CHAPTER 5

SWARM INTELLIGENCE

"There is some degree of communication among the ants, just enough to keep them from wandering off completely at random. By this minimal communication they can remind each other that they are not alone but are cooperating with teammates. It takes a large number of ants, all reinforcing each other this way, to sustain any activity - such as trail-building - for any length of time."

(D. R. Hofstadter, Gödel, Escher, Bach: An Eternal Golden Braid, Penguin Books, 20th Anniversary Edition, 2000; p. 316)

"Ants are everywhere, but only occasionally noticed. They run much of the terrestrial world as the premier soil turners, channelers of energy, dominatrices of the insect fauna ... They employ the most complex forms of chemical communication of any animals and their organization provides an illuminating contrast to that of human beings. ... [Ants] represent the culmination of insect evolution, in the same sense that human beings represent the summit of vertebrate evolution."

(B. Hölldobler and E. O. Wilson, The Ants, Belknap Press, 1990, p. 1)

5.1. INTRODUCTION

Several species benefit from sociality in various ways, usually resulting in greater survival advantages. For instance, life in social groups may increase the mating chances, facilitate the retrieval of food, reduce the probability of attack by predators, allow for a division of labor, and facilitate hunting. Social behaviors have also inspired the development of several computational tools for problem-solving and coordination strategies for collective robotics. This chapter explores the approach known as *swarm intelligence* demonstrating how several ideas from natural systems have been used in the development and implementation of computational swarm systems.

The term *swarm intelligence* was coined in the late 1980s to refer to cellular robotic systems in which a collection of simple agents in an environment interact according to local rules (Beni, 1988; Beni and Wang, 1989). Some definitions of swarm intelligence can be found in the literature:

> "Swarm intelligence is a property of systems of unintelligent agents of limited individual capabilities exhibiting collectively intelligent behavior." (White and Pagurek, 1998; p. 333)

> "[Swarm Intelligence] include[s] any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of social insects and other animal societies." (Bonabeau et al., 1999; p. 7)

Kennedy et al. (2001) quote a passage of a FAQ (Frequently Asked Questions) document from the Santa Fe Institute. They agree with their definition, though they add the possibility of escaping from physical space, allowing swarms to occur in cognitive space as well:

> "We use the term 'swarm' in a general sense to refer to any such loosely structured collection of interacting agents. The classic example of a swarm is a swarm of bees, but the metaphor of a swarm can be extended to other systems with a similar architecture. An ant colony can be thought of as a swarm whose individual agents are ants, a flock of birds is a swarm whose agents are birds, traffic is a swarm of cars, a crowd is a swarm of people, an immune system is a swarm of cells and molecules, and an economy is a swarm of economic agents. Although the notion of a swarm suggests an aspect of collective motion in space, as in the swarm of a flock of birds, we are interested in all types of collective behavior, not just spatial motion." (quoted by Kennedy et al., 2001; p. 102)

In these definitions, an agent can be simply understood as an entity capable of sensing the environment and acting on it. The actions may include the modification of the environment or interactions with other agents.

Millonas (1994) suggests five basic principles of swarm intelligence systems:

- *Proximity*: individuals should be able to interact so as to form social links.
- *Quality*: individuals should be able to evaluate their interactions with the environment and one another.
- *Diversity*: diversity is fundamental in most natural computing approaches, for it improves the capability of the system reacting to unknown and unexpected situations.
- *Stability*: individuals should not shift their behaviors from one mode to another in response to all environmental fluctuation.
- *Adaptability*: the capability of adapting to environmental and populational changes is also very important for swarm systems.

Therefore, a *swarm system* is the one composed of a set of individuals capable of interacting with one another and the environment. And *swarm intelligence* is an emergent property of the swarm system as a result of its principles of proximity, quality, diversity, stability, and adaptability.

Two main lines of research can be identified in swarm intelligence: (i) the works based on social insects, and (ii) the works based on the ability of human societies to process knowledge. Although the resultant approaches are quite different in sequence of steps and sources of inspiration, they present some commonalities. In general terms, both of them rely upon a population (colony or swarm) of individuals (social insects or particles) capable of interacting (directly or indirectly) with the environment and one another. As a result of these interactions there might be a change in the environment or in the individuals of the populations, what may lead to useful emergent phenomena.

This chapter starts with a discussion of social insects with particular emphasis on ants. The foraging behavior, clustering of dead bodies and larval sorting, and the collective prey retrieval capabilities of ants will be discussed, always referring to the computational tools developed with inspiration in these phenomena. Algorithms for the solution of combinatorial optimization problems, clustering and robotics coordination are just a sample of what the collective behavior of

ants can teach us. Insect societies have inspired much more research in computer science and engineering than only ant algorithms, and some of these other natural systems will be discussed in Chapter 8 under the umbrella of artificial life.

The last topic of this chapter, namely the Particle Swarm optimization (PS) approach, has its roots on artificial life, social psychology, engineering, and computer science. It is based on a swarm of particles that move around a search-space and that is capable of locating portions of this space corresponding to peaks of a quality (goodness) function. The swarm of particles self-organizes through an updating mechanism that takes into account the present position of the particle, its best position so far, and the best positions of a number of neighbors. The core idea of the PS approach is that individuals have some knowledge of their own past experience and the past experience of some of their neighbors, and they use this knowledge in order to drive their lives (minds and behaviors).

5.2. ANT COLONIES

Interactions among members of an animal group or society, particularly chemical and visual communication, may be important for the organization of a collective behavior. Furthermore, the interactions among individuals and the environment allow different collective patterns and decisions to appear under different conditions, with the same individual behavior. Though most clearly demonstrable in social insects, these principles are fundamental to schools of fish, flocks of birds, groups of mammals, and many other social aggregates (Deneubourg and Goss, 1989).

The analysis of collective behavior implies a detailed observation of both individual and collective behavior, sometimes combined with mathematical modeling linking the two. There have been a number of computational tools developed with the inspiration taken from the collective behavior of social insects, in particular ant colonies. The resulting models from observed behaviors have also played an important role in the development of computational tools for problem solving.

This section overviews some of the main works inspired by social insects, focusing on ant colonies. It starts with a discussion about *ant colony optimization*, composed of algorithms for discrete optimization developed with inspiration from observations of how some ant species forage for food. Field and experimental observations have led to the conclusion that some ant species organize dead bodies and the brood into clusters of items. One theoretical result derived to model such behaviors gave rise to *ant clustering algorithms*. Ant colonies have also provided good insights into how a set of non-intelligent agents is capable of performing complex tasks. Grouping a number of objects, foraging for food, and prey retrieval are just a sample of tasks performed by ants that inspired the development of collective robotic teams, named *swarm robotics*. Further examples of the collective behavior of other insect societies, such as bees and wasps, will be discussed in the context of Artificial Life in Chapter 8.

5.2.1. Ants at Work: How an Insect Society Is Organized

Ants are the most well studied social insects. People with the most diverse background and interests are fascinated by ants. Biologists and sociobiologists are interested in studying, among other things, how the many events occurring in different levels of an ant colony are related; how such simple organisms can perform fairly complex tasks like collective prey retrieval, foraging, and dead body clustering. Many of these scientists and even philosophers believe that a better understanding of social insects, in particular ant colonies, may lead to a better understanding of how nature works. They believe ant colonies have similar patterns of behavior to other systems, such as the nervous system.

The movie "Antz" by Dreamworks Pictures, directed by Eric Darnell, has an important intellectual appeal and provides an interesting, though not realistic, perspective on how an ant colony is organized. It starts with a worker ant named Z on the couch of a therapist office, moaning about his insignificance and the social order of the colony. "I feel insignificant", he complains. "You've made a big breakthrough", says the therapist. "I have?" replies Z. "Yes", answers the doctor, "You ARE insignificant".

One interesting aspect about Antz is that the whole film is presented under the perspective of an ant colony, with particular emphasis on individual ants. Z falls in love with the queen's daughter, princess Bala, who is engaged to the colony's General Mandible. But the princess does not love the general and foresees a boring and sad future life for herself. Eager to meet princess Bala, Z asks for the help of his best friend Weaver, a soldier ant. But the adventure of being a soldier leads Z into a terrifying battle with the dreaded termites, a species much larger and superior in weaponry. Z is the only survivor of the battle and builds a philosophical and social revolution. Meanwhile, the ambitious General Mandible wants to divide the colony into a superior and stronger race formed by soldiers, and a weak race formed by workers.

This popular view of an ant colony as a Stalinist or Marxist stereotype makes it clear the hierarchical view most of us have in relation to social insects. Most of us believe queen ants rule the colony, and we also usually believe in a hierarchical structure much like human societies. However, while ants are capable of remarkable coordinated activities, such as foraging for food, task allocation, nest building, and grouping of dead bodies, there is no centralized behavior in an ant colony. It is important to stress that the absence of centralized behaviors, so common in human societies, is a fundamental property of natural computing. Actually, the ant queen's main function is to lay eggs, not to rule the colony. Of course, by arguing that the perspective presented in the movie is somewhat misleading, I am not directly criticizing the movie itself, which I personally recommend (Section 5.6.1). Instead, it is a call for attention to the subject matter that is one of the central themes of natural computing: *adaptation and 'intelligence' emerging out of a decentralized system composed of 'unintelligent' agents*.

In a remarkably interesting, brief, and accessible description of how an ant colony is organized, D. Gordon (1999) discourses about her almost two decades of research with ants, particularly harvester ants. Her description ranges from

the individual to the colony level and the implications and motivation for studying ants. The first pages of Chapter 7 of her book provide a very good summary of the complexity involved in a colony of ants and the many tasks they perform; reasons why I quote from it at length:

> "[An ant colony] must collect and distribute food, build a nest, and care for the eggs, larvae and pupae. It lives in a changing world to which it must respond. When there is a windfall of food, more foragers are needed. When the nest is damaged, extra effort is required for quick repairs. Task allocation is the process that results in certain workers engaged in specific tasks, in numbers appropriate to the current situation.
>
> Task allocation is a solution to a dynamic problem and thus it is a process of continual adjustment. It operates without any central or hierarchical control to direct individual ants in particular tasks. Although 'queen' is a term that reminds us of human political systems, the queen is not an authority figure. She lays eggs and is fed and cared for by the workers. She does not decide which worker does what. In a harvester ant colony, many feet of intricate tunnels and chambers and thousands of ants separate the queen, surrounded by interior workers, from the ants working outside the nest and using only the chambers near the surface. It would be physically impossible for the queen to direct every worker's decision about which task to perform and when. The absence of central control may seem counterintuitive, because we are accustomed to hierarchically organized social groups in many aspects of human societies, including universities, business, governments, orchestras, and armies. This mystery underlies the ancient and pervading fascination of social insect societies … .
>
> No ant is able to assess the global needs of the colony, or to count how many workers are engaged in each task and decide how many should be allocated differently. The capacities of individuals are limited. Each worker need make only fairly simple decisions. There is abundant evidence, throughout physics, the social sciences, and biology, that such simple behavior by individuals can lead to predictable patterns in the behavior of the groups. It should be possible to explain task allocation in a similar way, as the consequence of simple decisions by individuals.
>
> Investigating task allocation requires different lines of research. One is to find out what colonies do. The other is to find out how individuals generate the appropriate colony state. An ant is not very smart. It can't make complicated assessments. It probably can't remember anything for very long. Its behavior is based on what it perceives in its immediate environment." (Gordon, 1999; p. 117-119)

The many ant collective tasks to be described in this chapter gave rise to useful tools for problem solving: an algorithm for solving combinatorial optimization problems, and an algorithm for solving clustering problems. All these behaviors served also as inspiration for the development of what is to date known as *swarm robotics*. Swarm robotics is based on the use of a 'colony' of autonomous robots, or agents, with limited capabilities, in order to perform a particular

task. The next three sections describe these patterns of behavior in some ant species and the respective systems developed inspired by them.

5.2.2. Ant Foraging Behavior

There is a large variety of ant species over the world. However, a number of them present similar foraging behaviors. Field and laboratory experiments with many species of ants have demonstrated an interesting capability of exploiting rich food sources without loosing the capability of exploring the environment, as well as finding the shortest path to a food source in relation to the nest (exploration × exploitation). Some of these observations have led to the development of models of ant behavior and further to the proposal of computational algorithms for the solution of complex problems. This section reviews one such algorithm.

Although most ants live in colonies, and thus present some form of social behavior, for few species of ants there is evidence of the presence of leaders, templates, recipes, or blueprints playing a role in the development of foraging patterns. Instead, the process appears to be based on local competition among information (in the form of varying concentrations of trail *pheromone*), and is used by individual ants to generate observable collective foraging decisions (Camazine et al., 2001).

The process of trail formation in ant colonies can be easily observable with the simple experiment illustrated in Figure 5.1. Place a dish of sugar solution (food source) in the vicinity of an ants' nest - Figure 5.1(a). After some time, *forager ants* will discover the sugar and shortly after, through a *recruitment* process, a number of foragers will appear at the food source - Figure 5.1(b). Observation will reveal ants trafficking between the nest and the food source as if they were following a *trail* on the ground - Figure 5.1(c). Note the presence of a few foragers not following the trail; an important behavior for the location of alternative food sources; that is, exploration of the environment.

Generally speaking, *recruitment* is a behavioral mechanism that enables an ant colony to assemble rapidly a large number of foragers at a desirable food source and to perform efficient decision making, such as the selection of the most profitable food source or the choice of the shortest path between the nest and the food source. Different recruitment mechanisms exist (Deneubourg et al., 1986). In *mass recruitment*, a scout (*recruiter*) discovers the food source and returns to the nest, laying a chemical (pheromone) trail. In the nest, some of its nestmates (the recruited) detect the trail and follow it to the food source. There they ingest food and return to the nest *reinforcing* the trail. In *tandem recruitment*, the scout invites ants at the nest to accompany her back to the food. One recruit succeeds in following the leader, the two ants being in close contact. In *group recruitment*, the recruiter leads a group of recruits to the food source by means of a short-range chemical attractant. After food ingestion, in each recruitment type, the recruited become recruiters. This typical self-reinforcing (positive feedback) process is the basis of many activities of ants' societies. The recruitment process slows down when there are fewer ants left to be recruited or when there are other forces, such as alternative food sources, competing for the ants' attention.

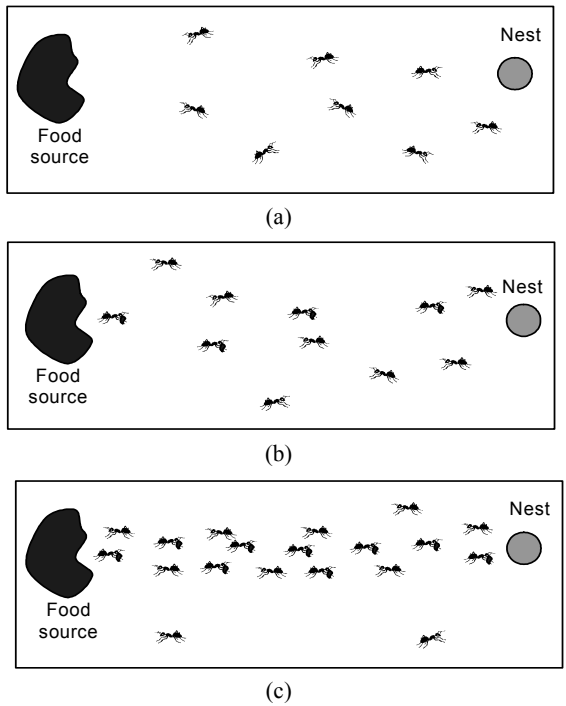


Figure 5.1: In the collective foraging of some ant species, ants recruit nestmates by releasing pheromone on the path from the food source to the nest; a pheromone trail is thus established. (a) Foraging ants. (b) A few ants find the food source and start recruiting nestmates by releasing pheromone. (c) A pheromone trail is formed.

The process of trail formation may be divided in two steps: *trail-laying* and *trail-following* (Camazine et al., 2001). Consider the particular case of mass recruitment and some of the experimental set ups used to observe it and propose mathematical models. The behavior of individual ants during trail recruitment starts when a forager ant, on discovering a food source, returns to the nest and lays a chemical trail all the way home. These chemicals are low molecular weight substances, called *pheromone*, produced in special glands or in the guts. Other ants, either foraging or waiting in the nest, are then stimulated under the influence of the pheromone to start exploiting the food source encountered. In addition to the pheromone trail, recruiter ants may provide other stimuli. The ants recruited follow the trail to the food source and load up with food. On their return journey to the nest, they add their own pheromone to the trail and may even provide further stimulation to other foragers in the nest and on the way. The pheromone thus has two important functions: 1) primarily to define the trail, and also 2) to serve as an orientation signal for ants traveling outside the nest. It is important to remark, however, that the pheromone evaporates, usually very slowly, with time. It means that, for example, if the food source is completely

depleted, the trail will disappear after some time due to the lack of reinforcement.

Goss et al. (1989) and Deneubourg et al. (1990) performed a number of experiments and demonstrated that the trail-laying trail-following behavior of some ant species enables them to find the shortest path between a food source and the nest. They used a simple yet elegant experimental set up with the Argentine ant *Iridomyrmex humilis*. Basically, their experiment can be summarized as follows. Laboratory colonies of *I. humilis* were given access to a food source in an arena linked to the nest by a bridge consisting of two branches of different lengths, arranged so that a forager going in either direction (from the nest to the food source or vice-versa) must choose between one or the other branch, as illustrated in Figure 5.2.

One experimental observation is that, after a transitory phase that can last a few minutes, most of the ants choose the shortest path. It is also observed that an ant's probability of selecting the shortest path increases with the difference in length between the two branches. The explanation for this capability of selecting the shortest route from the nest to the food source comes from the use of pheromone as an indirect form of communication, known as *stigmergy*, mediated by local modifications of the environment. Before starting a discussion about stigmergy in the process of pheromone trail laying and following, it must be asked what would happen if the shorter branch were presented to the colony after the longer one. In this case, the shorter path would not be selected because the longer branch is already marked with a pheromone trail. However, if the pheromone evaporates quickly enough, longer paths would have trouble to maintain stable pheromone trails. This is not the case for most ant species, but it is the approach usually taken by engineers and computer scientists in their computational implementations of *ant colony optimization* (ACO) algorithms.

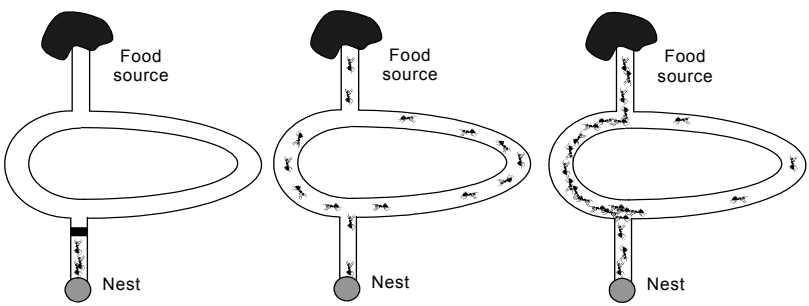


Figure 5.2: An experimental set up that can be used to demonstrate that the ant *I. Humilis* is capable of finding the shortest path between the nest and a food source. (b) The bridge is initially closed. (b) Initial distribution of ants after the bridge is open. (b) Distribution of ants after some time has passed since they were allowed to exploit the food source.

Another observation is that randomness plays an important role in ant foraging. Ants do not follow pheromone trails perfectly. Instead, they have a probabilistic chance of loosing their way as they follow trails. Deneubourg et al. (1986) argue that this 'ant randomness' is not a defective stage on an evolutionary path 'towards an idealistic deterministic system of communication.' Rather, this randomness is an evolutionarily adaptive behavior. In some of their laboratory experiments, they describe one case with two food sources near an ant nest: a rich food source far from the nest, and an inferior source close to the nest. Initially, the ants discover the inferior food source and form a robust trail to that source. But some ants wander off the trail. These lost ants discover the richer source and form a trail to it. Since an ant's pheromone emission is related to the richness of the food source, the trail to the richer source becomes stronger than the original trail. Eventually, most ants shift to the richer source. Therefore, the randomness of the ants provides a way for the colony to explore multiple food sources in parallel; that is, randomness allows for exploration. While positive feedback through pheromone trails encourages exploitation of particular sources, randomness encourages exploration of multiple sources.

Stigmergy

The case of foraging for food in the Argentine ants shows how stigmergy can be used to coordinate the ant's foraging activities by means of self-organization. Self-organized trail-laying by individual ants is a way of modifying the environment to communicate to other nestmates to follow that trail. In the experimental set up of Figure 5.2, when the ants arrive at the branching point, they have to make a probabilistic choice between one of the two branches to follow. Such choice is biased by the amount of pheromone they smell on the branches. This behavior has a self-reinforcing (positive feedback) effect, because choosing a branch will automatically increase the probability that this branch will be chosen again by other ants. At the beginning of the experiment there is no pheromone on the two branches and thus, the choice of a branch will be made with the same probability for both branches. Due to the different branch lengths, the ants traveling on the shortest branch will be the first ones to reach the food source. In their journey back to the nest, these ants will smell some pheromone trail (released by themselves) on the shorter path and will thus follow it. More pheromone will then be released on the shorter path, making it even more attractive for the other ants. Therefore, ants more and more frequently select the shorter path.

5.2.3. Ant Colony Optimization (ACO)

The choice of the shortest path enables ants to minimize the time spent traveling between nest and food source, thus taking less time to complete the route. This also allows ants to collect food more quickly and minimize the risk that this food source is found and monopolized by a stronger competitor, such as a larger colony. Shorter paths also mean lower transportation costs (Camazine et al., 2001).

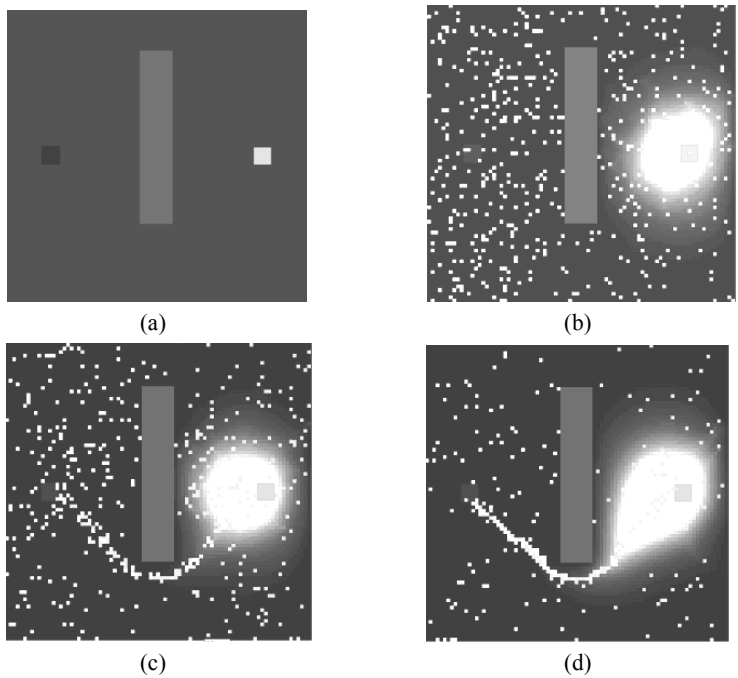


Figure 5.3: Artificial life simulation of pheromone trail laying and following by ants. (a) Environmental setup. The square on the left corresponds to the ants' nest, and the one on the right is the food source. In between the two there is an obstacle whose top part is slightly longer than the bottom part. (b) 500 ants leave the nest in search for food, and release pheromone (depicted in white color) while carrying food back to the nest. (c) The deposition of pheromone on the environment serves as a reinforcement signal to recruit other ants to gather food. (d) A strong pheromone trail in the shorter route is established.

To illustrate this, consider the artificial life simulation (Chapter 8) of pheromone trail-laying, trail-following illustrated in Figure 5.3. In this simulation, a food source is available (big square on the right hand side of the picture) together with an ant nest (big square on the left hand side of the picture) and an obstacle between them (rectangle centered). At first, the virtual ants explore the environment randomly. After they find the food source and start carrying it back to the nest, those ants that choose the shorter path to the nest return first, thus reinforcing the shorter path. Note that the top part of the obstacle is slightly longer than its bottom part, indicating that the shorter route from the nest to the food source is the bottom route. Next, the ants maintain almost only the shortest trail from the food source to the nest, leading to the exploitation of this trail.

The problem of finding the shortest route from the nest to the food source is akin to the well-known traveling salesman problem (TSP), discussed in Chapter 3. The salesman has to find the shortest route by which to visit a given number of cities, each exactly once. The goal is to minimize the cost (distance) of travel.

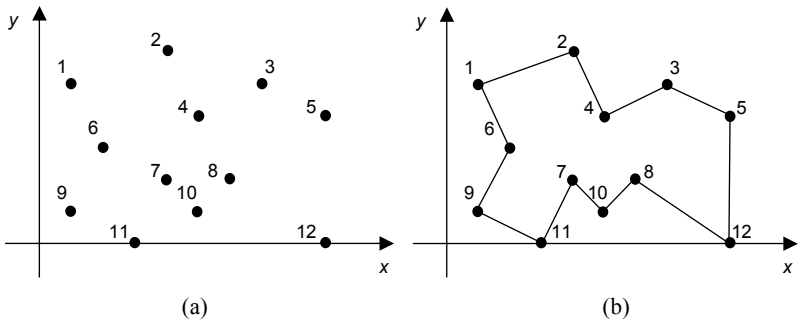


Figure 5.4: A simple instance of the TSP. (a) A set with 12 cities. (b) A minimal route connecting the cities.

Inspired by the experiments of Goss et al. (1989, 1990), Dorigo and collaborators (Dorigo et al., 1996) extended this ant model to solve the traveling salesman problem. Their approach relies on *artificial ants* laying and following *artificial pheromone trails*. Assume the simple TSP instance of Figure 5.4(a). A colony of artificial ants, each independently going from one city to another, favoring nearby cities but otherwise traveling randomly. While traversing a link, i.e., a path from one city to another, an ant deposits some pheromone on it. The amount of pheromone being inversely proportional to the overall length of the tour: the shorter the tour length, the more pheromone released; and vice-versa. After all the artificial ants have completed their tours and released pheromone, the links belonging to the highest number of shorter tours will have more pheromone deposited. Because the pheromone evaporates with time, links in longer routes will eventually contain much less pheromone than links in shorter tours. As the colony of artificial ants is allowed to travel through the cities a number of times, those tours less reinforced (by pheromone) will attract fewer ants in their next travel (Bonabeau and Théraulaz, 2000). Dorigo and collaborators (Dorigo et al., 1996) have found that by repeating this process a number of times, the artificial ants are able to determine progressively shorter routes, such as the one illustrated in Figure 5.4(b).

The Simple Ant Colony Optimization Algorithm (S-ACO)

Ant algorithms were first proposed by Dorigo et al. (1991) and Dorigo (1992) as a multi-agent approach to solve discrete optimization problems, such as the traveling salesman problem and the quadratic assignment problem (QAP). Similarly to all the other fields of investigation discussed in this book, there is a lot of ongoing research. Therefore, there are various versions and extensions of ant algorithms. This section reviews the simplest ACO algorithm, and the next section presents a general-purpose ACO algorithm along with its most relevant features.

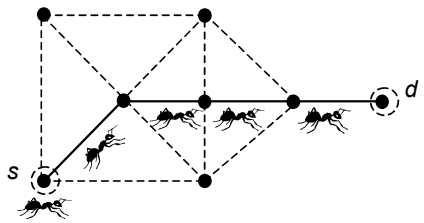


Figure 5.5: An ant travels through the graph from the source node *s* to the destination node *d* thus building a solution (path). (Note that this travel resembles the one performed by the traveling salesperson, who has a number of cities to visit and has to decide which path to take.)

Assuming a connected graph $G = (V, E)$, the simple ACO algorithm (S-ACO) can be used to find a solution to the shortest path problem defined on the graph G. (Appendix B.4.4 brings the necessary fundamentals from *graph theory*.) A solution is a path on the graph connecting a source node s to a destination node d and the path length is given by the number of edges traversed (Dorigo and Di Caro, 1999), as illustrated in Figure 5.5. Associated with each edge (i,j) of the graph there is a variable τ_{ij} termed *artificial pheromone trail*, or simply pheromone. Every *artificial ant* is capable of "marking" an edge with pheromone and "smelling" (reading) the pheromone on the trail.

Each ant traverses one node per iteration step t and, at each node, the local information about its pheromone level, τ_{ij}, is used by the ant such that it can probabilistically decide the next node to move to, according to the following rule:

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{\tau_{ij}(t)}{\sum_{j \in N_i} \tau_{ij}(t)} & \text{if } j \in N_i \\ 0 & \text{otherwise} \end{cases} \qquad \textbf{(5.1)}$$

where $p_{ij}^{k}(t)$ is the probability that ant k located at node i moves to node j, $\tau_{ij}(t)$ is the pheromone level of edge (i,j), all taken at iteration t, and N_i is the set of one step neighbors of node i.

While traversing an edge (i,j), the ant deposits some pheromone on it, and the pheromone level of edge (i,j) is updated according to the following rule:

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \Delta\tau \qquad \textbf{(5.2)}$$

where t is the iteration counter and $\Delta\tau$ is the constant amount of pheromone deposited by the ant.

The analysis of Equation (5.1) and Equation (5.2) lead to the conclusion that when an ant deposits some pheromone on a given edge (i,j), it increases the probability that this edge is selected by another ant, thus reinforcing the trail that passes through this edge. The presence of positive feedback favoring the selection of short paths is clear in this simple model.

Preliminary experiments with the S-ACO algorithm demonstrated that it is capable of determining the shortest route between a simulated nest and food source in a computer simulation similar to the laboratory experiment illustrated in Figure 5.2. However, the behavior of the algorithm becomes unstable when the complexity of the searched graph increases (Dorigo and Di Caro, 1999). In order to overcome this limitation and to avoid a quick convergence of all ants towards sub-optimal routes, an evaporation of the pheromone trails was allowed, thus changing Equation (5.2) into:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau \qquad \textbf{(5.3)}$$

where $\rho \in (0,1]$ is the pheromone decay rate.

General-Purpose Ant Colony Optimization Algorithm

Ant algorithms constitute all algorithms for discrete optimization that took inspiration from the observation of the foraging behavior of ant colonies (Dorigo et al., 1999). Although some authors have already developed versions of ant algorithms for continuous optimization (e.g., Bilchev and Parmee, 1995), not much research has been conducted in this direction and, thus, the focus of the discussion to be presented here is on ACO for discrete optimization.

An ACO algorithm alternates, for a maximum number of iterations `max_it`, the application of two basic procedures (Bonabeau et al., 1999). Assume a search is being performed on a connected graph $G = (V,E)$ with V nodes and E edges:

1. A parallel solution construction/modification procedure in which a set of N ants builds/modifies in parallel N solutions to the problem under study.
2. A pheromone trail updating procedure by which the amount of pheromone trail on the problem graph edges is changed.

The process of building or modifying a solution is made in a probabilistic fashion, and the probability of a new edge to be added to the solution being built is a function of the edge *heuristic desirability* η, and of the *pheromone trail* τ deposited by previous ants. The heuristic desirability η expresses the likelihood of an ant moving to a given edge. For instance, in cases the minimal path is being sought among a number of edges, the desirability η is usually chosen to be the inverse of the distance between a pair of nodes. The modification (updating) of pheromone trails is a function of both the evaporation rate ρ (see Equation (5.3)) and the quality of the solutions produced. Algorithm 5.1 depicts the standard, or general-purpose, ACO algorithm to perform discrete optimization (Bonabeau et al., 1999). The parameter e is the number of edges on the graph, and `best` is the best solution (path) found so far; the pheromone level of each edge is only updated after all ants have given their contributions (laid some pheromone). The generic algorithm assumes that the heuristic desirability function of an edge and how to update the pheromone trail have already been defined.

```
procedure [best] = ACO(max_it,N,τ0)
    initialize τij   //usually initialized with the same τ0
    initialize best
    place each ant k on a randomly selected edge
    t ← 1
    while t < max_it do,
        for i = 1 to N do,  //for each ant
            build a solution by applying a probabilistic
            transition rule (e − 1) times.
            //The rule is a function of τ and η
            //e is the number of edges on the graph G
        end for
        eval the cost of every solution built
        if an improved solution is found,
            then update the best solution found
        end if
        update pheromone trails
        t ← t + 1
    end while
end procedure
```

Algorithm 5.1: Standard ACO for discrete optimization.

Selected Applications from the Literature: A Brief Description

Algorithm 5.1 presents the general-purpose ACO algorithm. It is important to note the basic features of an ACO algorithm:

- A colony of ants that will be used to build a solution in the graph.
- A probabilistic transition rule responsible for determining the next edge of the graph to which an ant will move.
- A heuristic desirability that will influence the probability of an ant moving to a given edge.
- The pheromone level of each edge, which indicates how 'good' it is.

To illustrate how to use this simple algorithm in practical problems, consider the following two examples. The first example - traveling salesman problem - was chosen because it was one of the first tasks used to evaluate an ACO algorithm. In addition, the TSP is a shortest path problem to which the ant colony metaphor is easily adapted, it is an NP-hard problem, it is didactic, and it has been broadly studied (Lawer et al., 1985). The second example - vehicle routing - illustrates how a small variation in the ACO for the traveling salesman problem allows it to be applied to the vehicle routing problem.

Traveling Salesman

In the TSP the goal is to find a closed tour of minimal length connecting e given cities, where each city must be visited once and only once. Let d_{ij} be the Euclidean distance between cities i and j given by:

$$d_{ij} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2} \quad \textbf{(5.4)}$$

where x_m and y_m are the coordinates of city m, on the x-y plane. This problem can be more generally defined on a graph $G = (V,E)$, where the cities correspond to the nodes V and the connections between them are the edges E.

In (Dorigo et al., 1996; Dorigo and Gambardella, 1997a,b), the authors introduced the Ant System (AS) as an ACO algorithm to solve the TSP problem. In AS ants build solutions to the TSP by moving on the problem graph from one city to another until they complete a tour. During each iteration of the AS, an ant k, $k = 1, \ldots, N$ (a colony of N ants is assumed), builds a tour by applying a probabilistic transition rule ($e - 1$) times, as described in Algorithm 5.1. The iterations are indexed by the iteration counter t and a maximum number of iteration steps, `max_it`, is defined for the iterative procedure of adaptation, i.e., $1 \le t \le$ `max_it`. The description to be presented here follows those of (Dorigo et al., 1996; Dorigo and Gambardella, 1997a,b; Bonabeau et al., 1999).

For each ant, the transition from city i to city j at iteration t depends on: 1) the fact that the city has already been visited or not; 2) the inverse of the distance d_{ij} between cities i and j, termed *visibility* $\eta_{ij} = 1/d_{ij}$; and 3) the amount of pheromone τ_{ij} on the edge connecting cities i and j. As in the TSP problem each ant has to visit a city only once, some knowledge of the edges (cities) already visited has to be kept by the ants, thus they must possess some sort of memory, also called a *tabu list*. The memory is used to define the set J_i^k of cities that the ant k still has to visit while in city i.

The probability of an ant k going from city i to city j at iteration t is given by the following transition rule:

$$p_{ij}^k(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in J_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta} & \text{if } j \in J_i^k \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(5.5)}$$

where $\tau_{ij}(t)$ is the pheromone level of edge (i,j), η_{ij} is the visibility of city j when in city i, and J_i^k is the tabu list of cities still to be visited by ant k from node i. The parameters α and β are user-defined and control the relative weight of trail intensity $\tau_{ij}(t)$ and visibility η_{ij}. For instance, if $\alpha = 0$, the closest cities are more likely to be chosen, and if $\beta = 0$, by contrast, only pheromone amplification is considered.

Similarly to the simple ACO algorithm (S-ACO), while traversing an edge (city), an ant lays some pheromone on that edge. In the AS, the quantity $\Delta\tau_{ij}^k(t)$ of pheromone released on each edge (i,j) by ant k depends on how well it has performed according to the following rule:

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q / L^k(t) & \text{if } (i,j) \in T^k(t) \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(5.6)}$$

where $L^k(t)$ is the length of the tour $T^k(t)$ performed by ant k at iteration t, and Q is another user-defined parameter.

The pheromone updating rule is the same as the one used for the simple ACO algorithm (Equation (5.3)) taking into account the differential amount of pheromone released by each ant in each iteration:

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad \textbf{(5.7)}$$

where $\rho \in (0,1]$ is the pheromone decay rate, $\Delta\tau_{ij}(t) = \sum_k \Delta\tau^k_{ij}(t)$, and $k = 1, \ldots, N$ is the index of ants.

According to Dorigo et al. (1996), the number N of ants is an important parameter of the algorithm. Too many ants reinforce sub-optimal trails leading to bad solutions, whereas too few ants would not result in a sufficient cooperative behavior due to pheromone decay. They suggested to use $N = e$; that is, a number of ants equals to the number of cities.

```
procedure [best] = AS-TSP(max_it,α,β,ρ,N,e,Q,τ0,b)
    initialize τij   //usually initialized with the same τ0
    place each ant k on a randomly selected city
    Let best be the best tour found from the beginning and
    Lbest its length
    t ← 1
    while t < max_it do,
        for i = 1 to N do,          //for every ant
            //e is the number of cities on the graph
            build tour T^k(t) by applying (e − 1) times the fol-
            lowing step:
            At city i, choose the next city j with probabil-
            ity given by Equation (5.5)
        end for
        eval the length of the tour performed by each ant
        if a shorter tour is found,
            then update best and Lbest
        end if
        for every city e do,
            Update pheromone trails by applying the rule:
            τij(t+1) ← (1−ρ)τij(t) + Δτij(t) + b.Δτ^b_ij(t), where
            Δτij(t) = Σk Δτ^k_ij(t), k = 1,...N;
            Δτ^k_ij(t) = { Q/L^k(t)   if (i,j) ∈ T^k(t), and
                         { 0          otherwise
            Δτ^b_ij(t) = { Q/L_best   if (i,j) ∈ best.
                         { 0          otherwise
        end for
        t ← t + 1
    end while
end procedure
```

Algorithm 5.2: Ant system for the traveling salesman problem (AS-TSP).

In an effort to improve the AS performance when applied to the TSP problem, the authors also introduced the idea of "elitist ants", a term borrowed from evolutionary algorithms. An elitist ant is the one that reinforces the edges belonging to the best route found so far, `best`. Such ants would reinforce the trail (cities or edges) of the best tour by adding $b.Q/L_{best}$ to the pheromone level of these edges, where b is the number of elitist ants chosen, and L_{best} is the length of the best tour found from the beginning of the trial. Algorithm 5.2 depicts the ant system for the traveling salesman problem (AS-TSP). The authors employed the following values for the parameters in their experiments: $\alpha = 1$, $\beta = 5$, $\rho = 0.5$, $N = e$, $Q = 100$, $\tau_0 = 10^{-6}$, and $b = 5$.

It is important to remark that the results presented in Dorigo et al. (1996) were interesting but disappointing. The AS-TSP could outperform other approaches, such as a GA, when applied to small instances of the TSP, but its performance was poor when applied to TSP problems with a large number of cities. In order to enhance even further the performance of the ACO approach, the authors altered the transition rule (Equation (5.5)), employed local updating rules of the pheromone trail (Equation (5.6)), and used a candidate list to restrict the choice of the next city to visit (Dorigo and Gambardella, 1997a,b). These modifications made the algorithm competitive in relation to other approaches from the literature. Further improvements, variations, and applications of ant colony optimization algorithms can be found in Corne et al. (1999), Dorigo et al. (2002), Dorigo and Stützle (2004), and de Castro and Von Zuben (2004).

Vehicle Routing

The simplest *vehicle routing problem* (VRP) can be described as follows: given a fleet of vehicles with uniform capacity, a common depot, and several costumer demands (represented as a collection of geographical scattered points), find the set of routes with overall minimum route cost which service all the demands. All the itineraries start and end at the depot and they must be designed in such a way that each costumer is served only once and just by one vehicle. Note that the VRP is closely related to the TSP as presented here; the VRP consists of the solution of many TSPs considering a single common start and end city for each TSP, which now has a limited capacity.

Bullnheimer et al. (1999a,b) and Bullnheimer (1999) modified the AS-TSP algorithm to solve the vehicle routing problem with one central depot and identical vehicles. The VRP problem was represented by a complete *weighted directed graph* $G = (V,E,w)$, where $V = \{v_0,v_1, \ldots , v_N\}$ is the set of nodes of G, $E = \{(v_i,v_j) : i \neq j\}$ its set of edges, and $w_{ij} \geq 0$ is a weight connecting nodes i and j. The node v_0 denotes the depot and the other vertices are the customers or cities. The weights w_{ij}, associated with edge (v_i,v_j), represent the distance (time or cost) between v_i and v_j. A demand $d_i \geq 0$ and a service time $\delta_i \geq 0$ are associated with each customer v_i. The respective values for the depot are $v_0 = d_0 = 0$.

The objective is to find the minimum cost vehicle routes where:

- Every customer is visited only once by only one vehicle.

- All vehicle routes begin and end at the depot.
- For every vehicle route the total demand does not exceed the vehicle capacity D.
- For every vehicle route the total route length, including service times, does not exceed a given threshold L.

Similarly to the AS-TSP, to solve the vehicle routing problem one ant is placed on each node of the graph and the ants construct solutions by sequentially visiting a city, until all cities have been visited. Whenever the choice of another city would lead to an unfeasible solution; that is, vehicle capacity greater than D or total route length greater than the threshold L, the depot is chosen and a new tour is started.

The VRP problem was solved by applying the procedure presented in Algorithm 5.1 with the probabilistic transition rule presented in Equation (5.5), which takes into account the pheromone trail τ_{ij} and the heuristic desirability η_{ij}. The heuristic desirability of a city follows a *parametrical saving function* (Paessens, 1988):

$$\eta_{ij} = w_{i0} + w_{0j} - g.w_{ij} + f.\ |w_{i0} - w_{0j}| \qquad \textbf{(5.8)}$$

where g and f are two user-defined parameters.

In the standard ant system algorithm, after an ant has constructed a solution, the pheromone trails are laid by all ants depending on the objective value of the solution. It was discussed in the previous example that by using σ elitist ants, the performance of the algorithm was improved for the TSP problem. In (Bullnheimer et al., 1999b), the authors suggested that by ranking the ants according to the solution quality and using only the best ranked ants, also called elitist ants, to update the pheromone trails would improve even further the performance of the algorithm for the VRP problem. The new updating rule is as follows:

$$\tau_{ij}(t) \leftarrow (1 - \rho)\tau_{ij}(t) + \Delta\tau^{r}_{ij}(t) + \sigma.\Delta\tau^{+}_{ij}(t) \qquad \textbf{(5.9)}$$

where $\rho \in (0,1]$ is the pheromone decay rate. Only if an edge (v_i,v_j) was used by the μ-th best ant, the pheromone trail is increased by a quantity

$$\Delta\tau^{r}_{ij}(t) = \sum_{\mu=1}^{\sigma-1} \Delta\tau^{\mu}_{ij}(t) \qquad \textbf{(5.10)}$$

where $\Delta\tau^{\mu}_{ij}(t) = (\sigma - \mu)/L_{\mu}(t)$ if ant μ uses the edge (v_i,v_j), and $\Delta\tau^{\mu}_{ij}(t) = 0$ otherwise. $L_{\mu}(t)$ is the length of the tour ant μ performs at iteration t. All edges belonging to the best tour determined so far are emphasized as if σ elitist ants had used them. Therefore, each elitist ant increases the trail intensity by an amount $\Delta\tau^{+}_{ij}(t)$ that is equal to $1/L^{+}$ if edge (v_i,v_j) belongs to the so far best solution, and zero otherwise.

Scope of ACO Algorithms

Although the traveling salesman problem is an intuitive application of ACO, many other discrete (combinatorial) optimization problems can be solved with

ACO. Bonabeau et al. (2000) claim that ACO is currently the best available heuristic for the sequential ordering problem, for real-world instances of the quadratic assignment problem, and is among the best alternatives for the vehicle and network routing problems. Good surveys of applications of ACO algorithms, including a number of main references can be found in Bonabeau et al. (1999, 2000), Dorigo et al. (1999), Dorigo and Di Caro (1999), and Dorigo and Stützle (2004). The main problems tackled by ACO algorithms are: TSP, network routing, graph coloring, shortest common super sequence, quadratic assignment, machine scheduling, vehicle routing, multiple knapsack, frequency assignment, and sequential ordering.

ACO algorithms are suitable for solving problems that involve graph searching, mainly minimal cost problems. Similarly to all the other approaches discussed in this text, ACO algorithms may be applied only when more classical approaches, such as dynamic programming or other methods cannot be (efficiently) applied. Dorigo and Di Caro (1999) suggest a list of particular cases of interest for ACO algorithms:

- *NP problems*: a problem is said to be solvable in polynomial time if there is an algorithm to solve it in a time that is a polynomial function of the size of the input. NP stands for *nondeterministic polynomial*. It is a large class of problems that have the property that if any solution exists, there is at least one solution which may be verified as correct in polynomial time. Less rigorously, a problem is NP if we can check a proposed solution in polynomial time. A more detailed description of problem complexity is provided in Appendix B.3.3.
- *Static and dynamic combinatorial optimization problems*: a combinatorial problem is the one in which there is a large discrete set of possible solutions. The static problems are those whose characteristics do not change over time, e.g., the classic traveling salesman problem. The shortest path problems in which the properties of the graph representation of the problem change over time constitute the dynamic problems (e.g., network routing).
- *Distributed problems*: those in which the computational architecture is spatially distributed (e.g., parallel or network processing), and may require a set of agents to find suitable solutions. The inherent distribution of agents' resources, such as knowledge and capability, promotes the need of a cooperative work.

From Natural to Artificial Ants

Most of the ideas of ACO were inspired by the foraging behavior of trail-laying trail-following ants. In particular, the main features of ACO algorithms are the use of: 1) a colony of cooperating individuals, 2) a pheromone trail for local indirect (stigmergic) communication, 3) a sequence of local moves to find the shortest paths, and 4) a probabilistic decision rule using local information (Dorigo et al., 1999). Table 5.1 clarifies how the ACO approach uses the biological terminology borrowed from the natural world of ants.

Table 5.1: Interpretation of the biological terminology into the computational Ant Colony Optimization (ACO) algorithms.

Biology (Ant Foraging)	ACO Algorithms
Ant	Individual (agent) used to build (construct) a solution to the problem
Ant colony	Population (colony) of cooperating individuals known as artificial ants
Pheromone trail	Modification of the environment caused by the artificial ants in order to provide an indirect mean of communication with other ants of the colony This allows the assessment of the quality of a given edge on a graph
Pheromone evaporation	Reduction in the pheromone level of a given path (edge) due to aging

5.2.4. Clustering of Dead Bodies and Larval Sorting in Ant Colonies

Chrétien (1996) has performed a number of experiments to study the organization of cemeteries of the ant *Lasius niger*. It has been observed, in several species of ants, workers grouping corpses of ants, or parts of dead ants, to clean up their nests, thus forming cemeteries. Further research on different ant species (Deneubourg et al., 1991) has confirmed the cemetery organization capability of ants.

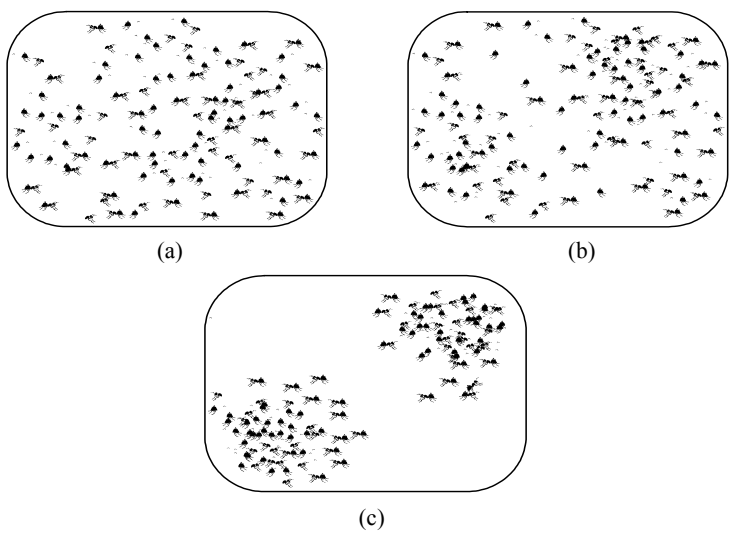


Figure 5.6: A qualitative example of cemetery formation in ants. (a) The corpses are initially randomly distributed over the arena. (b) After some time (e.g., a few hours) the workers start piling the corpses. (c) Clusters of corpses are formed by the worker ants.

The phenomenon observed in these experiments is the aggregation of dead bodies by workers. If dead bodies, or more precisely, items belonging to dead bodies, are randomly distributed in space at the beginning of the experiment, the workers will form clusters with these items within a few hours (Bonabeau, 1997). If the experimental arena is not sufficiently large or if it contains spatial heterogeneities, the clusters will be formed along the borders of the arena or along the heterogeneities. Figure 5.6 presents a qualitative illustration of how ants form clusters of corpses in a given arena. It is assumed that a number of corpses are spread over the arena and worker ants are allowed to pick them up and drop them somewhere within the arena.

The basic mechanism behind this type of clustering (aggregation) phenomenon is an attraction between dead items (corpses) mediated by ant workers. Small clusters of items grow by attracting more workers to deposit more dead items. It is this positive feedback mechanism that leads to the formation of larger and larger clusters of dead items. However, the exact individual behavior that leads to this positive feedback mechanism and the adaptive significance of corpse clustering are still unclear. Cleaning the nest by getting rid of dead items is an important task, though (Bonabeau, 1991; Bonabeau et al., 1999).

Another related phenomenon is larval sorting by the ant *Leptothorax unifasciatus*, a phenomenon widespread and broadly studied for this ant species (Franks and Sendova-Franks, 1992). Workers of this species gather the larvae according to their size. All larvae tend to be clustered, with small larvae located in the center and larger larvae in the periphery of the brood.

Stigmergy

The stigmergic principle is also clear in this corpse-gathering behavior of ant colonies. The observations show that ants tend to put the corpses of dead ants together in cemeteries in certain places far from the nest and that grow in size with time. If a large number of corpses are scattered outside the nest, the ants from the nest will pick them up, carry them about for a while and drop them. Within a short period of time, it can be observed that corpses are being placed in small clusters. As time goes by, the number of clusters decrease and the size of the remaining clusters increase, until eventually all the corpses are piled in one or two large clusters. Therefore, the increase in cluster size reinforces the enlargement of a cluster, and the reverse is also true.

5.2.5. Ant Clustering Algorithm (ACA)

In addition to path optimization to food sources, *data clustering* is another problem solved by real ants. Clustering problems have already been explored in the previous chapter while describing the self-organizing networks and some of their applications. Before proceeding with the presentation of the clustering algorithm derived from models of cemetery organization and brood sorting in ant colonies, the reader should refer to Appendix B.4.5 for further comments on data clustering.

The Standard Ant Clustering Algorithm (ACA)

In order to model the two phenomena of dead body clustering and larval sorting in some ant species, Deneubourg et al. (1991) have proposed two closely related models. Both models rely on the same principles, but the first one (cemetery organization) is a special case of the second one (brood sorting). While the clustering model reproduces experimental observations more faithfully, the second one has given rise to more practical applications. In their study, a colony of ant-like agents, randomly moving in a bi-dimensional grid, was allowed to move basic objects so as to cluster them.

The general idea is that isolated items should be picked up and dropped at some other location where more items of that type are present. Assume that there is a single type of item in the environment, and a number of "artificial ants" whose role will be to carry an item and drop it somewhere in the environment. The probability p_p that a randomly moving ant, who is not currently carrying an item, picks up an item is given by

$$p_p = \left(\frac{k_1}{k_1 + f} \right)^2 \qquad \textbf{(5.11)}$$

where f is the *perceived fraction of items* in the neighborhood of the ant, and k_1 is a threshold constant. For $f << k_1$, $p_p \approx 1$, thus the probability that the ant picks up an item is high when there are not many items in its neighborhood. In the case $f >> k_1$, $p_p \approx 0$, thus the probability that an item is removed from a dense cluster is small; that is, when there are many items in the neighborhood of an item it tends to remain there.

The probability p_d that a randomly moving ant, who is currently carrying an item, deposits that item in a certain position is given by

$$p_d = \left(\frac{f}{k_2 + f} \right)^2 \qquad \textbf{(5.12)}$$

where k_2 is another threshold constant. In this case, if $f << k_2$, then $p_d \approx 0$, thus the probability that the ant deposits the item is low when there are not many items in its neighborhood. In the case $f >> k_2$, $p_d \approx 1$, thus the probability that the item is deposited in a dense cluster is high; that is, items tend to be deposited in regions of the arena where there are many items.

In order to use this theoretical model to generate an *ant clustering algorithm* (ACA), two main aspects have yet to be discussed. First, in which kind of environment are the ants going to move? That is, how to project the space of attributes so that it is possible to visualize clusters? Second, how is the function f (the perceived fraction of items) going to be defined?

Projecting the Space of Attributes into a Bi-Dimensional Grid

In the standard ant clustering algorithm (Lumer and Faieta, 1994), ants are assumed to move on a discrete 2-D board. This board may be considered as a bi-

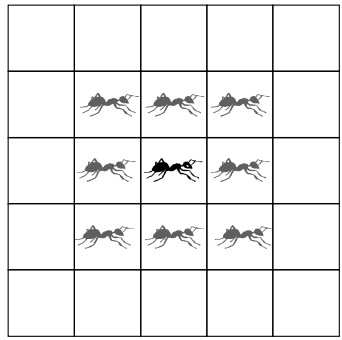


Figure 5.7: A bi-dimensional grid with 5 × 5 cells. An ant (dark ant in the center cell) is placed on a cell with its eight neighbors depicted (gray ants surrounding the center ant).

dimensional grid B of $m \times m$ cells. An ant is allowed to travel from one side of the board to another, meaning that the board is toroidal, and the ant perceives a surrounding region area, i.e., a squared neighborhood $\text{Neigh}_{(s \times s)}$ of $s \times s$ cells surrounding the current site (cell) r. In the example illustrated in Figure 5.7 the reference ant, the one in the middle, perceives a neighborhood of 3 × 3 cells.

This representation of the arena by a bi-dimensional board resembles the bi-dimensional grid of Kohonen's self-organizing map (Section 4.4.5), but is more akin to a bi-dimensional grid in a *cellular automaton* (Section 7.3). Therefore, each ant is modeled by an automaton able to move on the grid and displace objects according to probabilistic rules which are based only upon information of the local environment (neighborhood). The combined actions of a set (colony) of these automata (ants) will lead to the clustering in the same spatial region of the grid of data items belonging to the same classes.

In the algorithm proposed by Lumer and Faieta (1994), the bi-dimensional grid is used as a low dimensional space in which to project the space of attributes of the input data. In this case, the dimension is $z = 2$ so that, after the clustering by ants, clusters present the following property: intra-cluster distances should be small in relation to inter-cluster distances. Therefore, for $z = 2$, the input data are mapped onto a board composed of m^2 cells. Such a mapping, after the ant clustering process, must preserve the neighborhood inherent in the input data without creating too many new neighbors in z-dimensions that do not exist in the L-dimensional space corresponding to the original space of the input data set.

The idea of the ant clustering algorithm is thus to initially project the input data items in the bi-dimensional grid and scatter them randomly, as illustrated in Figure 5.8. The artificial ants (agents) are also spread all over the bi-dimensional grid, and allowed to move, pick up, and deposit items according to some similarity measure and given probabilities.

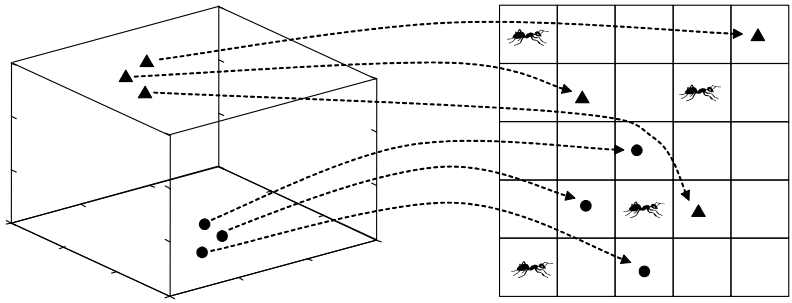


Figure 5.8: Projecting the input data into the bi-dimensional grid. (a) The input data lies in a three-dimensional space. (b) Before the ant clustering process starts, each input datum is projected into a random position of the bi-dimensional grid.

Defining the Perceived Fraction of Items: f

The second aspect that has to be accounted for is the definition of how f, the perceived fraction of items, is evaluated. Similarly to evolutionary algorithms (Chapter 3), f will be defined by the problem under study. For instance, in the context of robotic systems, Deneubourg et al. (1991) defined f as the quotient between the number M of items encountered during the last T time units and the largest possible number of items that can be encountered during these T time units.

Assuming that ants move in a bi-dimensional grid, the standard ACA can be summarized as in Algorithm 5.3. Let 'unloaded ant' be an ant not carrying an item, assume f is problem dependent, p_p and p_d are the probabilities of picking up and dropping off an item, respectively, and N is the number of ants. In this standard algorithm a cell can only be occupied by a single item.

Selected Applications from the Literature: A Brief Description

The standard ant clustering algorithm is described in Algorithm 5.3. Its applications in clustering range from exploratory data analysis to graph partitioning. To illustrate how to use the standard algorithm to different types of data, including to real-world data, it will be first described its application to simple numeric data and then to bioinformatics data. Both examples project the data into a bi-dimensional grid, but use different local density functions and picking up and dropping off probability functions. The examples focus on how to use the ACA standard algorithm to tackle these problems. Further details can be found in the cited literature.

Exploratory Data Analysis

Lumer and Faieta (1994) applied the standard ACA to exploratory data analysis. Their aim was to identify clusters in an unlabelled data set. To accomplish this

```
procedure [] = ACA(max_it,N,k1,k2)
    project every item i on a random cell of the grid
    place every ant on a random cell of the grid unoccupied
    by ants
    t ← 1
    while t < max_it do,
       for i = 1 to N do,  //for every ant
          compute f(xi)
          if unloaded ant AND cell occupied by item xi, then
             compute pp(xi)       //pick up probability
             pick up item xi with probability pp(xi)
          else if ant carrying item xi AND cell empty, then
             compute pd(xi)       //drop off probability
             deposit (drop) item xi with probability pd(xi)
          end if
          move randomly to an unoccupied neighbor
       end for
       t ← t + 1
    end while
    print location of items
end procedure
```

Algorithm 5.3: Standard ant clustering algorithm (ACA).

task, the input data items belonging to an L-dimensional Euclidean space, $\mathbf{X} \in \Re^L$, were projected into a bi-dimensional grid Z^2, which can be considered as a discretization of a real space. The artificial ants were allowed to move about in this grid (discrete space) and perceive a surrounding region of area s^2 that corresponds to a square $\text{Neigh}_{(s\times s)}$ of $s \times s$ cells surrounding the current cell r, as illustrated in Figure 5.7.

The authors suggested the following function f to calculate the local density of items in the neighborhood of object $\mathbf{x}_i$ situated at site r:

$$f(\mathbf{x}_i) = \begin{cases} \dfrac{1}{s^2} \displaystyle\sum_{\mathbf{x}_j \in \text{Neigh}_{(s\times s)}(r)} \left[1 - \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\alpha}\right] & \text{if } f > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5.13}$$

where $f(\mathbf{x}_i)$ is a measure of the average similarity of object $\mathbf{x}_i$ with another object $\mathbf{x}_j$ in the neighborhood of $\mathbf{x}_i$, α is a factor that defines the scale of dissimilarity, and $d(\mathbf{x}_i,\mathbf{x}_j)$ is the Euclidean distance between two data items in their original L-dimensional space. The parameter α determines when two items should or should not be located next to each other. For instance, if α is too large, there is not much discrimination between two items, leading to the formation of clusters composed of items that should not belong to the same cluster; and vice-versa.

Note: the distance $d(\cdot,\cdot)$ between items is calculated in their original space, not in the projected space.

Equation (5.13) takes into account the neighborhood of a cell ($1/s^2$), and the relative distance between items in this neighborhood. It proposes that the smaller the distance between the item an ant is carrying and the items in its neighborhood, the higher the local density of items perceived and, thus, the higher the probability of this item being dropped in that position (cell). Inversely, the more different the item being carried is from the items on a given neighborhood, the smaller the likelihood that this item is dropped in that vicinity.

The probabilities of picking up or depositing an item are given by the following equations:

$$p_p(\mathbf{x}_i) = \left(\frac{k_1}{k_1 + f(\mathbf{x}_i)} \right)^2 \quad \textbf{(5.14)}$$

$$p_d(\mathbf{x}_i) = \begin{cases} 2f(\mathbf{x}_i) & \text{if } f(\mathbf{x}_i) < k_2 \\ 1 & \text{otherwise} \end{cases} \quad \textbf{(5.15)}$$

where k_1 and k_2 are two constants playing similar roles to the constants in Equation (5.11) and Equation (5.12) respectively.

To illustrate the performance of this algorithm, the authors applied it, with p_p and p_d determined by Equation (5.14) and Equation (5.15), to an artificially generated set of points in $\Re^2$ (see Computational Exercise 3, Section 5.6.2).

Although the results obtained by Lumer and Faieta (1994) with the standard ant clustering algorithm when applied to the simple benchmark problem of Computational Exercise 3 were good, the authors observed the presence of more clusters in the projected space than in the original distribution of data. In order to alleviate this problem, they proposed a number of variations in their initially proposed algorithm: ants with different speeds, ants with memory, and behavioral switches:

1) *The use of ants with different moving speeds*: each ant in the colony is allowed to have its own speed. The speed of the swarm is distributed uniformly in the interval $[1, v_{\max}]$, where $v_{\max} = 6$ is the maximal speed of an ant, and v corresponds to the number of grid units (cells) walked per time unit by an ant along a given grid axis. This pace influences the probability an ant picks up or drops a given item according to the following equation:

$$f(\mathbf{x}_i) = \begin{cases} \dfrac{1}{s^2} \displaystyle\sum_{\mathbf{x}_j \in \text{Neigh}_{(s \times s)}(r)} \left[1 - \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\alpha + \alpha(v-1)/v_{\max}} \right] & \text{if } f > 0 \\ 0 & \text{otherwise} \end{cases} \quad \textbf{(5.16)}$$

   Therefore, fast moving ants are not as selective as slow ants in their estimation of the average similarity of an object to its neighbors.

2) *A short-term memory for the ants*: ants can remember the last m items they have dropped along with their locations. Each time an item is picked up, it is compared to the elements in the ant's memory, and the ant goes toward the location of the memorized element most similar to the one just collected. This behavior leads to the formation of a smaller number of sta-

tistically equivalent clusters, because similar items have a lower probability of starting a new cluster.

3) *Behavioral switches*: ants are endowed with the capability of destroying clusters that have not been reinforced by the piling up of more items during some specific time interval. This is necessary because items are less likely to be removed as clusters of similar objects are formed.

Bioinformatics Data

As discussed previously, ant clustering algorithms often generate a number of clusters that is much larger than the natural number of clusters in the data set. Besides, the standard ACA does not stabilize in a particular clustering solution; it constantly constructs and deconstructs clusters during adaptation. To overcome these difficulties, Vizine et al. (2005) proposed three modifications in the standard ACA: 1) a cooling schedule for k_1; 2) a progressive vision field that allows ants to 'see' over a wider area; and 3) the use of a pheromone function added to the grid as a way to promote reinforcement for the dropping of objects at denser regions of the grid.

Concerning the cooling schedule for k_1, the adopted scheme is simple: after one cycle has passed (10,000 ant steps), the value of the parameter k_1 starts being geometrically decreased, at each cycle, until it reaches a minimal allowed value, $k_{1\min}$, which corresponds to the stopping criterion for the algorithm:

$$\begin{aligned} k_1 &\leftarrow 0.98 \times k_1 \\ k_{1\min} &= 0.001 \end{aligned} \qquad \textbf{(5.17)}$$

In the standard ACA, the value of the perceived fraction of items, $f(\mathbf{x}_i)$, depends on the vision field, s^2, of each ant. The definition of a fixed value for s^2 may sometimes cause inappropriate behaviors, because a fixed perceptual area does not allow the distinction of clusters with different sizes. To overcome this difficulty, a progressive vision scheme was proposed (Vizine et al., 2005). When an ant i perceives a 'large' cluster, it increments its perception field (s_i^2) up to a maximal size. A large cluster is detected using function $f(\mathbf{x}_i)$; when $f(\mathbf{x}_i)$ is greater than a given threshold θ, then the vision s^2 of an ant is incremented by n_s units until it reaches its maximum value:

$$\begin{aligned} \text{If } f(\mathbf{x}_i) > \theta \quad &\text{and} \quad s^2 \le s^2{}_{\max} \\ &\text{then } s^2 \leftarrow s^2 + n_s \end{aligned} \qquad \textbf{(5.18)}$$

where $s^2{}_{\max} = 7 \times 7$ and $\theta = 0.6$ in their implementation.

Inspired by the way termites use pheromone to build their nests (Section 2.2.3), Vizine et al. (2005) proposed adding pheromone to the objects the ants carry and allowing the transference of this pheromone to the grid when an object is dropped.

The pheromone added to the carried items lead to the introduction of a pheromone level function $\phi(i)$ in the grid, which corresponds to how much pheromone is present at each grid cell i. During each iteration, the artificial pheromone $\phi(i)$

pheromone $\phi(i)$ at each cell of the grid evaporates at a fixed rate. This pheromone influences the picking and dropping probabilities as follows:

$$P_p(i) = \frac{1}{f(i)\phi(i)}\left(\frac{k_1}{k_1 + f(i)}\right)^2 \quad \textbf{(5.19)}$$

$$P_d(i) = f(i)\phi(i)\left(\frac{f(i)}{k_2 + f(i)}\right)^2 \quad \textbf{(5.20)}$$

Note that according to Equation (5.19) the probability that an ant picks up an item from the grid is inversely proportional to the amount of pheromone at that position and also to the density of objects around *i*. The reverse holds for Equation (5.20); the probability of an ant dropping an item on the grid is directly proportional to the amount of pheromone at that position and to the density of objects around *i*.

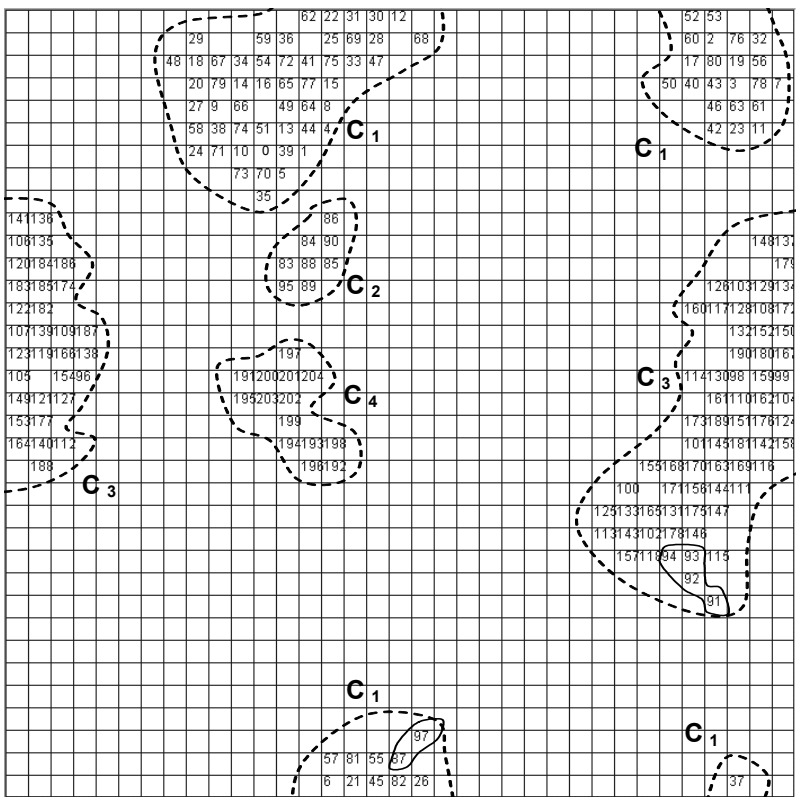


Figure 5.9: Final grid configuration for the bioinformatics data after the algorithm converged.

One of the problems the authors used to evaluate their proposal was in the field of bioinformatics, more specifically, the *yeast galactose* data set composed of 20 attributes (Yeung et al., 2003). Note that although each datum has dimension 20 all of them will be projected onto the bi-dimensional grid of the ACA. The authors used a subset of 205 data items, whose expression patterns or attributes reflect four functional categories (clusters) of genes formed by 83, 15, 93 and 14 genes (data). The authors used 10 ants to run the algorithm, $n_{ants} = 10$, a grid of size 35×35, $\alpha = 1.05$, $\theta = 0.6$, $k_1 = 0.20$, and $k_2 = 0.05$.

Figure 5.9 illustrates one grid configuration after the algorithm converged. Note the presence of four clusters of data (within dotted lines) and some misclassifications made by the algorithm (within solid lines). The numbers indicate the label of each datum in the data set. Note also the toroidal nature of the grid; that is, some clusters cross from one side of the picture to another, both from top to bottom and from left to right.

Scope of Ant Clustering Algorithms

The examples of application and the nature of the ant clustering algorithm suggest its potentiality for clustering problems. In more general terms, ACAs are suitable for exploratory data analysis, in which an unlabelled set of data is available and some information must be extracted from it. For instance, useful information is the degree of similarity between items and how to infer the cluster a new item (unknown to the system) belongs to.

It was discussed that the standard ACA fails in determining the appropriate number of clusters. However, some modifications can be introduced in the standard algorithm so as to make it suitable for determining a more accurate number of clusters in relation to the actual clusters of the input data set.

The ACA performs a dimensionality reduction in the space of attributes of the input data set into a bi-dimensional grid. This may be a useful approach when dealing with spaces of very high dimension, for it allows a visualization of neighborhood (similarity) relationships between the data items in the data set. Nevertheless, it may also cause some loss of information as a consequence of dimensionality reduction.

From Natural to Artificial Ants

Ant clustering algorithms were developed as a generalization of models of dead body clustering and brood sorting in ant colonies. The main features of ant clustering algorithms are a colony of ants that move within a bi-dimensional grid used to project the items (e.g., data and graph vertices) to be clustered. Ants are allowed to move items throughout this grid in a probabilistic manner, which is proportional to the degree of similarity between items. Table 5.2 presents one interpretation of the biological terminology to the computational ant clustering algorithm.

Table 5.2: Interpretation of the biological terminology into the computational ant clustering algorithm (ACA).

Biology (Ant Clustering)	Ant Clustering Algorithm
Environment (Arena)	Bi-dimensional grid in which items are projected and ants are allowed to move
Ant	Agent-like entity capable of moving about, picking up, and depositing items on the grid
Ant colony	Population (colony or swarm) of cooperating individuals known as artificial ants
Dead bodies or larvae	Items or objects (e.g., data patterns and vertices of a graph)
Pile (heap) of dead bodies	Clusters of items
Visibility of an ant	Perceived fraction of items f

5.2.6. Summary of Swarm Systems Based on Social Insects

This section discussed the behavior of social insects, focusing the case of ant colonies. It was observed that several different patterns of behavior can be seen in nature, from foraging for food to cemetery organization. These different behaviors lead to several computational tools and systems with various applications, such as combinatorial optimization and data analysis.

The majority of ant colony optimization algorithms are used to perform some sort of graph search associated with a combinatorial optimization problem. During the operation of an ACO algorithm, a colony of artificial ants is initialized in randomly chosen nodes of the graph G, and is allowed to move through adjacent nodes of this graph. The move of an ant from one node to another follows a probabilistic transition rule that is proportional to the pheromone level of a neighbor edge and the visibility (heuristic desirability) of the ant. By moving, ants incrementally build solutions to the problem and deposit pheromone on the edges traversed. This pheromone trail information will direct the search process of other ants, indicating that a certain route has been taken by (an)other ant(s). Pheromone trails are allowed to evaporate; that is, the level of pheromone of an edge is decreased over time, thus avoiding a rapid convergence towards suboptimal solutions.

The cemetery organization and larval sorting behavior of ants have led to the proposal of a theoretical model capable of reproducing a similar behavior. This model then led to the development of the ant clustering algorithm with potential applicability to exploratory data analysis, among other clustering tasks. The algorithm works by projecting the objects to be clustered into a bi-dimensional grid and probabilistically placing these objects onto the grid according to the density of similar items in a given region of the grid. The main aspects to be defined in this algorithm are the probability of placing or moving an item at or from a given position (site or cell) of the grid, and how to define the density function of items in this area of the grid.

5.3. SWARM ROBOTICS

In *autonomous robotics*, the so-called *swarm-based*, or *collective robotics*, relies on the use of metaphors and inspiration from biological systems, in particular social insects, to the design of distributed control systems for groups of robots (Kube and Bonabeau, 2000). Biological analogies and inspiration are thus pervasive in swarm robotics (Kube et al., 2004).

Well-known collective behaviors of social insects provide striking evidence that systems composed of simple *agents* (ants, bees, wasps, and termites) can accomplish sophisticated tasks in the real world. It is widely held that the cognitive capabilities of these insects are very limited, and that complex behaviors are an emergent property of interactions among the agents, which are themselves obeying simple rules of behavior (Cao et al., 1997). Thus, rather than following the AI tradition of modeling robots as rational, deliberative agents, researchers in swarm robotics have chosen a more bottom-up approach in which individual agents are more like *artificial insects*, in particular *artificial ants* - they follow simple rules and are reactive to environmental and group changes. Therefore, the concept of *self-organization* forms the core of swarm robotics.

Groups of mobile robots are built aimed basically at studying issues such as group architecture, conflict resolution, origin of cooperation, learning, and geometric problems. There has been an increasing interest in systems composed of multiple autonomous mobile robots exhibiting cooperative behavior. The current interest for swarm-based robotics is due to several factors (Cao et al., 1997; Kube and Bonabeau, 2000):

- Some tasks may be inherently too complex (or impossible) for a single robot to accomplish. For instance, moving an item (e.g., a load) that is much heavier than the robot itself.
- Performance benefits can be gained from the use of multiple robots. For example, a task such as garbage collection may be performed faster by a team of robots.
- Building and using simple robots is usually easier, cheaper, more flexible and more fault tolerant than having a single robot with sophisticated sensors, actuators and reasoning capabilities.
- The decrease in price of simple commercial robots, such as the Khepera® (cf. K-Team Web Site, the current enterprise responsible for commercializing the Khepera robots), make them more accessible for research experiments.
- The progress of mobile robotics over the last few years facilitated the implementation and testing of groups (swarms) of robots.
- The field of *Artificial Life* (Chapter 8), where the concept of emergent behavior is emphasized as essential to the understanding of fundamental properties of life, has done much to propagate ideas about collective behavior in biological systems, particularly social insects.

- The constructive, synthetic approach of swarm robotics can yield insights into fundamental problems in social and life sciences (e.g., cognitive psychology and theoretical biology).

Up to the early 1990s, the majority of research projects in robotics were concentrated in designing a single autonomous robot capable of functioning, and usually performing a given task, in a dynamic environment. However, by organizing multiple robots into collections of task achieving populations, useful tasks may be accomplished with fairly simple behavior-based control mechanisms. This is the approach adopted by the swarm-based robotics, in which a swarm of robots cooperate to accomplish a given task.

In swarm robotics, a set of (simple) individual behavior rules is defined as a means to determine how the robots are going to interact with one another and the environment. Task accomplishment is a result of the emergent group level behavior. Therefore, if the behavior of the swarm is an emergent property, how can one evaluate the rules set for the robots as appropriate or not? There is no answer to such question yet, though some insight can be gained from studies in the field of artificial life and research with the swarms of robots. Another difficulty in swarm robotics is that, due to the lack of global knowledge, a group of robots may stagnate or find itself in a deadlock where it can make no progress. Despite these difficulties and the youth of the field, it is still very promising, though some authors believe many potential applications of swarm robotics may require miniaturization (Bonabeau et al., 1999; Kube and Bonabeau, 2000).

There are a large number of researchers studying swarm robotics and a vast literature on the topic. Thus, instead of trying to make a broad survey, the aim of this section is to describe a selected sample of real-world implementations of swarm robotics. The focus will be on the type of collective behavior modeled, and on the main behavioral rules and environmental set ups used. No detail will be provided regarding technical, implementation or architectural issues. The examples of collective behavior to be described here were inspired basically by the following natural behaviors of ants:

- Foraging for food.
- Clustering of dead bodies.
- Clustering around food (recruitment).
- Collective prey retrieval.

5.3.1. Foraging for Food

Krieger et al. (2000) and Krieger and Billeter (2000) showed that an ant-inspired system for task allocation based on variation in individual stimulus thresholds provides a simple, robust and efficient way to regulate activities of a team of robots. In this case, the task was to collect food items when robots have no initial information about the environment and the total number of robots in the colony.

The essential components of social organization governing ant colonies were used to program swarms of robots foraging from a central nest. The authors chose foraging as the task assigned to the robots due to several reasons: 1) foraging is the task for which mechanisms of task allocation are best understood; 2) foraging efficiency is a key factor influencing colony productivity; and 3) foraging is the task that has received the greatest attention in cooperative robotics.

Given a colony of robots with one central nest and a set of "food items", all within a closed arena, the task the robots have to accomplish corresponds to collecting food so as to maintain a minimal energy level for the colony. The experimental setup and behavior rules for the robots can be summarized as follows:

1. Colony-level information about the energy of the colony is assessed and updated by each individual robot.
2. While in the nest, the robots receive information about the colony energy level via radio messages from a control station.
3. Upon return to the nest, each robot renews its energy reserves, resulting in a decrease in colony energy, and unload (if loaded) the food item collected in a basket, thereby increasing the energy of the colony.
4. The robots were programmed so as to avoid one another in order to minimize collisions and negative interactions.

Individual variation in the tendency to perform a task was also implemented in the robots, thus modeling the different individual stimulus thresholds for task allocation observed in ants and bees.

5. Therefore, robots do not leave the nest simultaneously, but only when the colony energy drops below a pre-defined threshold of a particular robot.
6. Finally, for some trials, robots were programmed to recruit another robot when they identify an area rich in food items, thus mimicking the tandem recruitment behavior observed in many ant species (Section 5.2.2).

In summary, the robots have generalized information about the energy level of the colony, ways to avoid interfering with another robot in space and time, and, in some experiments, the ability to recruit other robots to collect food. Their objective was to maintain the energy of the colony above a minimum threshold level.

The authors used Khepera miniature robots, incremented with a gripper turret, a custom made detection module and a radio turret, designed as a research tool at the *Laboratoire de Micro-Informatique* of the Swiss Federal Institute of Technology at Lausanne, Switzerland. A video of their experiment can be found at http://www.nature.com/nature/journal/v406/n6799/extref/406992ai1.mov. Further information is available at the K-Team and Nature web sites, and at the Collective Robotics Group web site: http://diwww.epfl.ch/w3lami/collective/. Figure 5.10 presents some screen shots from their movie.

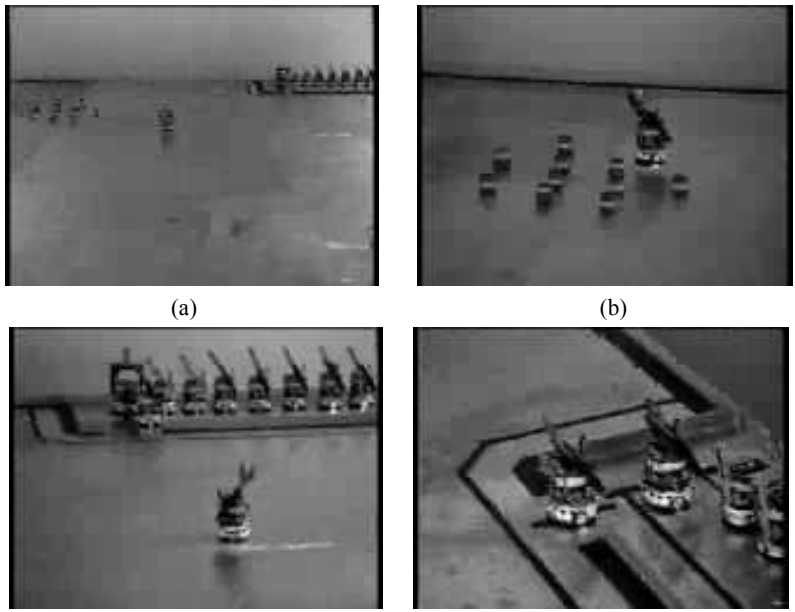


Figure 5.10: Screen shots from the movie by Krieger et al. (2000). (a) A robot leaves the nest at the top right corner and moves towards the food items. (b) The robot collects a food item and returns to the nest (c). The robot recruits another robot to collect food (tandem recruitment). (Reproduced with permission from [Krieger et al., 2000], © M. B. Krieger, J.-B. Billeter and L. Keller.)

The experiments performed showed that giving different but individually fixed activation levels to the robots results in an efficient dynamical task allocation, with great flexibility with regard to team (colony) size. Furthermore, relatively complex tasks can be performed by relatively simple and autonomous robots regulated in a decentralized way. It was shown that this strategy allows a swarm of robots to continue to work efficiently in case the number of robots is reduced by individual failures. Finally, it was shown that tandem recruitment of robots in an ant-like manner is an efficient strategy for exploiting clustered resources.

Note that there is a slight difference between tandem recruitment and the mass recruitment that gave rise to the ant colony optimization algorithm. In the former the recruitment is a result of direct contact between ants, while in the latter the interaction occurs indirectly via pheromone trails.

5.3.2. Clustering of Objects

Beckers et al. (1994) developed a system using multiple robots to gather together a dispersed set of objects (pucks) into a single cluster within an arena,

much like the corpse-gathering behavior of ants discussed in Section 5.2.4. A robot was designed so that it could move small numbers of objects, and such that the probability of an object to be left (deposited) in a given position of the arena was directly proportional to the number of objects in that region. This was accomplished by effectively sensing a very local density of objects via a simple threshold mechanism. The plan was to evaluate the performance of the robots with this mechanism and to develop the strategy and the behaviors required until the task could be performed reliably.

In their simulations, the robots were equipped with a C-shaped gripper to gather objects instead of a clipper as used in the works of Krieger et al. (2000). The gripper was equipped with a microswitch activated by the presence of three or more pucks within the gripper. Figure 5.11 illustrates the type of robot with a gripper used in the experiments performed by Beckers et al. (1994).

In their experiments, the robots have only three behaviors, and only one is active at any time:

1. When no sensor is activated, a robot executes the default behavior of moving in a straight line until an obstacle is detected or until a microswitch is activated.
2. On detecting an obstacle, the robot executes an obstacle avoidance behavior by turning on the spot away from the obstacle with a random angle. The default behavior takes over again and the robot moves in a straight line in the new direction. If the robot is pushing some of the pucks (spread over the arena) when it encounters an obstacle, the pucks will be retained by the gripper during the turn.
3. When the gripper pushes three or more pucks, the microswitch is activated triggering the puck dropping behavior, which consists of backing up by reversing both motors releasing the pucks from the gripper. It then executes a turn with a random angle and returns to the default behavior.

As the robots behave autonomously, with all sensors, motors, and control circuits independent of one another, there is no explicit communication between robots. Therefore, the resulting behavior of the swarm of robots is an emergent property achieved through stigmergic self-organization based upon the reaction of the robots to the local configuration of the environment (arena).

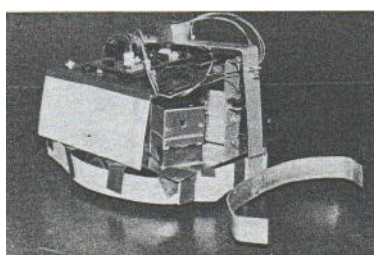


Figure 5.11: Robot equipped with a gripper to gather objects (pucks). (Reproduced with permission from [Beckers et al., 1994], © MIT Press.)

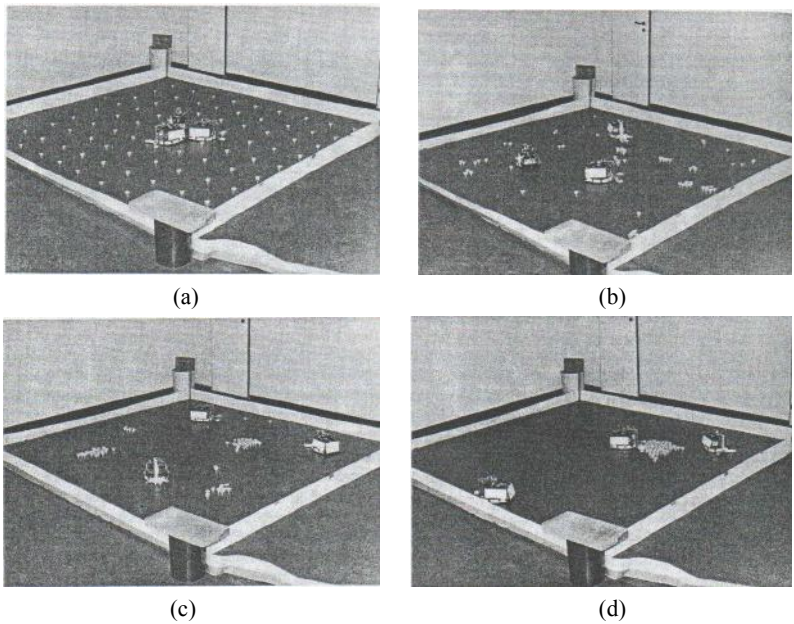


Figure 5.12: Type of behavior obtained in one of the experiments performed by Beckers et al. (1994). (a) Initial configuration of the environment. (b), (c) Evolution of the experiment after some time. (d) Final clustering of the pucks. (Reproduced with permission from [Beckers et al., 1994], © MIT Press.)

Figure 5.12 summarizes the type of behavior obtained in their experiments. Initially the pucks are distributed all over the arena (Figure 5.12(a)). Qualitatively, the robots initially move forward gathering pucks into the gripper one at a time, and dropping them in case three or more have been gathered. After a short period of time, a number of small clusters will have been formed (Figure 5.12(b)). By striking the clusters with a small angle, the robots remove some pucks from the clusters, resulting in the formation of larger clusters (Figure 5.12(c)). Finally, one single large cluster remains (Figure 5.12(d)).

The experimental results presented demonstrate that the simple behavioral rules allow the control and coordination of a number of robots without direct communication. This is a real-world evidence that stigmergic behaviors can be implemented in a swarm of robots so that they carry out a predefined task. The authors also suggested that this approach to multiple robotic systems is robust to individual robots' failures. This is mainly because each robot is seen as an individual agent whose rules of behavior are independent of the others' rules. The authors pointed out that such swarm robotic system has as advantage a gain in the speed of accomplishing some tasks because the addition of more robots does not require any reconfiguration of the system.

5.3.3. Collective Prey Retrieval

Several species of ants are capable of collectively retrieving preys so large that they could not possibly be retrieved by a single ant. If a single ant finds a prey and is capable of moving it alone, it takes the prey to the nest. Otherwise, the ant recruits nestmates via tandem recruitment (direct contact) or pheromone laying. When even a group of ants is not capable of moving a large prey item, specialized workers with large mandibles may be recruited in some species to cut the prey into smaller pieces. Such phenomena of large prey retrieval by ant colonies, though broadly studied, have no formal description yet (Bonabeau et al., 1999; Kube and Bonabeau, 2000).

There are several interesting aspects in the collective prey retrieval of ants. Some of these aspects have been explored for the development of behavioral rules for swarm robotic systems. One interesting aspect, agreed upon by most researchers (e.g., Moffett, 1988; Franks, 1989), is that collective transport is more efficient than solitary transport for large prey; i.e., the total weight that can be carried by a group of ants increases nonlinearly with the group size.

Another interesting aspect is that resistance to transport seems to be the reason why a food item is carried or dragged, and also if there will be more than one ant involved in the prey retrieval process. If a single ant tries to carry the item and fails, more ants may be recruited to help. The ant spends some time testing the resistance of the item to dragging before *realigning* the orientation of its body without releasing the item. In case realignment is not sufficient, the ant releases the item and finds another position to grasp it. If several of these *repositioning* attempts are not sufficient to result in a movement of the item, then the ant eventually recruits other nestmates (Sudd, 1960, 1965).

It is believed that the number of ants involved in prey retrieval is a function of the difficulty encountered in first moving the prey, not only of prey weight. A prey item that resists motion stimulates the recruitment of more ants. Success in carrying a prey item in one direction is followed by another attempt in the same direction. Finally, recruitment ceases as soon as a group of ants is capable of carrying the prey in a well-defined direction. In such situation the group size is adapted to the prey size.

Therefore, if a certain prey item cannot be moved, ants exhibit realigning and repositioning behaviors, and ultimately more ants are recruited to help. Only when realignment is not possible, repositioning is performed (Sudd, 1960, 1965). Coordination in collective transport thus seem to occur through the item being transported: a movement of one ant engaged in group transport is likely to modify the stimuli perceived by the other group members, possibly producing, in turn, orientational or positional changes in these ants (Bonabeau et al., 1999). This is another clear example of stigmergy.

Cooperative Box Pushing

Inspired by these (and other) observations of how ants cooperate in collective prey transport, Kube and Zhang (1992, 1994a,b), Kube and Bonabeau (2000),

and Kube et al. (2004) have consistently studied the problem of collective prey retrieval in ants with a view of implementing a robotic system to perform a task known as *cooperative box-pushing*. This is equivalent to cooperative prey retrieval in ants. They have initially introduced a simulation model for the robots, and lately implemented a physical system with a group of homogeneous (identical) robots.

The initial task under study was undirected box-pushing, in which a group of robots found a box and pushed it in a direction that was dependent upon the initial configuration. The next task studied was a directed box-pushing, in which the robots were supposed to push the box toward a predefined goal position. Finally, the transport task, which is a variation of the directed box-pushing with multiple sequenced goals, was studied and the robots had to push the box from one location to the next (Kube and Bonabeau, 2000).

Five hierarchically organized behaviors were defined for the robotic experiments of (Kube and Zhang, 1992, 1994a,b). Behavior 1 has a higher priority than behavior 2, and so forth:

1. An avoid behavior to prevent collisions.
2. A goal behavior to direct the robot towards the goal.
3. A slow behavior to reduce the robots' speed.
4. A follow behavior to direct the robot towards its nearest neighbor.
5. A find behavior that allows for exploration.

These hierarchical behaviors were used to simulate the simple three behavioral rules for the robots:

1. A single robot (ant) tries to drag or carry the object (food item) with numerous realignment and repositioning strategies.
2. If unsuccessful, the robot (ant) will recruit other robots (ants) to assist in the task.
3. The group of robots (ants) will collectively and cooperatively try to move the object (food item) by realigning and repositioning themselves until a satisfactory configuration of the robots leads to a motion in the object.

The robotic implementation presented by Kube and Zhang (1992, 1994, 1997) demonstrated that a group of simple robots is capable of cooperatively performing a task without any direct communication if the robots have a common goal and try to minimize collisions. In the undirected box-pushing task, a lit object was placed on the arena and had to be pushed towards any of its ends (see Figure 5.13). The group of robots eventually moves the lit box in a direction dependent on the initial configuration of the system. Both execution time and average success rates increased with group size up to a point, when interference among robots starts to play a role.

One problem identified with their initial approach was *stagnation*, when no improvement in task accomplishing can be made. To tackle this problem, Kube and Zhang (1994b) implemented the mechanisms of realignment and repositioning, known as stagnation recovery behaviors, inspired by the ants. In this case,

(a) (b)

Figure 5.13: Transport experiments performed by Kube and Zhang (1997). (a) Robots grouping around the box and pushing it toward the right corner of the arena (b). Robots pushing the box to the left side of the arena. (Reproduced with permission from [Kube and Zhang, 1997], © C. R. Kube and H. Zhang.)

each robot exerts a force on the box side at a certain angle, thus producing a resultant force and torque applied to the center of the box. If the total resultant force is greater than a threshold, the box translates in the plane. Else, if the total torque exceeds a torque threshold, the box rotates. Stagnation here refers to any situation where a robot is in contact with the box and it is not moving. The experimental results demonstrated that the application of random pushing motions by either realigning the pushing angle or repositioning the pushing force is an effective technique against stagnation.

Some of their experiments were recorded and can be downloaded from Kube's web site at http://www.cs.ualberta.ca/~kube. In the videos, the realignment and repositioning of the robots can be clearly observed. The path used to take the lit object from its original position to the lit corner of the arena (corresponding to the nest or goal) is not optimal and may vary from one experiment to another. However, the robotic system described gives some clues about cooperative transport in ants and task accomplishment without direct communication. At a fundamental level, as their model reproduces some of the collective features of cooperative prey retrieval in ants, it suggests that these assumptions serve the purpose of explaining similar behaviors in ants. Actually, it is the first formalized model of cooperative transport in ants (Bonabeau et al., 1999; Kube and Bonabeau, 2000). Figure 5.13 presents two screen shots of their video recordings depicting the experimental environment and task achievement (box pushed towards the lit corner on the left hand side of the arena).

Recruitment of Nestmates

One of the problems involved in large prey retrieval is the recruitment of ants around the prey. In the context of collective prey retrieval, Hölldobler and Wilson (1978) and Höldobler (1983) have suggested two categories of recruitment via chemical cues: short-range and long-range recruitment. In the former catego-

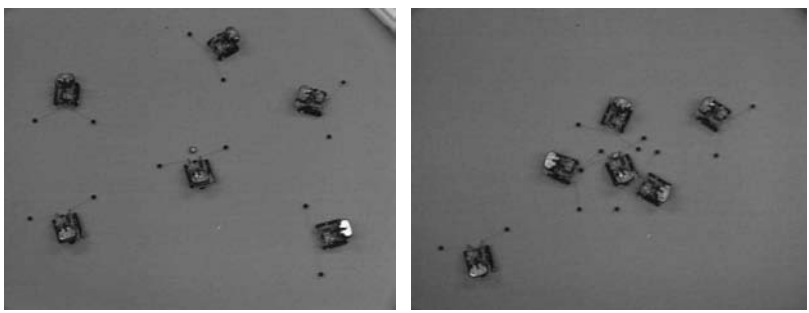
ry - short-range recruitment - when a large prey item is found by an ant it secretes some chemical substance in its vicinity to attract other ants. This is equivalent to the group recruitment described previously. In the later category - long-range recruitment - the ant that finds a large prey item secretes some pheromone making a trail connecting the food item to the nest. This is equivalent to the mass recruitment discussed in Section 5.2.2.

Inspired by the collective behaviors of ants, researchers at the MIT Artificial Intelligence Lab were interested in developing a community of cubic-inch microrobots to simulate what they termed AntFarm. There were two main goals for this project. The first was to try to integrate several sensors and actuators into a small package for the microrobots. The second was to form a structured robotic community from the interactions of many simple individuals whose communication was performed using infrared (IR) emitters.

One of the behaviors of ants they wanted to include in the AntFarm project was that of recruitment of ants around a food item. The robots are initially scattered within a given arena and a food item is placed on it. Then, the task to be performed in this part of their system was that of finding the item and recruiting other ants for retrieval. Figure 5.14 presents two screen shots from the movie available for their experiments (from http://www.ai.mit.edu/projects/ants/). In Figure 5.14(a) the food item is initially placed in a given position of the arena. The ants are recruited and cluster around the food item (Figure 5.14(b)).

Three basic rules were used for recruitment:

1. Once a robot detects food, it emits an “I found food” IR signal. Any robot within about 12 inches of it can detect the signal and head towards it.
2. When a robot receives the “I found food” signal, it heads towards the robot with the food while transmitting “I see an Ant with food”.
3. Any robot within range of the second robot receives the “I see an Ant with food” signal, heads towards the second robot, and transmits “I see an



(a) (b)

Figure 5.14: Recruiting of ants around the food item. (a) The food item is placed in the center of the arena. (b) Ants are recruited to the site around the food item. (Reproduced with permission from [http://www.ai.mit.edu/projects/ants/social-behavior.html], © MIT Computer Science and Artificial Intelligence Laboratory.)

Ant that sees an Ant with food", and so on.

It can be observed by the rules above that this system embodies the idea of short-range recruitment proposed by Höldobler and Wilson (1978). The communication among ants, in this case, is mediated by an IR signal that can be likened to the chemical released in the air when an ant finds a large food item it cannot carry by itself.

5.3.4. Scope of Swarm Robotics

As the production cost of integrated circuits and robots decreases, it is more and more tempting to consider applications involving teams of simple robots rather than a single complex robot. Several authors in swarm robotics have suggested that the potential applications of swarm-based robotics require miniaturization (Bonabeau et al., 1999; Kube and Bonabeau, 2000). Very small robots, micro- and nano-robots, which will, by construction, have limited sensing and computing capabilities, will have to operate in large groups or swarms in order to perform useful tasks.

Défago (2001) identified four broad application areas in which swarm-robots, more specifically micro- and nano-robotics, is already being used or could potentially be used:

- *Exploration*: the unmanned exploration of inaccessible environments, such as the space, deep sea, plumbing, etc., is one application for cooperative and nano-robotics.
- *Manufacturing*: the manufacturing of human-scale objects by nanofactories, also called assemblers, is unlikely to be achieved by a single assembler. This will almost certainly require very large teams of assemblers to cooperate.
- *Medicine and surgery*: there are several medical applications using colonies of nano-robots, such as micro-surgery and the control of the precise location and the exact amount of drugs delivered in the body. Some researchers are already developing nano-robots made of a combination of organic and inorganic components (e.g., Bachand and Montemagno, 2000).
- *Smart materials*: these are materials made of a multitude of tiny modules linked to one another forming a large structure capable of autonomously changing its shape and physical properties.

5.3.5. Summary of Swarm Robotics

This section has briefly described some works from the literature that drew inspiration from the behavior of ants in order to design cooperative teams of robots capable of performing particular tasks. The ant-inspired tasks discussed include foraging for food, clustering of dead bodies and larval sorting, and collective prey retrieval.

From an architectural point of view, there are three main advantages of the swarm-robotics approach (Martinoli, 2001). First, the resulting collective systems are scalable, because the control architecture is exactly the same from a few to thousands of robots. Second, such systems are flexible, because individual robots can be dynamically added or removed without explicit reorganization by the operator. Third, these systems are robust, not only due to robot redundancy but also due to the minimalist robot design.

The approaches derived or inspired by social insects demonstrated to be feasible and interesting, though they are not the single alternative for the control and coordination of groups of small robots. The reader may be disappointed by the great simplicity of the tasks reported here as the state of the art in swarm robotics. Besides, collective robotics, including swarm-robotics, still lacks a rigorous theoretical foundation. In spite of these apparent drawbacks, it must be kept in mind that swarm robotics may still be one of the most useful strategies for collective robotics in the light of miniaturization; that is, micro- and nano-robots, though much more has to be done than has actually already been accomplished.

5.4. SOCIAL ADAPTATION OF KNOWLEDGE

The two previously described sections focused on how the social behavior of insect societies led to the development of computational algorithms for problem solving and strategies for the coordination of collective robotics. The *particle swarm* (PS) algorithm, to be discussed in this section, has as one of its motivations to create a simulation of human social behavior; that is, the ability of human societies to process knowledge (Kennedy and Eberhart, 1995; Kennedy, 1997; Kennedy, 2004). As is characteristic of all swarm intelligence approaches (actually, most biologically inspired algorithms), PS also takes into account a population of individuals capable of interacting with the environment and one another, in particular some of its neighbors. Thus, population level behaviors will emerge from individual interactions. Although the original approach has also been inspired by particle systems (Chapter 7) and the collective behavior of some animal societies, the main focus of the algorithm is on its social adaptation of knowledge; the same focus taken here.

A very simple sociocognitive theory underlies the particle swarm approach (Kennedy et al., 2001; Kennedy, 2004). The authors theorize that the process of cultural adaptation comprises a low-level component corresponding to the actual behavior of individuals, and a high-level component in the formation of patterns across individuals. Therefore, each individual within a population has its own experience and they know how good it is and, as social beings, they also have some knowledge of how other neighboring individuals have performed.

These two types of information correspond to *individual learning* and *cultural or social transmission*, respectively. The probability that a given individual takes a certain decision is a function of how successful this decision was to him/her in the past. The decision is also affected by social influences, though the exact rules in human societies are not very clear (Kennedy, 2004; Kennedy et

al., 2001). In the particle swarm proposal, individuals tend to be influenced by the best successes of anyone they are connected to, i.e., the members of their social (sociometric) neighborhood that have had the most success so far.

Kennedy (1998) and Kennedy et al. (2001) use three principles to summarize the process of cultural adaptation. These principles may be combined so as to enable individuals to adapt to complex environmental challenges:

- *Evaluate*: the capability of individuals to sense the environment allows them to quantify their degree of *goodness* (quality or suitability) in relation to some parameter(s) or task(s), such as height when one wants to become a basketball player.
- *Compare*: people usually use others as standards for assessing themselves, what may serve as a kind of motivation to learn and change. For instance, we look at other people and evaluate our wealth, look, humor, beauty, exam mark, whom to vote, and so forth.
- *Imitate*: human imitation comprises taking the perspective of someone else, not only by imitating a behavior but by realizing its purpose and executing the behavior of others when it is appropriate. Imitation is central to human sociality and the acquisition and maintenance of mental abilities.

5.4.1. Particle Swarm

In the particle swarm (PS) algorithm, individuals searching for solutions to a given problem learn from their own past experience and from the experiences of others. Individuals evaluate themselves, compare to their neighbors and imitate only those neighbors who are superior to themselves. Therefore, individuals are able to evaluate, compare and imitate a number of possible situations the environment offers them.

Although there are two basic versions of the PS algorithm, binary and real-valued, this section focuses only on the latter one, namely, the real-valued or continuous PS. This is mainly because this version has a much broader applicability and it has been more extensively studied as well. Under this perspective, the PS algorithm can be viewed as a numeric optimization procedure inspired by the social adaptation of knowledge discussed above. It searches for optima in an L-dimensional real-valued space, $\Re^L$. Therefore, the discussion regarding problem solving as a search in a search space presented in Section 3.2 is also appropriate for the purposes of this section.

In real-valued spaces, the variables of a function to be optimized can be conceptualized as a vector that corresponds to a point in a multidimensional search space. Multiple individuals can thus be plotted within a single set of coordinates, where a number of individuals will correspond to a set of points or particles in the space. Individuals similar to one another in relevant features appear closer to one another in the space, as illustrated in Figure 5.15.

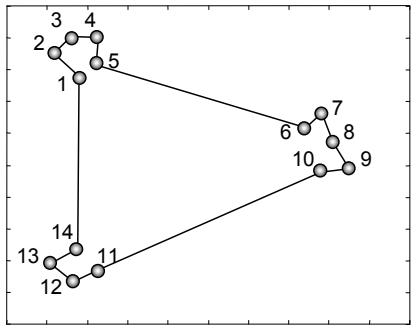


Figure 5.15: Particles in a bi-dimensional real-valued search-space. Particles are connected to their topological neighbors, and neighbors tend to cluster in the same regions of the space.

The individuals are viewed as points in a search-space and their change over time is represented as movements. In this case, individuals truly correspond to *particles*. Forgetting and learning are seen as a decrease or increase on some dimensions, attitude changes are seen as movements between the negative and positive ends of an axis, and emotion and mood changes of numerous individuals can be plotted conceptually in a coordinate system. The swarm of particles corresponds to the multiple individuals in the population.

One insight from social psychology is that these particles will tend to move toward one another and to influence one another as individuals seek agreement with their neighbors (Kennedy et al., 2001; Kennedy, 2004). Another insight is that the space in which the particles move is heterogeneous with respect to evaluation; that is, some regions are better than others, meaning that they have a higher goodness value.

In mathematical terms, the PS algorithm can be implemented as follows. The position of a particle i is given by $\mathbf{x}_i$, which is an L-dimensional vector in $\Re^L$. The change of position of a particle is denoted by $\Delta\mathbf{x}_i$, which is a vector that is added to the position coordinates in order to move the particle from one iteration t to the other $t + 1$:

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \Delta\mathbf{x}_i(t+1) \qquad \textbf{(5.21)}$$

The vector $\Delta\mathbf{x}_i$ is commonly referred to as the velocity $\mathbf{v}_i$ of the particle. The particle swarm algorithm samples the search-space by modifying the velocity of each particle. The question that remains is how to define the velocity of the particle so that it moves in the appropriate directions and with the appropriate "step size" in the search-space. In addition, the neighborhood of each particle (individual) has to be defined.

The inspiration taken from the social-psychological sciences suggests that individuals (particles) should be influenced by their own previous experience and

the experience of its neighbors. Neighborhood here does not necessarily mean similarity in the parameter space; instead, it refers to topological similarity in a given structure of the population. In a social network, a given individual particle is capable of influencing the ones to which it is socially connected. Neighborhood is thus defined for each individual particle based on its position in a topological array, usually implemented as a ring structure (the last member is a neighbor of the first one).

There are a number of different schemes to connect the individuals of the population. Most particle swarm implementations use one of two simple sociometric principles. The first, called *gbest* (*g* for global), conceptually connects all members of the population to one another (Figure 5.16(a)). The effect of this is that each particle is influenced by the very best performance of any member of the entire population. The second, termed *lbest* (*l* for local), creates a neighborhood for each individual comprising itself and its *k*-nearest neighbors in the population (Figure 5.16(a)). To illustrate, consider the particles in Figure 5.15: each individual particle is connected with its 2-nearest neighbors in an *lbest* scheme ($k = 2$). Thus, particle $\mathbf{x}_3$ has neighbors $\mathbf{x}_2$ and $\mathbf{x}_4$, particle $\mathbf{x}_8$ has neighbors $\mathbf{x}_7$ and $\mathbf{x}_9$, and so on.

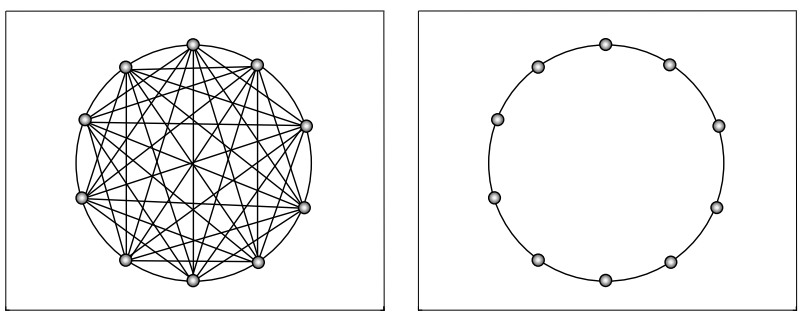
A particle will move in a certain direction as a function of its current position $\mathbf{x}_i(t)$, its change in position $\Delta\mathbf{x}_i(t)$, the location of the particle's best success so far $\mathbf{p}_i$, and the best position found by any member of its neighborhood $\mathbf{p}_g$:

$$\mathbf{x}_i(t+1) = f(\mathbf{x}_i(t), \Delta\mathbf{x}_i(t), \mathbf{p}_i, \mathbf{p}_g) \qquad \textbf{(5.22)}$$

The influence of the terms $\Delta\mathbf{x}_i(t)$, $\mathbf{p}_i$, and $\mathbf{p}_g$ can be summarized by a change $\Delta\mathbf{x}_i(t+1)$ to be applied at iteration $t+1$:

$$\Delta\mathbf{x}_i(t+1) = \Delta\mathbf{x}_i(t) + \boldsymbol{\varphi}_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \boldsymbol{\varphi}_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t)) \qquad \textbf{(5.23)}$$

where $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ represent positive random vectors composed of numbers drawn from uniform distributions with a predefined upper limit: $\boldsymbol{\varphi}_1 = U(0, AC_1)$ and $\boldsymbol{\varphi}_2 = U(0, AC_2)$; $U(0, AC)$ is a vector composed of uniformly distributed random numbers, and AC is called the *acceleration constant*.



(a) (b)

Figure 5.16: Illustration of *gbest* and *lbest* neighborhoods. (a) *gbest*. (b) *lbest* for $k = 2$.

According to Kennedy (2004), the sum of the two acceleration constants should equal 4.1, $AC_1 + AC_2 = 4.1$ (usually both are 2.05). The second term on the right hand side of Equation (5.23) is proportional to the difference between the particle's previous best and its current position, and the last term on the right hand side of Equation (5.23) is proportional to the difference between the neighborhood's best and the current position of the particle. The symbol $\otimes$ represents the elementwise vector multiplication.

In (Kennedy, 1997), the author clearly identifies the meaning of each term in Equation (5.23) and performs a number of experiments, with a single application problem, in order to evaluate the importance of "social" and "cognitive" interactions in the PS algorithm. He defines Equation (5.23) without the last term as the "cognition-only" model, and without the second term, but with all the others, as the "social-only" model.

In order to limit the change in position of a particle so that the system does not "explode", two values $v_{\min}$ and $v_{\max}$ are defined for the change $\Delta\mathbf{x}$, thus guaranteeing that the particles oscillate within some predefined boundaries:

If $\Delta x_{id} > v_{\max}$, then $\Delta x_{id} = v_{\max}$,

Else if $\Delta x_{id} < v_{\min}$ then $\Delta x_{id} = v_{\min}$

```
procedure [X] = PS(max_it, AC1, AC2, vmax, vmin)
    initialize X   //usually xi, ∀i, is initialized at random
    initialize Δxi //at random, Δxi ∈ [vmin, vmax]
    t ← 1
    while t < max_it do,
        for i = 1 to N do,          //for each particle
            if g(xi) > g(pi),
                then pi = xi,       //best indiv. performance
            end if
            g = i                   //arbitrary
            //for all neighbors
            for j = indexes of neighbors
                if g(pj) > g(pg),
                    then g = j,     //index of best neighbor
                end if
            end for
            Δxi ← Δxi + φ1⊗(pi − xi) + φ2⊗(pg − xi)
            Δxi ∈ [vmin, vmax]
            xi ← xi + Δxi
        end for
        t ← t + 1
    end while
end procedure
```

Algorithm 5.4: Standard particle swarm optimization (PS) algorithm.

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be the swarm of particles, $g(\mathbf{x}_i)$ the goodness of particle $\mathbf{x}_i$, $\Delta\mathbf{x}_i$ its change in position, $v_{\min}$ and $v_{\max}$ the lower and upper limit for the change in position (velocity), respectively, and L the dimension of the particles. Assume a maximal number of iterations `max_it` to be run and a swarm of a given size N (according to the authors, the swarm size is usually $N \in [10,50]$). The original PS algorithm is described in Algorithm 5.4. In the section on applications (Section 5.4.2), two simple modifications of the algorithm that considerably improve performance will be introduced.

Note that solving a problem using the PS algorithm involves the same three aspects suggested in the previous chapter: the choice of a representation (in this case it is fixed: real-valued vectors), the choice of an objective function, and the choice of a goodness function according to the desired objective.

5.4.2. Selected Applications from the Literature: A Brief Description

Algorithm 5.4 presents the original particle swarm optimization algorithm developed taking inspiration from the social adaptation of knowledge and bird flocking behavior. The description presented here emphasized the social behavior of human beings as the primary motivation for the algorithm because this is the emphasis given in Kennedy et al. (2001) and Kennedy (2004), two of the most important fundamental works of the field.

To illustrate in what types of problem the algorithm has been applied to, two applications will be presented. The first example - optimization of neural network weights - illustrates the importance of the PS algorithm as a tool to be hybridized with other strategies. The second example - numerical function optimization - presents two slight variations of the original algorithm and discusses its potentiality to solve multi-dimensional function optimization problems in general.

Optimization of Neural Network Weights

One of the first applications of the particle swarm algorithm was to the problem of defining appropriate weights for artificial neural networks (Kennedy and Eberhart, 1995). It has been discussed in the previous chapter that a neural network is comprised of a structured set of nodes partially or fully connected. The neural network performs a mapping from a set of input data into one or more output node(s). Each network node is a processing unit that receives stimuli from the environment or (an)other node(s), processes these inputs and produces an output signal. Assuming that the nodes in the network are homogeneous; that is, of same (predefined) type, and the network structure is of a fixed given size, the resultant network learning task is reduced to the problem of determining an appropriate weight set that will lead to a satisfactory network behavior when given input stimuli are presented. Figure 5.17 illustrates a simple network with two input nodes, one output node, and two intermediary nodes between the input and output nodes.

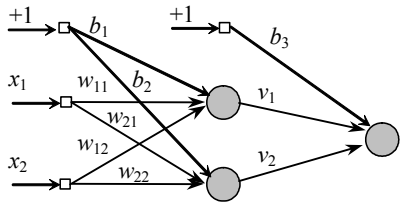


Figure 5.17: An example of a simple artificial neural network with two input nodes, two intermediate or hidden nodes, and one output node.

Inputs		Output
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

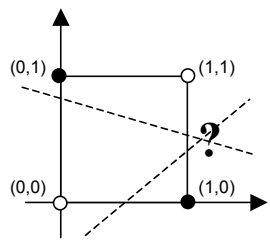


Figure 5.18: The XOR function and its graphical representation. The black dot corresponds to 1, while the circles are equivalent to 0.

Each connection between the nodes has an assigned weight w_{ij}, v_i, and b_i, where w and v correspond to weights of different layers in the network, and b_i corresponds to the bias of a node. Usually, the weight vectors of this type of network are determined according to a learning rule or algorithm, responsible for updating the network weights along an iterative procedure of adaptation. In the present example, a PS algorithm will be used to determine the network weights instead of a standard learning algorithm.

Consider the problem of using the neural network of Figure 5.17 to determine a mapping capable of simulating the exclusive-or (XOR) logic function depicted in Figure 5.18. The difficulty in solving this problem, seen as a classification problem, resides in the fact that it is not possible to separate the input patterns that lead to an output 1 from the input patterns that result in an output 0 by simply drawing a line or a plane on the graph. That means that the two classes available (0 or 1) are not linearly separable. This is a classic benchmark problem for the network structure presented in Figure 5.17, though nowadays there are various algorithms that solve this problem very easily.

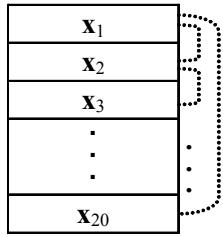


Figure 5.19: The structured neighborhood used in the PS implementation to define neural network weights. Particle $\mathbf{x}_1$ is connected to $\mathbf{x}_2$ and $\mathbf{x}_{20}$, particle $\mathbf{x}_2$ is connected to $\mathbf{x}_1$ and $\mathbf{x}_3$, and so forth. This is a circular neighborhood such as the one depicted in Figure 5.16(b).

In the experimental results reported in (Kennedy and Eberhart, 1995; Kennedy, 1997; Kennedy et al., 2001), the authors used a swarm composed of 20 particles ($N = 20$) initialized at random. Each particle has a dimension $L = 9$ corresponding to the nine weights of the network (w_{11},w_{12},w_{21},w_{22},b_1,b_2,b_3,v_1,v_2) to be determined. The swarm of particles can be defined by using a matrix $\mathbf{X}$ with 20 rows and 9 columns, $\mathbf{X} \in \Re^{20\times 9}$, where each row corresponds to a particle of dimension $L = 9$.

The neighborhood is of the 2-nearest neighbor type, as illustrated in Figure 5.19. The goodness of each particle is defined as the error (e.g., the mean squared error, MSE) between the current network output and the desired outputs given in Figure 5.18. The goal is thus to determine an appropriate weight set that minimizes the mean squared error between the network output for all input patterns and the desired output.

As each particle corresponds to one of the parameters to be adjusted in the neural network, the movements of the particles in the search space allow for a selection of particles that provide the desired network performance. In their simulations the desired criterion was $MSE < 0.02$. Experimental results were very satisfactory, and presented for different values of v_{max}.

Numerical Function Optimization

Similarly to the application of evolutionary algorithms, Angeline (1998), and Shi and Eberhart (1999) applied the PS algorithm to perform numeric function optimization. The authors used some standard test functions (e.g., Sphere, Rosenbrock, Rastrigin, Griewank) taken from the literature of evolutionary algorithms to assess the performance of the PS algorithm:

Sphere $$f_0(x) = \sum_{i=1}^{L} x_i^2$$

Rosenbrock $$f_1(x) = \sum_{i=1}^{L} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$$

Rastrigin $$f_2(x) = \sum_{i=1}^{L} \left(x_i^2 - 10.\cos(2\pi x_i) + 10\right)$$

Griewank $$f_3(x) = \frac{1}{4000}\sum_{i=1}^{L} x_i^2 - \prod_{i=1}^{L} \mathbf{c}os\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

In (Shi and Eberhart, 1999), all functions were implemented using $L = 10$, 20 and 30 dimensions, and the PS algorithm was run for 1000, 1500, and 2000 iterations, respectively. Also, the authors studied different population sizes for the PS, $N = 20$, 40, 80 and 160. In this application, each particle corresponds to one candidate solution to the problem, in a form similar to that used in genetic algorithms, but with real-valued particles instead of a binary representation. They suggested that by using a decreasing factor termed *inertia weight* w the PS algorithm was able to better balance exploration with exploitation:

$$\Delta\mathbf{x}_i(t+1) = w.\ \Delta\mathbf{x}_i(t) + \boldsymbol{\varphi}_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \boldsymbol{\varphi}_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t)) \qquad \textbf{(5.24)}$$

where $\boldsymbol{\varphi}_1$ and $\boldsymbol{\varphi}_2$ were defined as previously and w was assigned an initial value but was allowed to be reduced (or cooled, such as the temperature in simulated annealing) during the iterative procedure of adaptation. The decreasing inertia weight started at 0.9 and was linearly decreased until $w = 0.4$. The limiting values for $\Delta\mathbf{x}$ were set equal to v_{max} according to a pre-defined table.

Another variant of the original algorithm that is currently known as the standard particle swarm algorithm (Kennedy, 2004) involves the use of a global constriction coefficient χ (Clerc and Kennedy, 2002) as follows:

$$\Delta\mathbf{x}_i(t+1) = \chi\ (\Delta\mathbf{x}_i(t) + \boldsymbol{\varphi}_1 \otimes (\mathbf{p}_i - \mathbf{x}_i(t)) + \boldsymbol{\varphi}_2 \otimes (\mathbf{p}_g - \mathbf{x}_i(t))) \qquad \textbf{(5.25)}$$

where the parameter $\chi \approx 0.729$.

There are some java applets that show the movement of particles when applied to some of these functions. Most of them use the decreasing inertia weight. Some of the codes and demos available about the PSO algorithm can be found at the PSO website: http://www.swarmintelligence.org.

5.4.3. Scope of Particle Swarm Optimization

The original applications of PS focused primarily on neural network parameter adjustment and model selection. This corresponds to the problem of designing neural networks, including the network weights adjustment and structure definition processes. Model selection has been one of the most important application areas of evolutionary algorithms as well. Actually, some similarities can be observed between these approaches, but their distinctions are also undeniable; for instance, there can be no evolution if there is no selection, and PS does not account for selection, individuals move but never die. Kennedy et al. (2001) and Kennedy (2004) provide a good discussion about the similarities and differences between PS and other approaches.

PS algorithms have been applied to a number of numeric and parameter optimization problems, including real-world tasks. For instance, works can be found in the literature applying the PS approach to tasks like human tremor analysis, milling optimization, ingredient mix optimization, reactive power and voltage control, battery pack state-of-charge estimation, and improvised music composition.

5.4.4. From Social Systems to Particle Swarm

Section 5.4 discussed the importance of the social influence for the acquisition and improvement of knowledge. Some of these concepts led to the proposal of the particle swarm algorithm. Table 5.3 summarizes how to interpret the social-psychological views presented as a particle swarm approach.

Table 5.3: Interpretation of the sociocognitive domain into the particle swarm approach.

Social-Psychology	PS Algorithms
Individual (minds)	Particles in space
Population of individuals	Swarm of particles
Forgetting and learning	Increase or decrease in some attribute values of the particle
Individual own experience	Each particle has some knowledge of how it performed in the past and uses it to determine where it is going to move to
Social interactions	Each particle also has some knowledge of how other particles around itself performed and uses it to determine where it is going to move to

5.4.5. Summary of Particle Swarm Optimization

Although not explicitly discussed in the chapter that introduces the PS algorithm in Kennedy's book (Kennedy et al., 2001; Chapter 7), the algorithm was also motivated by the social behavior of organisms such as bird flocking and fish schooling (Kennedy and Eberhart, 1995). The algorithm is not only a tool for optimization, but also a tool for representing sociocognition of human and artificial agents, based on principles of social psychology. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also a result of interactions with other people. There are reports in the literature of feral humans, human children who grew up in the wild are commonly uneducable, unable to learn or to speak, unable to adapt to social life, and never able to show much evidence of cognition or thinking (Kennedy et al., 2001; Kennedy, 2004).

The PS algorithm as an optimization tool provides a population-based search procedure in which individuals, called particles, change their position with time. In real-valued PS, the particles fly around in a multidimensional search-space. During flight, each particle adjusts its position according to its own experience and according to the experience of a number of neighboring particles, making use of its best position so far and the best position of its neighbors as well. Therefore, the PS algorithm combines local search with global search, attempting to balance exploration with exploitation.

A particle swarm is thus another self-organizing system whose global dynamics emerge from local interactions. As each individual trajectory is adjusted toward its success and the success of neighbors, the swarm converges or clusters in optimal regions of the search-space.

5.5. SUMMARY

This chapter has discussed four main approaches in swarm intelligence: ant colony optimization, ant clustering, swarm robotics, and particle swarm. Ant colony optimization algorithms are suitable for finding minimal cost paths on a graph, mainly in those cases in which other more classical algorithms cannot be efficiently applied. Although each ant of the colony builds a solution to the problem at hand, good quality solutions can only emerge from the cooperative behavior of the colony. The communication among ants is performed indirectly via pheromone trails. Note also that the ants themselves are not adaptive individuals. Instead, they adapt the environment by releasing pheromone, thus, changing the way the problem is represented and perceived by other ants. This is one example of a stigmergic algorithm.

The clustering of dead bodies and larval sorting in ants has led to the development of the ant clustering algorithm. ACA is suitable for exploratory data analysis, in particular for the clustering of unlabelled data sets. It works by projecting the data into a bi-dimensional grid in which ants are allowed to move about carrying items to and from specific cells. Regions containing a large number of similar data reinforce the release of similar data, and vice-versa.

Most collective robotic systems were inspired by the collective behavior of ant colonies. Foraging, clustering of dead bodies, clustering around food, and collective prey retrieval have motivated the development of robotic systems to perform equivalent tasks. These systems are believed to be much promising when miniaturization is pursued. In such cases, they can aid in the unmanned exploration of inaccessible environments, in the large scale manufacturing of products, and in the medical sciences and surgery, among other things. The swarm robotics systems described here are other examples of stigmergic systems.

The particle swarm approach is inspired by the human social adaptation of knowledge and has demonstrated good performances in a number of applications, such as numeric function optimization and optimal design of other natural computing techniques (e.g., neural networks). The algorithm is conceptually

simple and easily implementable. Although this chapter has only presented its most well-known version, real-valued PS, binary and discrete versions of the PS algorithm are also available.

5.6. EXERCISES

5.6.1. Questions

1. Describe how the ant colony of the movie "Antz" is organized. Contrast the perspective of ant colonies presented in the movie with that of real ant colonies.
2. How many queens an ant or a termite colony may have? Discuss.
3. It is known that scientists do not have a detailed knowledge of the inner workings of insect swarms. The identification of the rules that govern individual behaviors is a major challenge, and without such knowledge it is hard to develop appropriate models and thus to predict their behavior.

   Can this (apparent) unpredictability of behavior be a drawback of the swarm intelligence approaches inspired by social insects?

   Would it be possible that a swarm of robots goes out of control due to this lack of knowledge?
4. The Ant Farm project of the MIT Artificial Intelligence Lab had as one of its final goals to design a structured community of micro-robots inspired by the social behavior of ants. By running the video recording of some of their experiments (http://www.ai.mit.edu/projects/ants/social-behavior.html), it is possible to observe a recruitment behavior of robots. Based upon the discussion presented in Section 5.3.3 and the information available at their website, identify which type of communication is being performed by the robots: direct or indirect communication. Comment on its efficiency.

   Can this recruitment strategy be used in conjunction with the work presented in Kube's videos (Section 5.3.3)? Discuss.
5. Section 5.3 was dedicated to swarm robotics. Suggest two real-world applications (different from those presented here) for each of these systems and discuss how these could be accomplished by swarm robotics.
6. In Section 2.2.3, it was described the basic behavior of termites' nest building. It was seen that, during nest building, termites are stimulated to work according to the current configuration of the local environment and other termites. In this case, the accumulation of pellets reinforces the dropping of more pellets in a given portion of the space. The intermediate results of their building process are pillars, while the final result is a mound. Is the model presented in Section 5.2.4 of the present chapter for the clustering of dead bodies and larval sorting appropriate to model termite nest building as well? Discuss.

7. The particle swarm approach was developed with an inspiration in the collective behavior of animals, such as flocks of birds, and can also be viewed as a sociocognitive theory of knowledge. In this case sociocognition refers to those qualities of thought about social beings and objects with their own features and in relation to others. In the particular case of human beings, it suggests that in order for us to become intelligent we need to interact with other humans. Using the authors own words:

   "In order for a brain to become mental, it needs to have interaction with other minds. A fetal brain grown to adulthood in a vat containing nutritious fluids would never acquire the qualities of mind. If it were transplanted into a Frankesteinian body, it would be unable to reason, remember, categorize, communicate, or do any of the other things that a minimal mind should be able to. Feral humans, that is, humans who have grown up in the wild, have never been known to show signs of having anything like a mind - it is not sufficient to have perceptual stimuli; social ones are necessary. The relation between mind and society is unquestionable … ." (Kennedy et al., 2001; p. 118)

   According to the authors, for a healthy human brain to develop into a mind, it has to become conscious and learn to think. These are some minimum requirements for something to be called a mind. To what extent do you believe that the interactions with other beings are important for the development of our intelligence and mind? There are known examples of human beings who were born in the wild and raised by wolves or monkeys and lately brought into towns. What happened to these human beings? Were they ever capable of adapting to social life and develop several basic skills such as eating with knifes? Could they ever fall in love?

8. Similarly to the Kohonen's self-organizing map (SOM) presented in the previous chapter, the ant clustering algorithm (ACA) is applicable to the unsupervised classification of data. Both algorithms make a projection of the input data set into an output grid. In the case of the SOM, the grid can be of any dimension, while for the ACA it is assumed a bi-dimensional grid. Given their main features, compare both strategies?

9. The particle swarm algorithm presents two features in common with the Kohonen self-organizing map studied in the previous chapter: (i) the use of a topological neighborhood among the components of the system; and (ii) the fact that particles are moved in the space taking into account their neighborhood.

   Discuss the similarities and differences between the neighborhood in SOM and in PS.

   Provide a geometric interpretation, in a 2D space, of the movement of a particle in the space taking into account Equations (5.21) and (5.23).

5.6.2. Computational Exercises

1. Write a pseudocode for the simple ACO (S-ACO) algorithm considering pheromone evaporation, implement it computationally, and apply it to solve the TSP instance presented in Section 3.10.4. Discuss the results obtained.

   Remove the pheromone evaporation term (Equation (5.3)), apply the algorithm to the same problem, and discuss the results obtained.

2. Apply the AS-TSP algorithm described in Algorithm 5.2 to solve the simple TSP instance described in Section 3.10.4.

   Compare the performance of the AS-TSP with that of the genetic algorithm applied to this same problem.

   Study and discuss the influence of parameters α and β in the performance of the algorithm.

3. In the first experiment performed by Lumer and Faieta (1994), the authors applied the ant clustering algorithm to a simple data set generated by four Gaussian distributions of mean μ and standard deviation σ, $G(\mu,\sigma)$. The authors used the distributions below to generate four different clusters of 200 data points each ($M = 800$, where M is the total number of data points), and the following simulation parameters $k_1 = 0.1$, $k_2 = 0.15$, $\alpha = 0.5$, $s^2 = 9$, and max_it = 10^6 to run their algorithm:

   $x \propto G(0.2,0.1)$, $y \propto G(0.2,0.1)$;

   $x \propto G(0.2,0.1)$, $y \propto G(0.8,0.1)$.

   $x \propto G(0.8,0.1)$, $y \propto G(0.2,0.1)$;

   $x \propto G(0.8,0.1)$, $y \propto G(0.8,0.1)$;

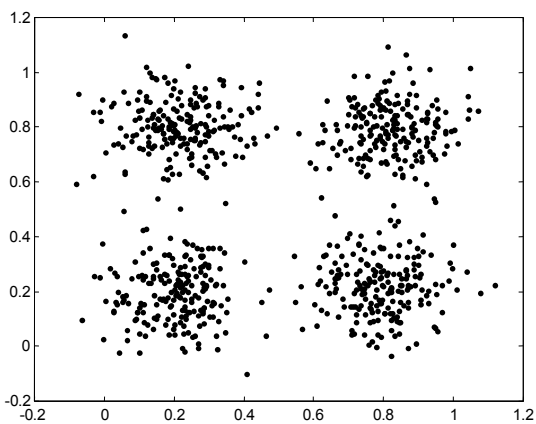


Figure 5.20: Distribution of points in $\Re^2$.

Each set of 200 points generated by each of the distributions above correspond to a cluster in $\Re^2$, as illustrated in Figure 5.20.

Implement the ant clustering algorithm (ACA) presented in Section 5.2.5 and test its performance when applied to the problem described above. Discuss the results obtained.

Was the algorithm capable of clustering similar data items into the same clusters in the bi-dimensional grid?

How many clusters appeared on the grid?

4. In Exercise 3 above, instead of using Equation (5.15) to determine the probability of depositing an item, use Equation (5.12).

   What happens with the performance of the algorithm?

5. Repeat the previous exercise introducing the three modifications proposed by Lumer and Faieta (1994) for the standard ACA: ants with different speeds, short term memory, and behavioral switches. Compare the performance of the modified algorithm with its standard version.

6. Repeat Exercise 3 introducing the three modifications proposed by Vizine et al. (2005): a cooling schedule for k_1; a progressive vision scheme; and the addition of pheromone to the environment. Compare the performance of this version of the algorithm with that of Exercises 3 and 4.

7. The Animals data set was originally proposed by Ritter and Kohonen (1989) to verify the self-organizing capability of one artificial neural network in creating a topographic map based on a symbolic data set (Section 4.4.5). The data set is composed of 16 input vectors each representing an animal with the binary feature attributes as shown in Table 5.4. A value of 1 in this table corresponds to the presence of an attribute, while a value of 0 corresponds to the lack of this attribute.

   Apply the ant clustering algorithm (ACA) to the animals data set presented above and study its performance. The local density f of items in the neighborhood of an object $\mathbf{x}_i$ situated at site r can be computed by a variation of Equation (15) with the distance between two items $\mathbf{x}_i$ and $\mathbf{x}_j$, $d(\mathbf{x}_i,\mathbf{x}_j)$, given by their Hamming distance in their original attribute space:

$$f(\mathbf{x}_i) = \begin{cases} \dfrac{1}{s^2} \displaystyle\sum_{\mathbf{x}_j \in \text{Neigh}_{(s\times s)}(r)} \left[L - \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\alpha} \right] & \text{if } f > 0 \\ 0 & \text{otherwise} \end{cases}$$

   where L is the length (number of attributes) of $\mathbf{x}_i$.

8. Apply the PS algorithm described in Section 5.4.1 to the maximization problem of Example 3.3.3. Compare the relative performance of the PS algorithm with that obtained using a standard genetic algorithm.

9. Modify the standard PS algorithm in order to accommodate the inertia weight given in Equation (5.24) and apply it to the same problem as above. Compare the results obtained.

Table 5.4: Animals data set: animals' names and their attributes.

		Dove	Hen	Duck	Goose	Owl	Hawk	Eagle	Fox	Dog	Wolf	Cat	Tiger	Lion	Horse	Zebra	Cow
Is	Small	1	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0
	Medium	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
	Big	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
Has	Two legs	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
	Four legs	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hair	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
	Hooves	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
	Mane	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
	Feathers	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
Likes to	Hunt	0	0	0	0	1	1	1	1	0	1	1	1	1	0	0	0
	Run	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
	Fly	1	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0
	Swim	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0

10. Repeat the previous exercise for the constriction factor χ.

5.6.3. Thought Exercises

1. The ant clustering algorithm (ACA) is based upon the projection of an input data set into a bi-dimensional grid. Is it possible to apply this algorithm without projecting the input data into a lower dimensional grid? What are the advantages and disadvantages of this dimensionality reduction process?
2. Equation (5.6) corresponds to the amount of pheromone each ant k is going to release on the edge (i,j) when this is traversed. As it stands, the amount of pheromone is absolute, in the sense that it is only proportional to the quality of the tour performed by this particular ant. Modify this equation so that the relative performance of ant k is taken into account. Discuss.

   Modify your implementation of ACO and investigate the effects of your proposed relative pheromone deposition rate.

5.6.4. Projects and Challenges

1. In Bullnheimer et al. (1999a) the authors applied the ant colony optimization algorithm in several instances of the vehicle routing problem (VRP) extracted from Christofides et al. (1979). All these instances are benchmark for combinatorial optimization algorithms and can be found in the literature.

   Implement the ACO algorithm and apply it to two instances of the VRP and compare its performance with other results from the literature.

2. The three main problem solving techniques discussed in this chapter are the ant colony optimization algorithms (ACO), the ant clustering algorithms (ACA), and the particle swarm (PS) algorithm. All these approaches involve a number of user-defined parameters, such as α, β, ρ, N, Q, and b for the ACO algorithm, k_p and k_d for the ACA, and w or χ, the number of particles, etc., for the PS algorithm.

   Choose one benchmark problem from the literature for each of these algorithms and study their sensitivity to these user-defined parameters. For example, by varying α, what happens to the performance of the ACO algorithm, and so on.

3. The ACO algorithm was originally developed to solve combinatorial optimization problems. However, it is known that some colonies of real ants are capable of finding the minimal path between the nest and the food source in the real world, i.e., in a continuous environment. Can you develop a modified (or new) ACO algorithm to solve continuous optimization problems? What would be the requirements for the artificial ants, visibility (if it is to be included), and artificial pheromone trails? Provide a number of problems in which this new ACO algorithm could be applied.

4. In Section 5.4.2 we described the use of a particle swarm algorithm for determining an appropriate weight set for a multi-layer perceptron neural network. Repeat Project 4 of Section 4.8.4 using a PS algorithm to train the neural network instead of using an evolutionary algorithm.

5.7. REFERENCES

[1] Angeline, P. A. (1998), "Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences", In. V. W. Porto, N. Saravanan, D. Waagen and A. E. Eiben (eds.), *Lecture Notes in Computer Science*, **1447**, Springer-Verlag, pp. 601–610.

[2] Bachand, G. D. and Montemagno, C. D. (2000), "Constructing Organic/Inorganic Nems Devices Powered by Biomolecular Motors", *Biomedical Microdevices*, **2**, pp. 179–184.

[3] Beckers, R., Holland, O. E. and Deneubourg, J. L. (1994), "From Local Actions to Global Tasks: Stigmergy and Collective Robotics", In R. A. Brooks and P. Maes (eds.) *Proc. of the 4th Int. Workshop on the Synthesis and Simulation of Life*, *Artificial Life IV*, MIT Press, pp. 181–189.

[4] Beni, G. (1988), "The Concept of Cellular Robotic Systems", *Proc. of the IEEE Int. Symp. on Intelligent Control*, pp. 57–62.

[5] Beni, G. and Wang, J. (1989), "Swarm Intelligence", *Proc. of the 7th Annual Meeting of the Robotics Society of Japan*, pp. 425–428.

[6] Bilchev, G. and Parmee, I. C., (1995), "The Ant Colony Metaphor for Searching Continuous Design Spaces", *Evolutionary Computing, AISB Workshop*, pp. 25–39

[7] Bonabeau, E. (1997), "From Classical Models of Morphogenesis to Agent-Based Models of Pattern Formation", *Artificial Life*, **3**, pp. 191–211.