improvement in the best solution found.

These parameters can be adjusted to optimize the performance of the PSO algorithm for a particular problem. However, finding the optimal values for these parameters can be challenging and may require a trial-and-error approach or the use of optimization techniques.

3.2 Variations of PSO

Variations of PSO Variants of the standard PSO algorithm exist. These variants differ in terms of:

- Initial values of positions and velocities
- Update rules, e.g. update the local and global position only after the swarm is updated.
- Velocity update equation

1. **Constriction Factor PSO:** This variation introduces a constriction factor that limits the maximum change in velocity of each particle. This helps to prevent the particles from overshooting the global best position and increases the convergence rate of the algorithm.

2. **Inertia Weight PSO:** Inertia weight PSO modifies the velocity update equation by adding an inertia weight term that balances the exploration and exploitation of the search space. This weight factor controls the balance between the current velocity and the difference between the current position and the global best position.

3. **Cooperative PSO:** Cooperative PSO introduces multiple sub-swarms that work independently and exchange information periodically. The sub-swarms explore different areas of the search space, and the information exchange helps to improve the overall performance of the algorithm.

4. **Hybrid PSO:** Hybrid PSO combines the PSO algorithm with other optimization algorithms, such as genetic algorithms or simulated annealing. This approach helps to improve the global search capabilities of the PSO algorithm and overcome its limitations.

5. **Multi-objective PSO:** Multi-objective PSO extends the PSO algorithm to solve optimization problems with multiple objectives. This variation uses Pareto optimization to find the set of non-dominated solutions that provide a trade-off between different objectives.

3.3 Applications of PSO

Particle Swarm Optimization (PSO) has been applied to a wide range of problems in various domains. Some of the common application domains of PSO are:

1. **Engineering:** PSO has been used for optimization problems in different fields of engineering such as mechanical, civil, electrical, and chemical engineering. For example, PSO has been used to optimize the design of structures, heat exchangers, and control systems.

2. **Finance:** PSO has been used in finance to optimize investment portfolios, predict stock prices, and credit scoring.

3. **Computer Science:** PSO has been used in computer science for problems such as image processing, clustering, and classification.

4. **Machine Learning:** PSO has been used to optimize the weights and biases of artificial neural networks and improve their performance.

5. **Renewable Energy:** PSO has been used to optimize the placement and sizing of renewable energy systems, such as wind turbines and solar panels.

6. **Robotics:** PSO has been used for trajectory planning, motion control, and path planning of mobile robots.

7. **Health and Medicine:** PSO has been used for problems such as drug discovery, medical image analysis, and disease diagnosis.

The versatility of the algorithm has made it a popular choice for many optimization problems, especially those that are difficult to solve using traditional methods.

References

1] M. Dorigo and K. Socha, "An introduction to ant colony optimization," IRIDIA, Tech. Rep., 2007.
2] X.S. Yang, Nature Inspired Optimization Algorithms. Elsevier, 2014.