Chapter 11
Introduction to Particle Swarm Optimization and Ant Colony Optimization

11.1 Introduction

In this chapter, a brief introduction is given to Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). Optimization is the process to find a best optimal solution for the problem under consideration. Particle Swarm Optimization and Ant Colony Optimization achieve finding an optimal solution for the search problems using the social behavior of the living organisms. Particle swarm optimization is a form of swarm intelligence and Ant colony optimization is a population-based metaheuristic that can be used to find approximate solutions to difficult optimization problems. The chapter gives an overview of basic concepts and fucntional operation of Particle Swarm Optimization and Ant Colony Optimization.

11.2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization technique developed by Dr. Eberhart and Dr. Kennedy in 1995, inspired by social behavior of bird flocking or fish schooling. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms (GA). The system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles.

In past several years, PSO has been successfully applied in many research and application areas. It is demonstrated that PSO gets better results in a faster, cheaper way compared with other methods. Another reason that PSO is attractive is that there are few parameters to adjust. One version, with slight variations, works well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

11.2.1 Background of Particle Swarm Optimization

Particle swarm optimization (PSO) is a form of swarm intelligence and is inspired by bird flocks, fish schooling and swarm of insects. The flock of birds, fish schooling and swarm of insects is as shown in Fig. 11.1.

Consider Fig. 11.1 and imagine a swarm of insects or a school of fish. If one sees a desirable path to go (e.g., for food, protection, etc.) the rest of the swarm will be able to follow quickly even if they are on the opposite side of the swarm. On the other hand, in order to facilitate felicitous exploration of the search space, typically one wants each particle to have a certain level of "craziness" or randomness in their movement, so that the movement of the swarm has a certain explorative capability: the swarm should be influenced by the rest of the swarm but also should independently explore to a certain extent.

This is performed by particles in multidimensional space that have a position and a velocity. These particles are flying through hyperspace (i.e., $\Re^n$) and have two essential reasoning capabilities: their memory of their own best position and knowledge of the swarm's best, "best" simply meaning the position with the smallest objective value. Members of a swarm communicate good positions to each other and

Fig. 11.1 Social behavior

adjust their own position and velocity based on these good positions. There are two main ways this is done:

- a global best that is known to all and immediately updated when a new best position is found by any particle in the swarm
- "neighborhood" bests where each particle only immediately communicates with a subset of the swarm about best positions

Each particle keeps track of its coordinates in the problem space which are associated with the best solution (fitness) it has achieved so far. (The fitness value is also stored.) This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the neighbors of the particle. This location is called *lbest*. when a particle takes all the population as its topological neighbors, the best value is a global best and is called *gbest*.

The particle swarm optimization concept consists of, at each time step, changing the velocity of (accelerating) each particle toward its p*best* and *lbest* locations (local version of PSO). Acceleration is weighted by a random term, with separate random numbers being generated for acceleration toward *pbest* and *lbest* locations.

11.2.2 Operation of Particle Swarm Optimization

Consider Swarm of particles is flying through the parameter space and searching for optimum. Each particle is characterized by,

$$\text{Position vector} \ldots\ldots \boldsymbol{x}_i(t)$$
$$\text{Velocity vector} \ldots\ldots \boldsymbol{v}_i(t)$$

as shown in Fig. 11.2.

During the process, each particle will have its individual knowledge *pbest*, i.e., its own best-so-far in the position and social knowledge *gbest* i.e., *pbest* of its best neighbor as shown in Fig. 11.3.

Performing the velocity update, using the formula given below,

$$v_i(t+1) = \alpha\, v_i + c_1 \times rand \times (pbest(t)\text{-}x_i(t)) + c_2 \times rand \times (gbest(t)\text{-}x_i(t)) \tag{11.1}$$

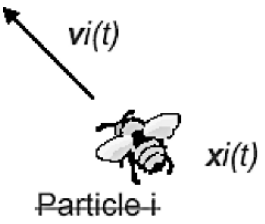

Fig. 11.2 A particle with position vector and velocity vector

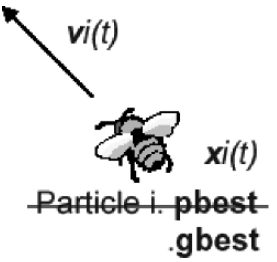

Fig. 11.3 Particle with *pbest* and *gbest*

where α is the inertia weight that controls the exploration and exploitation of the search space. c_1 and c_2, the cognition and social components respectively are the acceleration constants which changes the velocity of a particle towards the *pbest* and *gbest*. *rand* is a random number between 0 and 1. Usually c_1 and c_2 values are set to 2. The velocity update is based on the parameters as shown in Fig. 11.4.

Now, performing the position update,

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{11.2}$$

The position update process is as shown in Fig. 11.5

The above process discussed is repeated for each and evry particle considered in the computation and the best optimal solution is obtained.

PSO utilizes several searching points like genetic algorithm (GA) and the searching points gradually get close to the optimal point using their pbests and the gbest. The first term of RHS of (11.1) is corresponding to diversification in the search procedure. The second and third terms of that are corresponding to intensification in the search procedure. Namely, the method has a well balanced mechanism to utilize diversification and intensification in the search procedure efficiently. The original PSO can be applied to the only continuous problem. However, the method can be expanded to the discrete problem using discrete number position and its velocity easily.

The above feature can be explained as follows. The RHS of (11.1) consists of three terms. The first term is the previous velocity of the agent. The second and third

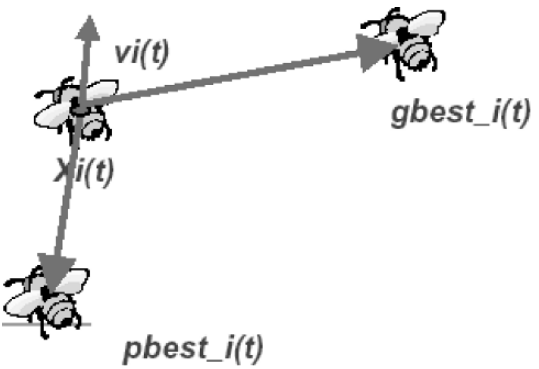

Fig. 11.4 Parameters for velocity update

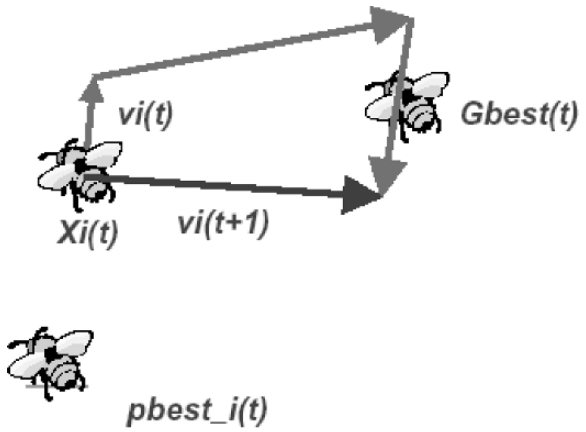

Fig. 11.5 Position update using $x_i(t)$ and $v_i(t+1)$

terms are utilized to change the velocity of the agent. Without the second and third terms, the agent will keep on "flying" in the same direction until it hits the boundary. Namely, it tries to explore new areas and, therefore, the first term is corresponding to diversification in the search procedure. On the other hand, without the first term, the velocity of the "flying" agent is only determined by using its current position and its best positions in history. Namely, the agents will try to converge to the their pbests and/or gbest and, therefore, the terms are corresponding to intensification in the search procedure.

11.2.3 Basic Flow of Particle Swarm Optimization

The basic operation of PSO is given by,

Step 1: Initialize the *swarm* from the solution space
Step 2: Evaluate *fitness* of individual particles
Step 3: Modify *gbest, pbest* and *velocity*
Step 4: Move each *particle* to a new *position*
Step 5: Goto step 2, and repeat until convergence or stopping condition is satisfied

The pseudo code of the procedure is as follows

```
For each particle
    Initialize particle
END
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 (11.1)
Update particle position according equation (11.2)
End

While maximum iterations or minimum error criteria is not attained

Particles' velocities on each dimension are clamped to a maximum velocity V_{max}. If the sum of accelerations would cause the velocity on that dimension to exceed V_{max}, which is a parameter specified by the user. Then the velocity on that dimension is limited to V_{max}.

The basic flowchart of Particle Swarm optimization is as shown in Fig. 11.6. In that Repository refers to the memory location of each particle.

11.2.4 Comparison Between PSO and GA

The strength of GAs is in the parallel nature of their search. A GA implements a powerful form of hill climbing that preserves multiple solutions, eradicates unpromising solutions, and provides reasonable solutions. Through genetic operators, even weak solutions may continue to be part of the makeup of future candidate solutions. The genetic operators used are central to the success of the search. All GAs require some form of recombination, as this allows the creation of new solutions that have, by virtue of their parent's success, a higher probability of exhibiting a good performance. In practice, crossover is the principal genetic operator, whereas

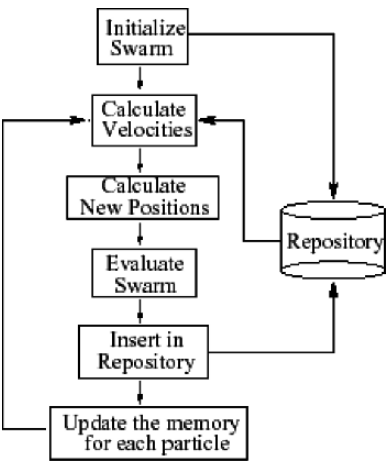

Fig. 11.6 Basic flowchart of PSO

mutation is used much less frequently. Crossover attempts to preserve the beneficial aspects of candidate solutions and to eliminate undesirable components, while the random nature of mutation is probably more likely to degrade a strong candidate solution than to improve it. Another source of the algorithm's power is the implicit parallelism inherent in the evolutionary metaphor. By restricting the reproduction of weak candidates, GAs eliminate not only that solution but also all of its descendants. This tends to make the algorithm likely to converge towards high quality solutions within a few generations.

Most of evolutionary techniques have the following procedure:

1. Random generation of an initial population
2. Reckoning of a fitness value for each subject. It will directly depend on the distance to the optimum.
3. Reproduction of the population based on fitness values.
4. If requirements are met, then stop. Otherwise go back to 2.

From the procedure, one can learn that PSO shares many common points with GA. Both algorithms start with a group of a randomly generated population; both have fitness values to evaluate the population. Both update the population and search for the optimum with random techniques. Both systems do not guarantee success. However, PSO does not have genetic operators like crossover and mutation. Particles update themselves with the internal velocity. They also have memory, which is important to the algorithm. Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gbest (or lbest) gives out the information to others. It is a one -way information sharing mechanism. The evolution only looks for the best solution. Compared with GA, all the particles tend to converge to the best solution quickly even in the local version in most cases.

Particle Swarm Optimization shares many similarities with evolutionary computation (EC) techniques in general and GAs in particular. All three techniques begin with a group of a randomly generated population, all utilize a fitness value to evaluate the population. They all update the population and search for the optimum with random techniques. A large inertia weight facilitates global exploration (search in new areas), while a small one tends to assist local exploration. The main difference between the PSO approach compared to EC and GA, is that PSO does not have genetic operators such as crossover and mutation. Particles update themselves with the internal velocity; they also have a memory that is important to the algorithm. Compared with EC algorithms (such as evolutionary programming, evolutionary strategy and genetic programming), the information sharing mechanism in PSO is significantly different. In EC approaches, chromosomes share information with each other, thus the whole population moves like one group towards an optimal area. In PSO, only the "best" particle gives out the information to others. It is a one-way information sharing mechanism; the evolution only looks for the best solution. Compared with ECs, all the particles tend to converge to the best solution quickly even in the local version in most cases. Compared to GAs, the advantages of PSO are that PSO is easy to implement and there are few parameters to adjust.

11.2.5 Applications of PSO

PSO has been successfully applied in many areas: function optimization, artificial neural network training, fuzzy system control, and other areas where GA can be applied. The various application areas of Particle Swarm Optimization include:

- Power Systems operations and control
- NP-Hard combinatorial problems
- Job Scheduling problems
- Vehicle Routing Problems
- Mobile Networking
- Modeling optimized parameters
- Batch process scheduling
- Multi-objective optimization problems
- Image processing and Pattern recognition problems

and so on. Currently, several researchers are being carried out in the area of particle swarm optimization and hence the application area also increases tremendously.

11.3 Ant Colony Optimization

Ant Colony Optimization (ACO) is a population-based, general search technique for the solution of difficult combinatorial problems, which is inspired by the pheromone trail laying behavior of real ant colonies. In ACO, a set of software agents called artificial ants search for good solutions to a given optimization problem. To apply ACO, the optimization problem is transformed into the problem of finding the best path on a weighted graph. The artificial ants (hereafter ants) incrementally build solutions by moving on the graph. The solution construction process is stochastic and is biased by a pheromone model, that is, a set of parameters associated with graph components (either nodes or edges) whose values are modified at runtime by the ants.

The first member of ACO class of algorithms, called Ant System, was initially proposed by Colorni, Dorigo and Maniezzo. The main underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result. The collective behavior emerging from the interaction of the different search threads has proved effective in solving combinatorial optimization (CO) problems.

11.3.1 Biological Inspiration

In the 40s and 50s of the 20th century, the French entomologist Pierre-Paul Grass observed that some species of termites react to what he called “significant stimuli”.

He observed that the effects of these reactions can act as new significant stimuli for both the insect that produced them and for the other insects in the colony. Grass used the term stigmergy to describe this particular type of communication in which the "workers are stimulated by the performance they have achieved".

The two main characteristics of stigmergy that differentiate it from other forms of communication are the following.

- Stigmergy is an indirect, non-symbolic form of communication mediated by the environment: insects exchange information by modifying their environment; and
- Stigmergic information is local: it can only be accessed by those insects that visit the locus in which it was released (or its immediate neighborhood).

Stigmergy is an indirect and asynchronous form of communication in which the insects manipulate the environment to transport information to the other insects, which then respond to the change. The insects therefore do not have to be at the same place at the same time as the others to communicate with them. In many ant species colonies, stigmergy refers to the deposition of pheromone by ants while they are moving. Other ants can then smell the deposited pheromone and have a natural tendency to follow the laid trail. This constitutes an asynchronous and indirect communication scheme, where one ant communicates with other ants wherever they are, and it is how positive feedback is created. A little pheromone on a path might lead other ants to follow the same path, depositing even more pheromone, which can lead to a positive feedback effect, if the selected path is good (leading to food) thus recruiting even more ants to follow the path.

The main elements of a biological stigmergic system are shown in Fig. 11.7:

- The insect as the acting individual.
- The pheromone as an information carrier, used to create a dissipation field.
- The environment as a display and distribution mechanism for information.

Examples of stigmergy can be observed in colonies of ants. In many ant species, ants walking to and from a food source deposit on the ground a substance called pheromone. Other ants perceive the presence of pheromone and tend to follow paths where pheromone concentration is higher. Through this mechanism, ants are able to transport food to their nest in a remarkably effective way. The basic behavior of the ants is discussed as follows:

Real ants are capable of finding shortest path from a food source to the nest without using visual cues. Also, they are capable of adapting to changes in the environment, for example finding a new shortest path once the old one is no longer

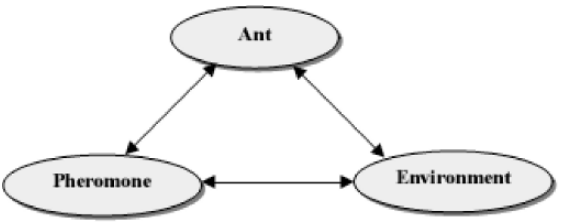

Fig. 11.7 Elements of a stigmeric system

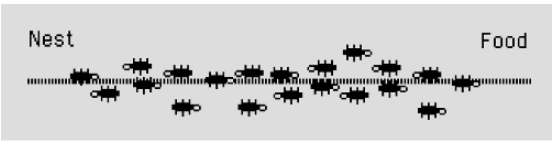

Fig. 11.8 Movement of ant in a straight line

feasible due to a new obstacle. Consider the following Fig. 11.8 in which ants are moving on a straight line, which connects a food source to the nest:

It is well known that the main means used by ants to form and maintain the line is a pheromone trail. Ants deposit a certain amount of pheromone while walking, and each ant probabilistically prefers to follow a direction rich in pheromone rather than a poorer one. This elementary behavior of real ants can be used to explain how they can find the shortest path, which reconnects a broken line after the sudden appearance of an unexpected obstacle, has interrupted the initial path (Fig. 11.9).

In fact, once the obstacle has appeared, those ants, which are just in front of the obstacle, cannot continue to follow the pheromone trail and therefore they have to choose between turning right or left. In this situation we can expect half the ants to choose to turn right and the other half to turn left. The very same situation can be found on the other side of the obstacle (Fig. 11.10).

It is interesting to note that those ants which choose, by chance, the shorter path around the obstacle will more rapidly reconstitute the interrupted pheromone trail compared to those which choose the longer path. Hence, the shorter path will receive a higher amount of pheromone in the time unit and this will in turn cause a higher number of ants to choose the shorter path. Due to this positive feedback (autocatalytic) process, very soon all the ants will choose the shorter path (Fig. 11.11).

The most interesting aspect of this autocatalytic process is that finding the shortest path around the obstacle seems to be an emergent property of the interaction between the obstacle shape and ants distributed behavior: Although all ants move at approximately the same speed and deposit a pheromone trail at approximately the same rate, it is a fact that it takes longer to contour obstacles on their longer side than on their shorter side which makes the pheromone trail accumulate quicker on the shorter side. It is the ants' preference for higher pheromone trail levels, which makes this accumulation still quicker on the shorter path.

Deneubourg et al. thoroughly investigated the pheromone laying and following behavior of ants. In an experiment known as the "double bridge experiment", the nest of a colony of Argentine ants was connected to a food source by two bridges

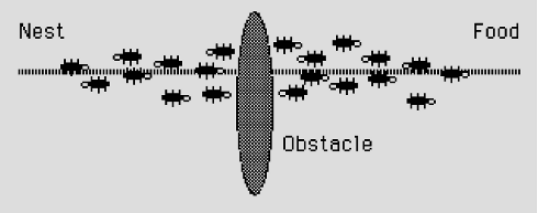

Fig. 11.9 Obstacle on ant paths

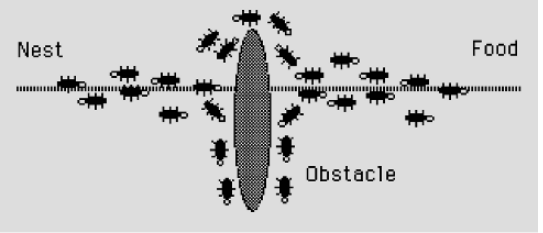

Fig. 11.10 Behavior of ants to obstacle

of equal lengths, as shown in Fig. 11.12. In such a setting, ants start to explore the surroundings of the nest and eventually reach the food source. Along their path between food source and nest, Argentine ants deposit pheromone. Initially, each ant randomly chooses one of the two bridges. However, due to random fluctuations, after some time one of the two bridges presents a higher concentration of pheromone than the other and, therefore, attracts more ants. This brings a further amount of pheromone on that bridge making it more attractive with the result that after some time the whole colony converges toward the use of the same bridge.

This colony-level behavior, based on autocatalysis, that is, on the exploitation of positive feedback, can be used by ants to find the shortest path between a food source and their nest. Goss et al. considered a variant of the double bridge experiment in which one bridge is significantly longer than the other, as shown in Fig. 11.13. In this case, the stochastic fluctuations in the initial choice of a bridge are much reduced and a second mechanism plays an important role: the ants choosing by chance the short bridge are the first to reach the nest. The short bridge receives, therefore, pheromone earlier than the long one and this fact increases the probability that further ants select it rather than the long one. Goss et al. developed a model of the observed behavior: assuming that at a given moment in time m_1 ants have used the first bridge and m_2 the second one, the probability p_1 for an ant to choose the first bridge is:

$$p_1 = \frac{(m_1 + k)^h}{(m_1 + k)^h + (m_2 + k)^h}$$

where parameters k and h are to be fitted to the experimental data—obviously $p_2 = 1\text{-}p_1$.

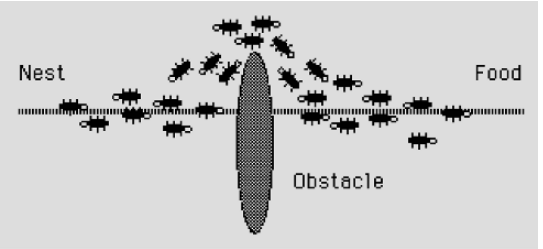

Fig. 11.11 Ants choosing shorter path

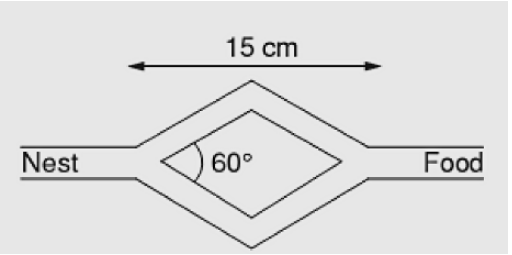

Fig. 11.12 Double bridge experiment—Bridges of equal length

11.3.2 Similarities and Differences Between Real Ants and Artificial Ants

Most of the ideas of ACO stem from real ants. In particular, the use of a colony cooperating individuals, an artificial pheromone trail for local stigmergetic communication a sequence of local moves to find shortest paths and a stochastic decision policy using local information. Researchers have used the most ideas from real ants behavior in order to build Ant System (AS). There exist some differences and similarities between real and artificial ants which could be stated as follows:

11.3.2.1 Similarities

- Colony of cooperating individuals–Both real ant colonies and ant algorithms are composed of a population, or colony of independent individual agents. They globally cooperate in order to find a good solution to the task under consideration. Although the complexity of each artificial ant is such that it can build a feasible solution (as a real ant can find somehow a path between the nest and the food), high quality solutions are the result of the cooperation among the individuals of the whole colony.
- Pheromone trail and stigmergy–Like real ants, artificial ants change some aspects of their environment while walking. Real ants deposit a chemical substance called pheromone on the visited state. Artificial ants will change some numerical information of the problem state, locally stored, when that state is visited. Based on analogy, these information and changes could be called an artificial pheromone trail. Ant system algorithms assume that a local pheromone trail is the single

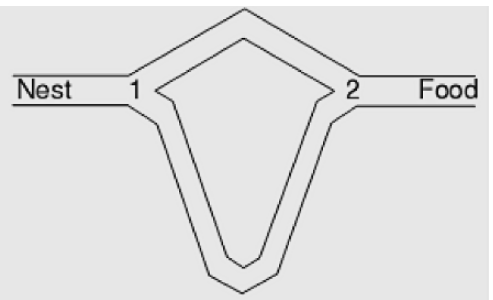

Fig. 11.13 Double bridge experiment—Bridges of varying length

way of communication among artificial ants. AS algorithms include artificial pheromone evaporation in form of reduction in the artificial pheromone trail over time as in nature. Pheromone evaporation in nature and in AS algorithms are important because it will allow ant colony to slowly forget the history and direct the searching process in new directions. Artificial pheromone evaporation could be helpful to move the searching process toward new regions and to avoid stacking in local extremes.

- Local moves and the shortest path searching–Despite real ants are walking through adjacent states and artificial ants are jumping from one to another adjacent state of the considered problem, both walking and jumping have the same purpose, which is finding the shortest path between the origin and the destination.
- Transition policy–Both real ants and artificial ones will build solutions by applying decision making procedures to move through adjacent states. Decision making procedures could be based on some probabilistic rules or probabilities could be calculated based on approximate reasoning rules. In both cases, the transition policy will use local information that should be local in the space and time sense. The transition policy is a function of local state information represented by problem specifications (this could be equivalent to the terrain's structure that surrounds the real ants) and the local modification of the environment (existing pheromone trails) introduced by ants that have visited the same location.
- Deposited amount of pheromone–Amount of pheromone that an artificial ant will deposit is mostly a function of the quality of the discovered solution. In nature, some ants behave in a similar way, the deposited amount of pheromone is highly dependent on the quality of the discovered food source.

11.3.2.2 Differences

The differences are as follows:

- Artificial ants live in a discrete world. All their moves are jumps from one discrete state to another adjacent one.
- Artificial ants have memory, they could remember states that have been visited already (tabu lists in the model).
- Pheromone deposit methodology is significantly different between real and artificial ants. Timing in pheromone laying is problem dependent and often does not have similarities with the real ants pheromone deposit methodology.
- To improve overall performance, AS algorithms could be enriched with some additional capabilities that cannot be found in real ant colonies. Most AS contains some local optimization techniques to improve solutions developed by ants.

11.3.3 Characteristics of Ant Colony Optimization

The characteristics of ant colony optimization are as follows:

- **Natural algorithm** since it is based on the behavior of real ants in establishing paths from their colony to source of food and back.

- **Parallel and distributed** since it concerns a population of agents moving simultaneously, independently and without a supervisor.
- **Cooperative** since each agent chooses a path on the basis of the information, pheromone trails laid by the other agents, which have previously selected the same path. This cooperative behavior is also autocatalytic, i.e., it provides a positive feedback, since the probability of choosing a path increases with the number of agents that previously chose that path.
- **Versatile** that it can be applied to similar versions of the same problem; for example, there is a straightforward extension from the traveling salesman problem (TSP) to the asymmetric traveling salesman problem (ATSP).
- **Robust** that it can be applied with minimal changes to other combinatorial optimization problems such as quadratic assignment problem (QAP) and the job-shop scheduling problem (JSP).

11.3.4 Ant Colony Optimization Algorithms

The model proposed by Deneubourg and co-workers for explaining the foraging behavior of ants was the main source of inspiration for the development of ant colony optimization. In ACO, a number of artificial ants build solutions to the considered optimization problem at hand and exchange information on the quality of these solutions via a communication scheme that is reminiscent of the one adopted by real ants.

Different ant colony optimization algorithms have been proposed. The original ant colony optimization algorithm is known as Ant System and was proposed in the early 90s. Since then, a number of other ACO algorithms were introduced. Table 11.1 gives a list of successful variants of Ant Colony Optimization Algorithms. Figure 11.14 gives a narrow overview on Ant Colony Optimization Algorithms.

ACO is a class of algorithms, whose first member, called Ant System, was initially proposed by Colorni, Dorigo and Maniezzo. The main underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several

Table 11.1 Development of various Ant Colony Optimization Algorithms

ACO algorithm	Authors	Year
Ant System	Dorigo, Maniezzo & Colomi	1991
Elitist AS	Dorigo	1992
Ant-Q	Gambardella & Dorigo	1995
Ant Colony System	Dorigo & Gambardella	1996
$\mathcal{MM}$AS	Stützle & Hoos	1996
Rank-based AS	Bullnheimer, Hartl & Strauss	1997
ANTS	Maniezzo	1998
Best-Worst AS	Cordón, et al.	2000
Hyper-cube ACO	Blum, Roli, Dorigo	2001

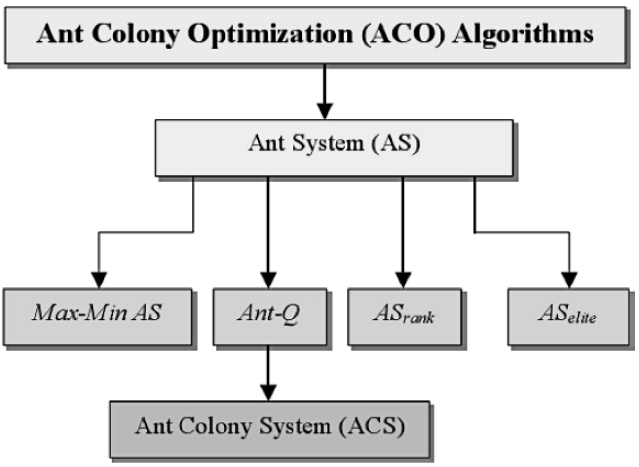

Fig. 11.14 Overview of ACO algorithms

constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result. The collective behavior emerging from the interaction of the different search threads has proved effective in solving combinatorial optimization (CO) problems.

The notation is used as follows: A combinatorial optimization problem is a problem defined over a set $C = c_1, \ldots, c_n$ of basic components. A subset S of components represents a solution of the problem; $F \subseteq 2^C$ is the subset of feasible solutions, thus a solution S is feasible if and only if S Œ F. A cost function z is defined over the solution domain, $z : 2^C \rightarrow R$, the objective being to find a minimum cost feasible solution S^*, i.e., to find $S^* : S^* \in F$ and $z(S^*) \leq z(S), \forall S \in F$.

Given this, the functioning of an ACO algorithm can be summarized as follows. A set of computational concurrent and asynchronous agents (a colony of ants) moves through states of the problem corresponding to partial solutions of the problem to solve. They move by applying a stochastic local decision policy based on two parameters, called trails and attractiveness. By moving, each ant incrementally constructs a solution to the problem. When an ant completes a solution, or during the construction phase, the ant evaluates the solution and modifies the trail value on the components used in its solution. This pheromone information will direct the search of the future ants.

Furthermore, an ACO algorithm includes two more mechanisms: trail evaporation and, optionally, daemon actions. Trail evaporation decreases all trail values over time, in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants, such as the invocation of a local optimization procedure, or the update of global information to be used to decide whether to bias the search process from a non-local perspective.

More specifically, an ant is a simple computational agent, which iteratively constructs a solution for the instance to solve. Partial problem solutions are seen as

states. At the core of the ACO algorithm lies a loop, where at each iteration, each ant moves (performs a step) from a state ι to another one ψ, corresponding to a more complete partial solution. That is, at each step σ, each ant k computes a set $A_k{}^{\sigma}(t)$ of feasible expansions to its current state, and moves to one of these in probability. The probability distribution is specified as follows. For ant k, the probability $p_{\tau\psi}{}^{k}$ of moving from state ι to state ψ depends on the combination of two values:

- the attractiveness $n_{\iota\psi}$ of the move, as computed by some heuristic indicating the a priori desirability of that move;
- the trail level $\tau_{\iota\psi}$ of the move, indicating how proficient it has been in the past to make that particular move: it represents therefore an a posteriori indication of the desirability of that move.

Trails are updated usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively. The general framework just presented has been specified in different ways by the authors working on the ACO approach. A brief introduction is given for Ant System (AS) and Ant Colony System (ACS) to traveling salesman problem.

11.3.4.1 Ant System

Ant System is the first ACO algorithm proposed in the literature. Ant System applied to traveling Sales Man problem is discussed here. Its main characteristic is that, at each iteration, the pheromone values are updated by all the m ants that have built a solution in the iteration itself. The pheromone τ_{ij} associated with the edge joining cities i and j, is updated as follows:

$$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}, \tag{11.3}$$

where ρ is the evaporation rate, m is the number of ants, and $\Delta\tau_{ij}^{k}$ is the quantity of pheromone laid on edge (i, j) by ant k:

$$\Delta\tau_{ij}^{k} = \begin{cases} Q/L_k & \text{if ant } k \text{ used edge } (i,\ j) \text{ in its tour,} \\ 0 & \text{otherwise,} \end{cases} \tag{11.4}$$

Where Q is a constant, and L_k is the length of the tour constructed by ant k.

In the construction of a solution, ants select the following city to be visited through a stochastic mechanism. When ant k is in city i and has so far constructed the partial solution s^p probability of going to city j is given by:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{ij} \in \mathrm{N}(s^p)} \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}} & \text{if } c_{ij} \in \mathrm{N}(s^p), \\ 0 & \text{otherwise,} \end{cases} \tag{11.5}$$

where $\mathrm{N}(s^p)$ is the set of feasible components; that is, edges (i, l) where l is the city not yet visited by the ant k. The parameters α and β control the relative importance of the pheromone versus the heuristic information η_{ij}, which is given by,

$$\eta_{ij} = \frac{1}{d_{ij}}, \tag{11.6}$$

where d_{ij} is the distance between cities i and j.

11.3.4.2 Ant Colony System

The most interesting contribution of ACS is the introduction of a local pheromone update in addition to the pheromone update performed at the end of the construction process (called offline pheromone update). The local pheromone update is performed by all the ants after each construction step. Each ant applies it only to the last edge traversed:

$$\tau_{ij} = (1 - \varphi) \cdot \tau_{ij} + \varphi \cdot \tau_0, \tag{11.7}$$

where $\varphi \in (0, 1)$ is the pheromone decay coefficient, and τ_0 is the initial value of the pheromone.

The main goal of the local update is to diversify the search performed by subsequent ants during an iteration: by decreasing the pheromone concentration on the traversed edges, ants encourage subsequent ants to choose other edges and, hence, to produce different solutions. This makes it less likely that several ants produce identical solutions during one iteration. The offline pheromone update is applied at the end of each iteration by only one ant, which can be either the iteration-best or the best-so-far. However, the update formula is:

$$\tau_{ij} \leftarrow \begin{cases} (1 - \rho) \cdot \tau_{ij} + \rho \cdot \Delta\tau_{ij} & \text{if } (i, j) \text{ belongs to best tour,} \\ \tau_{ij} & \text{otherwise.} \end{cases} \tag{11.8}$$

where $\Delta\tau_{ij} = 1/\mathrm{L_{best}}$, where $\mathrm{L_{best}}$ can be either $\mathrm{L_{ib}}$ or $\mathrm{L_{bs}}$. $\mathrm{L_{best}}$ is the length of the tour of the best ant. This may be (subject to the algorithm designer decision) either the best tour found in the current iteration—*iteration-best*, $\mathrm{L_{ib}}$—or the best solution found since the start of the algorithm—*best-so-far*, $\mathrm{L_{bs}}$—or a combination of both.

Another important difference between ACS and AS is in the decision rule used by the ants during the construction process. In ACS, the so-called *pseudorandom proportional rule* is used: the probability for an ant to move from city i to city j

depends on a random variable q uniformly distributed over [0, 1], and a parameter q_0; if $q \leq q_0$, then $j = \arg\max_{c_{ij} \in N(s^p)} \{\tau_{il} \eta_{il}^{\beta}\}$ otherwise (11.5) is used.

11.3.4.3 Basic Flow of ACO

The basic operational flow in Ant Colony Optimization is as follows:

Step 1: Represent the solution space by a construction graph
Step 2: Initialize ACO parameters
Step 3: Generate random solutions from each ant's random walk
Step 4: Update pheromone intensities
Step 5: Goto Step 3, and repeat until convergence or a stopping condition is satisfied.

The generic ant algorithm is given as shown in Fig. 11.15,

The generalized flowchart for Ant Colony Optimization algorithm is as shown in Fig. 11.16.

The step by step procedure to solve combinatorial optimization problems using ACO in a nutshell is:

- Represent the problem in the form of sets of components and transitions or by means of a weighted graph that is travelled by the ants to build solutions.
- Appropriately define the meaning of the pheromone trails, i.e., the type of decision they bias. This is a crucial step in the implementation of an ACO algorithm. A good definition of the pheromone trails is not a trivial task and it typically requires insight into the problem being solved.
- Appropriately define the heuristic preference to each decision that an ant has to take while constructing a solution, i.e., define the heuristic information associated to each component or transition. Notice that heuristic information is crucial for good performance if local search algorithms are not available or can not be applied.
- If possible, implement an efficient local search algorithm for the problem under consideration, because the results of many ACO applications to NP-hard

Step 1: Initilization
– Initialize the pheromone trail
Step 2: Iteration
– For each Ant Repeat
– Solution construction using the current pheromone trail
– Evaluate the solution constructed
– Update the pheromone trail
– Until stopping criteria

Fig. 11.15 A generic ant algorithm

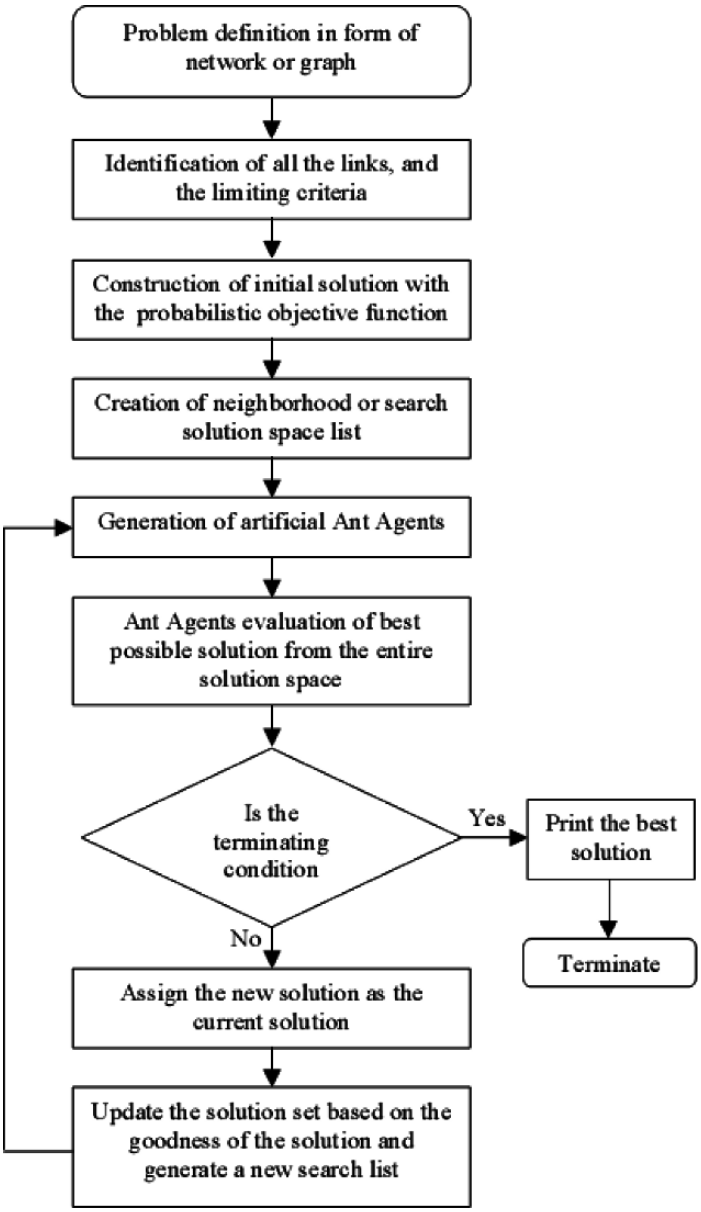

Fig. 11.16 Basic flowchart of Ant Colony Optimization Algorithm

combinatorial optimization problems show that the best performance is achieved when coupling ACO with local optimizers.
- Choose a specific ACO algorithm and apply it to the problem being solved, taking the previous aspects into consideration.
- Tune the parameters of the ACO algorithm. A good starting point for parameter tuning is to use parameter settings that were found to be good when applying the ACO algorithm to similar problems or to a variety of other problems.

It should be clear that the above steps can only give a very rough guide to the implementation of ACO algorithms. In addition, the implementation is often an iterative process, where with some further insight into the problem and the behavior of the algorithm; some initially taken choices need to be revised. Finally, we want to insist on the fact that probably the most important of these steps are the first four, because a poor choice at this stage typically can not be made up with pure parameter fine-tuning.

An ACO algorithm iteratively performs a loop containing two basic procedures, namely:

- A procedure specifying how the ants construct/modify solutions of the problem to be solved;
- A procedure to update the pheromone trails.

The construction/modification of a solution is performed in a probabilistic way. The probability of adding a new item to the current partial solution is given by a function that depends on a problem-dependent heuristic and on the amount of pheromone deposited by ants on the trail in the past. The updates in the pheromone trail are implemented as a function that depends on the rate of pheromone evaporation and on the quality of the produced solution.

11.3.5 Applications of Ant Colony Optimization

There are numerous successful implementations of the ACO meta-heuristic applied to a number of different combinatorial optimization problems. The applications include:

- Traveling Salesman Problem, where a salesman must find the shortest route by which he can visit a given number of cities, each city exactly once.
- Quadratic Assignment Problem, the problem of assigning n facilities to n locations so that the costs of the assignment are minimized.
- Job-Shop Scheduling Problem, where a given set of machines and set of job operations must be assigned to time intervals in such a way that no two jobs are processed at the same time on the same machine and the maximum time of completion of all operations is minimized.

- Vehicle Routing Problem, the objective is to find minimum cost vehicle routes such that:

  (a) Every customer is visited exactly once by exactly one vehicle;
  (b) For every vehicle the total demand does not exceed the vehicle capacity;
  (c) The total tour length of each vehicle does not exceed a given limit;
  (d) Every vehicle starts and ends its tour at the same position.

- Shortest Common Super sequence Problem, where—given a set of strings over an alphabet—a string of minimal length that is a super sequence of each string of the given set has to be found (a super sequence S of string A can be obtained from A by inserting zero or more characters in A).
- Graph-Coloring Problem, which is the problem of finding a coloring of a graph so that the number of colors used is minimal.
- Sequential Ordering Problem, which consists of finding a minimum weight Hamiltonian path 2 on a directed graph with weights on the arcs and on the nodes, subject to precedent constraints among the nodes.
- Connection-Oriented Network Routing, where all packets of the same session follow the same path selected by a preliminary setup phase.
- Connectionless Network Routing where data packets of the same session can follow different paths (Internet-type networks).

Table 11.2 gives an overall view for the application areas of ant colony optimization and researchers who performed it.

Table 11.2 Applications of ACO algorithms

PROBLEM TYPE	PROBLEM NAME	AUTHORS	YEAR
ROUTING	TRAVELING SALESMAN	DORIGO ET AL.	1991, 1996
		DORIGO & GAMBARDELLA	1997
		STÜTZLE & HOOS	1997, 2000
	VEHICLE ROUTING	GAMBARDELLA ET AL.	1999
		REIMANN ET AL.	2004
	SEQUENTIAL ORDERING	GAMBARDELLA & DORIGO	2000
ASSIGNMENT	QUADRATIC ASSIGNMENT	STÜTZLE & HOOS	2000
		MANIEZZO	1999
	COURSE TIMETABLING	SOCHA ET AL.	2002, 2003
	GRAPH COLORING	COSTA & HERTZ	1997
SCHEDULING	PROJECT SCHEDULING	MERKLE ET AL.	2002
	TOTAL WEIGHTED TARDINESS	DEN BESTEN ET AL.	2000
		MERKLE & MIDDENDORF	2000
	OPEN SHOP	BLUM	2005
SUBSET	SET COVERING	LESSING ET AL.	2004
	l-CARDINALITY TREES	BLUM & BLESA	2005
	MULTIPLE KNAPSACK	LEGUIZAMÓN & MICHALEWICZ	1999
	MAXIMUM CLIQUE	FENET & SOLNON	2003
OTHER	CONSTRAINT SATISFACTION	SOLNON	2000, 2002
	CLASSIFICATION RULES	PARPINELLI ET AL.	2002
		MARTENS ET AL.	2006
	BAYESIAN NETWORKS	CAMPOS, ET AL.	2002
	PROTEIN FOLDING	SHMYGELSKA & HOOS	2005
	PROTEIN-LIGAND DOCKING	KORB ET AL.	2006

11.4 Summary

In this chapter, the basic concepts of Particle Swarm Optimization and Ant Colony Optimization are discussed. PSO and ACO algorithms operate on the social behavior of the living organisms. PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation. Thus, PSO is an extremely simple algorithm that seems to be effective for optimizing a wide range of functions. Also, PSO is attractive is that there are only few parameters to adjust. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement. Ant Colony Optimization has been and continues to be a fruitful paradigm for designing effective combinatorial optimization solution algorithms. After more than ten years of studies, both its application effectiveness and its theoretical groundings have been demonstrated, making ACO one of the most successful paradigms in the metaheuristic area.

Review Questions

1. Give the history of the development of swarm intelligence.
2. Define: particles and ants.
3. How are social behaviors of living organisms helpful in developing optimization techniques?
4. Explain in detail on the operation of Particle Swarm Optimization.
5. Mention the advantages and applications of Particle Swarm optimization.
6. Discuss the behavior of real ants and compare it with artificial ants.
7. List the various types of ant colony optimization algorithms.
8. With a neat flowchart, explain the algorithm of Ant Colony Optimization
9. State the various applications of ACO.
10. Compare and Contrast—Genetic Algorithm, Particle Swarm Optimization and Ant Colony Optimization.

Exercise Problems

1. Write a MATLAB program to implement particle swarm optimization and ant colony optimization for traveling salesman problem.
2. Implement a vehicle routing problem using the concept of particle swarm optimization.
3. Write a computer program to perform ant colony optimization for a Graph Coloring Problem.
4. Ant colony optimization algorithm is best suited for protein-folding problems—Justify
5. Develop a C program for performing image segmentation using particle swarm optimization.