Chapter 8
Swarm Intelligence

Daniel Merkle and Martin Middendorf

8.1 Introduction

The complex and often coordinated behavior of swarms fascinates not only biologists but also computer scientists. Bird flocking and fish schooling are impressive examples of coordinated behavior that emerges without central control. Social insect colonies show complex problem-solving skills arising from the actions and interactions of nonsophisticated individuals.

Swarm intelligence is a field of computer science that designs and studies efficient computational methods for solving problems in a way that is inspired by the behavior of real swarms or insect colonies (e.g. see Bonabeau et al. 1999; Kennedy et al. 2001). Principles of self-organization and local or indirect communication are important for understanding the complex collective behavior (Sumpter 2009). Examples where insights into the behavior of natural swarms has influenced the design of algorithms and systems in computer science are:

- Models for the division of labor between members of an ant colony were used to regulate the joint work of robots and collective transport of ants has inspired the design of controllers of robots for doing coordinated work (e.g. Labella et al. 2006);
- Brood sorting behavior of ants motivated several clustering and sorting algorithms (e.g. Handl and Meyer 2002; Lumer and Faieta 1994);

D. Merkle
Department of Mathematics and Computer Science, University of South Denmark, Odense, Denmark
e-mail: daniel@imada.sdu.dk

M. Middendorf (✉)
Department of Computer Science, University of Leipzig, Leipzig, Germany
e-mail: middendorf@informatik.uni-leipzig.de

E.K. Burke and G. Kendall (eds.), *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, DOI 10.1007/978-1-4614-6940-7_8,

- The synchronized flashing behavior observed in some species of fireflies has inspired algorithms for detecting non-operational robots in a swarm robotic system (Christensen et al. 2009).

In this chapter we focus on swarm intelligence methods for solving optimization and search problems. The two main areas of swarm intelligence that are relevant for such problems are ant colony optimization (ACO) and particle swarm optimization (PSO). A third emerging area is honey bee optimization (HBO).

ACO is a metaheuristic for solving combinatorial optimization problems. It is inspired by the way real ants find shortest paths from their nest to food sources. An essential aspect thereby is the indirect communication of the ants via pheromone, i.e. a chemical substance which is released into the environment and that influences the behavior or development of other individuals of the same species. Ants mark their paths to the food sources by laying trail pheromone along their way. The pheromone traces can be smelled by other ants and lead them to the food source.

PSO is a metaheuristic that is mainly used for finding maximum or minimum values of a function. PSO is inspired by the behavior of swarms of fishes or flocks of birds to find a good food place. The coordination of movements of the individuals in the swarm is the central aspect that inspires PSO.

HBO denotes a class of algorithms that are inspired by the collective behavior of honey bees. In particular, bee's mating behavior and related genetic principles, bee's foraging behavior, and their collective nest site selection behavior have been utilized for algorithm design. Depending on their type HBO algorithms are used for combinatorial optimization or function optimization. HBO is not covered in this tutorial. More information on HBO can be found in the overview articles by Diwold et al. (2011) and Karaboga and Akay (2009).

8.2 Ant Colony Optimization

A famous biological experiment called the double-bridge experiment was the inspiring source for the first ACO algorithm (Dorigo 1992; Dorigo et al. 1991). The double-bridge experiment (Deneubourg et al. 1990; Goss et al. 1989) was designed to investigate the pheromone trail laying and following behavior of the Argentine ant *Iridomyrmex humilis*. In the experiment a double bridge with two branches of different lengths connected the nest of this species with a food source (see Fig. 8.1). The long branch of the bridge was twice as long as the shorter branch. In most runs of this experiment it was found that after a few minutes nearly all ants use the shorter branch. This is interesting because Argentine ants cannot see very well. The explanation of this behavior has to do with the fact that the ants lay pheromone along their path. It is likely that ants which randomly choose the shorter branch arrive earlier at the food source. When they go back to the nest they smell some pheromone on the shorter branch and therefore prefer this branch. The pheromone on the shorter branch will accumulate faster than on the longer branch so that after some time the concentration of pheromone on the former is much higher and nearly all ants take the shorter branch. Similar to the experiment with branches

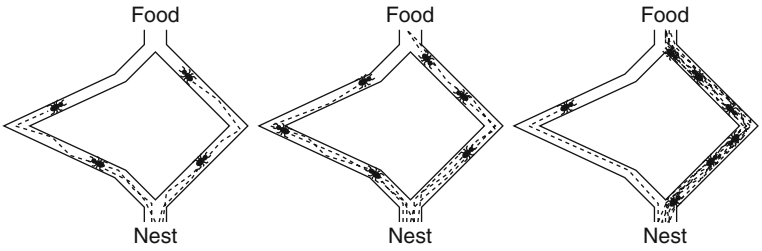

Fig. 8.1 Double-bridge experiment

of different lengths, when both branches have the same length, after some minutes nearly all ants use the same branch. But in several repetitions it is a random process which of the two branches will be chosen. The explanation is that when one branch has got a slightly higher pheromone concentration due to random fluctuations this branch will be preferred by the ants so that the difference in pheromone concentration will increase and after some time all ants take this branch.

Inspired by this experiment Dorigo and colleagues designed an algorithm for solving the traveling salesperson problem (TSP) (Dorigo 1992; Dorigo et al. 1991) and initiated the field of ACO. In recent years this field of research has become quite rich so that ACO algorithms have now been designed for various application problems and different types of combinatorial optimization problems including dynamic and multi-objective optimization problems. Some progress has been made in the last years on the theory of ACO algorithms (see Dorigo and Blum 2005; Gutjahr 2011 for an overview). An ACO metaheuristic has been formulated as a generic frame that contains most of the different ACO algorithms that have been proposed so far (see Dorigo and Di Caro 1999).

The idea of ACO is to let artificial ants construct solutions for a given combinatorial optimization problem. A prerequisite for designing an ACO algorithm is to have a constructive method which can be used by an ant to create different solutions through a sequence of decisions. Typically an ant constructs a solution by a sequence of probabilistic decisions where every decision extends a partial solution by adding a new solution component until a complete solution is derived. The sequence of decisions for constructing a solution can be viewed as a path through a corresponding decision graph (also called construction graph). Hence, an artificial ant that constructs a solution can be viewed as walking through the decision graph. The aim is to let the artificial ants find paths through the decision graph that correspond to good solutions. This is done in an iterative process where the good solutions found by the ants of an iteration should guide the ants of following iterations. Therefore, ants that have found good solutions are allowed to mark the edges of the corresponding path in the decision graph with artificial pheromone. This pheromone guides following ants of the next iteration so that they search near the paths to good solutions. In order that pheromone from older iterations does not influence the following iterations for too long, during an update of the pheromone values some percentage of the pheromone evaporates. Thus, an ACO algorithm is an iterative process where pheromone information is transferred from one iteration

to the next one. The process continues until some stopping criterion is met, e.g. a certain number of iterations has been done or a solution of a given quality has been found. A scheme of an ACO algorithm is given in the following:

ACO scheme:
Initialize pheromone values
repeat
for ant $k \in \{1, \ldots, m\}$
construct a solution
endfor
forall pheromone values **do**
decrease the value by a certain percentage {evaporation}
endfor
forall pheromone values corresponding to good solutions
do
increase the value {intensification}
endfor
until stopping criterion is met

We illustrate how the general ACO scheme can be applied to a broad class of optimization problems by means of three examples. In the first example a more detailed ACO scheme is described and applied to the TSP. An alternative approach is contained in the second example. The third example is an application of ACO to a scheduling problem which is used in comparison to the first example to discuss some additional aspects that have to be considered for designing ACO algorithms.

8.2.1 Example 1: Basic ACO and the TSP

The objective of ACO is to find good solutions for a given combinatorial optimization problem (Dorigo 1992; Dorigo and Di Caro 1999; Dorigo et al. 1991). For an easier description we restrict the following description to the broad class of optimization problems which have solutions that can be expressed as permutations of a set of given items. Such problems are called permutation problems and a famous example is the TSP. After a definition of the TSP we describe the elements of the ACO scheme that constitute an ACO algorithm, namely

- Pheromone information
- Solution construction
- Pheromone update: evaporation + intensification
- Stopping criterion

8.2.1.1 The TSP Problem

This problem is to find for a given set of n cities with distances d_{ij} between each pair of cities $i, j \in [1 : n]$ a shortest closed tour that visits every city exactly once.

Every such tour together with a start city can be characterized by the permutation of all cities as they are visited along the tour. Vice versa, each permutation of all cities corresponds to a valid solution, i.e. a closed tour.

8.2.1.2 Pheromone Information

An important part in the design of an ACO algorithm is to find a definition of the pheromone information so that it reflects the most relevant information for the solution construction. The pheromone information for permutation problems can usually be encoded in an $n \times n$ pheromone matrix $[\tau_{ij}]$, $i, j \in [1:n]$. For the TSP problem pheromone value τ_{ij} expresses the desirability to assign city j after city i in the permutation. The pheromone matrix for the TSP problem is initialized so that all values τ_{ij} with $i \neq j$ are the same. Note that the values τ_{ii} are not needed because each city is selected only once.

TSP-ACO:
  Initialize pheromone values
repeat
 for ant $k \in \{1,\ldots,m\}$ {solution construction}
   $S := \{1,\ldots,n\}$ {set of selectable cities}
   choose city i with probability p_{0i}
  repeat
   choose city $j \in S$ with probability p_{ij}
   $S := S - \{j\}$
   $i := j$
  until $S = \emptyset$
 endfor
 forall i, j **do**
  $\tau_{ij} := (1-\rho) \cdot \tau_{ij}$ {evaporation}
 endfor
 forall i, j in iteration best solution **do**
  $\tau_{i,j} := \tau_{ij} + \Delta$ {intensification}
 endfor
until stopping criterion is met

8.2.1.3 Solution Construction

An iterative solution construction method that can be used by the ants is to start with a random item and then always choose the next item from the set S of selectable items that have not been selected so far until no item is left. Initially, the set of selectable items S contains all items; after each decision, the selected item is removed from S. Recall that in the case of the TSP the items are the cities. Every decision is made randomly where the probability equals the amount of pheromone relative to the sum of all pheromone values of items in the selection set S:

$$p_{ij} := \frac{\tau_{ij}}{\sum_{z \in S} \tau_{iz}} \quad \forall j \in S.$$

For most optimization problems additional problem-dependent heuristic information can be used to give the ants additional hints on which item to choose next. To each pheromone value τ_{ij} there is defined a corresponding heuristic value η_{ij}. For the TSP a suitable heuristic is to prefer a next city j that is near to the current city i, e.g. by setting $\eta_{ij} := 1/d_{ij}$. The probability distribution when using a heuristic is

$$p_{ij} := \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{z \in S} \tau_{iz}^{\alpha} \cdot \eta_{iz}^{\beta}} \quad \forall j \in S, \tag{8.1}$$

where parameters α and β are used to determine the relative influence of pheromone values and heuristic values.

In order to more strongly exploit the pheromone information it has been proposed that the ant follows with some probability $q_0 \in (0,1)$ the strongest trail, i.e. the edge in the decision graph with the maximal product of pheromone value and corresponding heuristic information (Dorigo and Gambardella 1997). For this case q_0 is a parameter of the algorithm and with probability q_0 an ant chooses next city j from the selectable cities in S which maximizes $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$. With probability $1 - q_0$ the next item is chosen according to the probability distribution determined by Eq. (8.1).

8.2.1.4 Pheromone Update

All m solutions that are constructed by the ants in one iteration are evaluated according to the respective objective function and the best solution π^* of the current iteration is determined. Then the pheromone matrix is updated in two steps:

1. Evaporation: All pheromone values are reduced by a fixed proportion $\rho \in (0,1)$:

$$\tau_{ij} := (1-\rho) \cdot \tau_{ij} \quad \forall\, i,j \in [1:n].$$

2. Intensification: All pheromone values corresponding to the best solution π^* are increased by an absolute amount $\Delta > 0$:

$$\tau_{i\pi^*(i)} := \tau_{i\pi^*(i)} + \Delta \quad \forall\, i \in [1:n].$$

8.2.1.5 Stopping Criterion

The ACO algorithm executes a number of iterations until a specified stopping criterion has been met. The most commonly used stopping criteria are (possibly used in combination) that a predefined maximum number of iterations has been executed, a specific level of solution quality has been reached, or the best solution has not changed over a certain number of iterations.

Table 8.1 ACO variables and parameters

τ_{ij}	Pheromone value
η_{ij}	Heuristic value
m	Number of ants per iteration
$\bar{m}$	Number of ants per iteration allowed to increase pheromone
α	Influence of pheromone
β	Influence of heuristic
ρ	Evaporation rate
Δ	Amount of pheromone added during pheromone intensification
q_0	Probability to follow the strongest trail
π^*	Best solution in the actual iteration
π^e	Best solution found so far (elitist solution)

A good comparison of the optimization behavior of different ACO implementations for the TSP problem can be found in Stützle and Hoos (2000). The parameters and variables of ACO algorithms introduced in this section are summarized in Table 8.1.

8.2.2 Example 2: Population-Based ACO and TSP

In standard ACO algorithms the information that is transferred from one iteration to the next is the pheromone information—in the case of permutation problems this is the pheromone matrix. An alternative approach is population-based ACO (P-ACO) which was proposed by Guntsch and Middendorf (2002b). The idea of P-ACO is to transfer a small amount of only the most important information from one iteration to the next. This is done in the form of a small population of good solutions. An advantage of P-ACO is that the population of solutions makes it possible to apply operations from other metaheuristics to it, e.g. the crossover operation from genetic algorithms which builds a new solution by combining properties of two solutions that are already in the population. In this section we describe the differences between P-ACO and standard ACO for permutation problems. It has been shown that P-ACO is competitive to the state-of-the-art ACO algorithms with the advantage of finding good solution quality in a shorter computation time (Guntsch and Middendorf 2002b; Oliveira et al. 2011). A scheme of a P-ACO algorithm for the TSP is given in the following (compare with the scheme of ACO-TSP).

8.2.2.1 Information Transfer and Population Matrix

Instead of a complete pheromone matrix as in ACO, P-ACO transfers a small population P of the k best solutions that have been found in past iterations. Since each solution for a permutation problem is a permutation of n items, the population can be stored in an $n \times k$ matrix $P = [p_{ij}]$, where each column of P contains one solution.

P-ACO-TSP:
$P := \emptyset$
Initialize pheromone values
repeat
for ant $k \in \{1,\ldots,m\}$ {solution construction}
$S := \{1,\ldots,n\}$ {set of selectable cities}
choose city i with probability p_{0i}
for $i = 1$ **to** n **do**
choose city j with probability p_{ij}
$S := S - \{j\}$
$i := j$
endfor
endfor
If $|P| = k$ remove the oldest solution $\bar{\pi}$ from the population: $P := P - \bar{\pi}$
Determine the best solution of the iteration and add it to the population: $P := P + \pi^*$
Compute the new pheromone matrix from P
until stopping criterion is met

This matrix is called the population matrix. It contains the best solution of each of the preceding k iterations. When employing an elitism strategy, the best solution found so far in all iterations is—as in standard ACO—also always transferred to the next iteration. In that case the population matrix contains an additional column for the elitist solution.

8.2.2.2 Population Matrix Update

When the ants in an iteration have constructed their solutions the population (matrix) is updated. The best solution of the current iteration is added to P. If, afterwards, P contains $k+1$ solutions the oldest solution is removed from P. The initial population is empty and after the first k iterations the population size remains k. Hence, for an update only one column in the population matrix has to be changed. Additionally, if elitist update is used and the best solution of the iteration is better than the elitist solution, the corresponding column is overwritten by the new solution. Note that each solution in the population has an influence on the decisions of the ants over exactly k subsequent iterations. Other schemes for deciding which solutions should enter/leave the population are discussed in Guntsch and Middendorf (2002a).

8.2.2.3 Construction of Pheromone Matrix

In P-ACO a pheromone matrix (τ_{ij}) is used by the ants for solution construction in the same way as in standard ACO. But differently, in P-ACO the pheromone matrix is derived in every iteration anew from the population matrix as follows.

Each pheromone value is set to an initial value $\tau_{init} > 0$ and is increased, if there are corresponding solutions in the population:

$$\tau_{ij} := \tau_{init} + \zeta_{ij} \cdot \Delta \tag{8.2}$$

with ζ_{ij} denoting the number of solutions $\pi \in P$ with $\pi(i) = j$, i.e. $\zeta_{ij} = |\{h : p_{ih} = j\}|$. Hence, in P-ACO a pheromone value is equal to one of the following possible values $\tau_{init}, \tau_{init} + \Delta, \ldots, \tau_{init} + k \cdot \Delta$ (when using an elitist solution $\tau_{init} + (k+1) \cdot \Delta$ is also possible). An update of the pheromone values is done implicitly by a population update:

- A solution π entering the population, corresponds to a positive update:

$$\tau_{i\pi(i)} := \tau_{i\pi(i)} + \Delta.$$

- A solution σ leaving the population, corresponds to a negative update:

$$\tau_{i\sigma(i)} := \tau_{i\sigma(i)} - \Delta.$$

Note that a difference to the standard ACO algorithm is that no evaporation is used to reduce the pheromone values at the end of an iteration.

8.2.3 Example 3: ACO for a Scheduling Problem

In this section the ACO approach is applied to a scheduling permutation problem called the single machine total weighted tardiness problem (SMTWTP). The differences between the ACO algorithm for the SMTWTP and the TSP-ACO illuminate two important aspects for the design of ACO algorithms, namely the pheromone encoding and the pheromone evaluation (see Blum and Sampels 2002b; Merkle and Middendorf 2002, 2005 for more results on the importance of the pheromone model). Moreover, the proper adaptation of heuristics to be used for ACO is discussed. These aspects can be arranged as follows into the list of elements that constitute an ACO algorithm:

- Pheromone information
  - Pheromone encoding
- Solution construction
  - Pheromone evaluation
  - Adaptation of heuristics.

8.2.3.1 The SMTWTP Problem

For the SMTWTP n jobs are given that have to be scheduled onto a single machine. Every job $j \in [1 : n]$ has a due date d_j, a processing time p_j and a weight w_j. If C_j

denotes the completion time of job j in a schedule, then $L_j = C_j - d_j$ defines its lateness and $T_j = \max(0, L_j)$ its tardiness. The objective is to find a schedule that minimizes the total weighted tardiness of all jobs $\sum_{j=1}^{n} w_j T_j$.

8.2.3.2 Pheromone Encoding

When designing an ACO algorithm for an optimization problem it is important to encode the pheromone information in a way that is suitable for the problem. For the TSP it is relevant which cities are following each other in the permutation because the distance between the cities determines the quality of the solution. Therefore, pheromone values τ_{ij} are used to express the desirability that city j comes after i. For the SMTWTP the relative position of a job in the schedule is much more important than its direct predecessor or its direct successor in the schedule (see also Blum and Sampels 2002a for other scheduling problems). Therefore pheromone values for the SMTWTP are used differently than for the TSP. Pheromone value τ_{ij} expresses the desirability to assign item j at place i of the permutation. Thus, this pheromone matrix is of type place $\times$ item whereas the pheromone matrix used for the TSP is of type item $\times$ item. For SMTWTP an ant starts to decide which job is the first in the schedule and then always decides which job is on the next place. The pheromone matrix for the SMTWTP problem is initialized so that all values τ_{ij}, $i, j \in [1 : n]$ are the same.

8.2.3.3 Pheromone Evaluation

Another important aspect of ACO algorithms is how the pheromone information is used by the ants for their decisions. Real ants use trail pheromone only locally because they cannot smell it over long distances. The artificial ants in TSP-ACO also use the pheromone values locally which means that an ant at city i considers only the pheromone values τ_{ij} that lead to a possible next city $j \in S$. In principle, a local evaluation of the pheromone values is also possible for the SMTWTP (and has been used so—Bauer et al. 1999). An ant that has to decide which job is in the next place i in the permutation considers all values τ_{ij}, $j \in S$ which indicates how well the selectable jobs have performed in this place. But assume that for some selectable job $j \in S$ its highest pheromone value is τ_{lj} for an $l < i$. This indicates that for job j place l in the schedule is very good. But this also means that job j should not be placed much later than place l in order not to risk a due date violation. Therefore, even when the value τ_{ij} is small the ant should choose job l with high probability. Therefore, for SMTWTP a global pheromone evaluation rule has been proposed which is called summation evaluation because an ant that has to decide about place i of the permutation makes the selection probability for every selectable job dependent on the sum of all pheromone values for this job up to place i (Merkle and Middendorf 2003b):

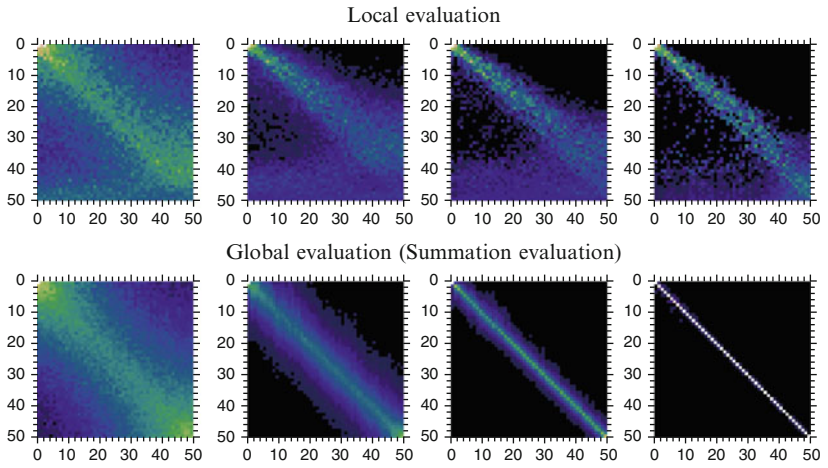

Fig. 8.2 Comparison between SMTWTP-ACO with local evaluation and summation evaluation: pheromone matrices averaged over 25 runs in iterations 800, 1,500, 2,200 and 2,900 (left to right): brighter *gray* colors indicate higher pheromone values

$$p_{ij} = \frac{(\sum_{l=1}^{i} \tau_{lj})^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{z \in S} (\sum_{l=1}^{i} \tau_{lz})^{\alpha} \cdot \eta_{iz}^{\beta}} \quad \forall j \in S. \tag{8.3}$$

To demonstrate the influence of the pheromone evaluation method and the change of pheromone values in ACO we show results of a very simple SMTWTP test instance (for more results and another global pheromone evaluation method see Merkle and Middendorf 2003a). It consists of 50 jobs where job $i \in [1:50]$ has processing time $p_i = 1$, due date $d_i = i$ and weight $w_i = 1$. Clearly, to place job i on place i is the only optimal solution with total weighted tardiness 0. Figure 8.2 shows the average change of pheromone values for ACO-SMTWTP with local and with global evaluation (no heuristic was used) for several runs with $m = 10$ ants per iteration. The figure shows clearly that for this problem summation evaluation performs much better than local evaluation. Compared to local evaluation the results of summation evaluation depicted in Fig. 8.2 show a very symmetric behavior and do not have the undesired property that some of the jobs with a small number are scheduled very late.

Table 8.2 Influence of global pheromone evaluation and adapted heuristic on solution quality for SMTWTP. Difference of total tardiness to total tardiness of best results from the literature average over 125 instances (see Merkle and Middendorf 2003b for details); Σ with summation evaluation; H with adapted heuristic (8.5)

ACO-ΣH	ACO-Σ	ACO-H	ACO
79.5	200.0	204.5	1198.6

8.2.3.4 Adaptation of Heuristics

For many (scheduling) problems there exist priority heuristics which can be used to decide which job is next when building a schedule. An example for the unweighted form of the SMTWTP is the modified due date rule (MDD), that is

$$\eta_{ij} = \frac{1}{\max\{\mathcal{T} + p_j, d_j\}}, \tag{8.4}$$

where $\mathcal{T}$ is the total processing time of all jobs already scheduled. Observe, that the heuristic prefers jobs with a small due date from all jobs that would finish before their due date when scheduled next. Furthermore, of all those jobs that will finish after their due date, the jobs with short processing times are preferred. Some care has to be taken when using standard priority heuristics for scheduling problems in an ACO algorithm because the heuristic values might not properly reflect the relative influence they should have on the decisions of the ants. In the case of the MDD heuristic the problem occurs that the values of $\max\{\mathcal{T} + p_j, d_j\}$ become much larger—due to $\mathcal{T}$—when deciding about jobs to place further at the end of the schedule. As a consequence, the heuristic differences between the jobs are, in general, small at the end of the schedule. This means that the ants cannot really differentiate between the various alternatives. To avoid this effect an accordingly adapted heuristics should be used (Merkle and Middendorf 2003b), e.g.

$$\eta_{ij} = \frac{1}{\max\{\mathcal{T} + p_j, d_j\} - \mathcal{T}}. \tag{8.5}$$

To illustrate the effect of an adapted heuristic together with global pheromone evaluation some test results for benchmark problems with $n = 100$ jobs from the OR-Library (2012) are given. The ACO parameters used are $m = 20$ ants per generation $\alpha = 1$, $\beta = 1$, $\rho = 0.1$, $q_0 = 0.9$, and local optimization was applied to solutions found by the ants (see Merkle and Middendorf 2003b for more details). Table 8.2 compares the behavior of the algorithm using non-adapted heuristic (8.4) and local pheromone evaluation with the algorithms that use one or both of adapted heuristic (8.5) and global pheromone evaluation. The results clearly show that using an adapted heuristic (8.5) or the global pheromone evaluation improves the results significantly, and using both is best.

8.2.4 Advanced Features of ACO

In this section several variations and extension of the ACO algorithms are described that often lead to an increased search efficiency and better optimization results.

8.2.4.1 Variants of Pheromone Update

Several variations of pheromone update have been proposed in the literature:

- *Quality-dependent pheromone update*. In some ACO algorithms not only the best solution, but the $\bar{m} < m$ best solutions of each iteration are allowed to increase the pheromone values. In addition the amount of pheromone that is added can be made dependent on the quality of the solution so that the more pheromone added, the better the solution is (Dorigo et al. 1996). For the TSP this means that for shorter tours more pheromone is added.
- *Rank-based pheromone update*. Here the $\bar{m} \leq m$ best ants of an iteration are allowed to increase the pheromone. The amount of pheromone an ant is allowed to add depends on its rank within the $\bar{m}$ best solutions and the quality of the solution (Bullnheimer et al. 1999).
- *Elitist solution pheromone update*. It can be advantageous to enforce the influence of the best solution that has been found so far over all iterations, called the elitist solution (Dorigo et al. 1996). This is done by adding pheromone during pheromone intensification also according to this solution. Several variations have been studied: e.g. to let randomly update either the iteration best or the elitist solution with increasing probability for an elitist update (Stützle and Hoos 2000) or to apply elitist pheromone update but to forget the elitist solution after several iterations by replacing it with the iteration best solution (Merkle et al. 2002).
- *Best–worst pheromone update*. This pheromone update method in addition to the standard pheromone update reduces the pheromone values according to the worst solution of an iteration provided that a pheromone value does not also correspond to the elitist solution (Cordón et al. 2000). A problem for this method is that often a decision which can lead to bad solutions can also lead to a very good solution.
- *Online step-by-step pheromone update* (Dorigo and Gambardella 1997; Dorigo et al. 1991). This means that an ants adds or removes pheromone from an edge in the decision graph it has chosen immediately after the decision was made (see Dorigo and Di Caro 1999 for more details). One motivation to use online step-by-step pheromone update in addition to the standard update is to remove pheromone to increase the variability in the choices of the ants during an iteration.
- *Moving-average pheromone update*. A pheromone update scheme where each constructed solution is allowed to update the pheromone (Maniezzo 1999). When the actual solution is better than the average quality of the last $k > 0$ solutions than it increases its corresponding pheromone values and otherwise it decreases them.
- *Minimum pheromone values*. The use of minimum pheromone values was proposed in order to guarantee that each possible choice always has a minimum probability to be chosen (Stützle and Hoos 2000).

8.2.4.2 Other ACO Variants

Several variants of ACO algorithms which do not use the pheromone update have also been proposed (see Dorigo and Di Caro 1999 for an overview):

- *Candidate lists*. A candidate list defines for each decision a set of preferable choices (Dorigo and Gambardella 1997). For the TSP a candidate list can be defined for each city to determine the set of preferred successor cities. An ant then chooses, if possible, the next city only from cities that are in the selection set S and also in the candidate list.
- *Lower bounds*. The use of lower bounds on the cost of completing a partial solution was proposed in Maniezzo (1999). The lower bounds give additional heuristic information about the possible choices.
- *Lookahead*. A lookahead strategy was proposed by Michels and Middendorf (1999) where for each possible choice of an ant the maximum $\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}$ value that would result from this choice is evaluated and taken into account when actually making a decision.
- *Stagnation recovery*. For longer runs of an ACO algorithm there is the danger that after some time the search concentrates too much on a small search region. Several authors have proposed methods for modification of the pheromone information to counteract such stagnation behavior of ACO algorithms. When stagnation is detected the approach of Gambardella et al. (1999) is to reset all elements of the pheromone matrix to their initial values. Stützle and Hoos (1997) suggested increasing the pheromone values proportionately to their difference to the maximum pheromone value.
- *Changing α, β values*. Merkle et al. (2002) proposed reducing the value of β during a run to increase the influence of the pheromone at later stages of the algorithm. See item "Stagnation recovery" (above) for changing α values.
- *Repelling pheromone*. Some experiments with pheromone that allow the ants to avoid choosing an edge have been performed by Kawamura et al. (2000) and Montgomery and Randall (2002) in order to enforce ants (or different colonies of ants) to search in different regions of the search space.
- *Moving direction*. The use of ants that "move in different directions" can improve the optimization behavior (Michels and Middendorf 1999). For a permutation problem this could mean that some ants decide first which item is in place one of the permutation and other ants decide first which item is in the last place. In general, it can be said that the ants should make important decisions early (Merkle and Middendorf 2005).
- *Local improvement of solutions*. The use of local optimization strategies to improve the solutions that have been found by the ants has been applied so successfully (e.g. Dorigo and Gambardella 1997; Stützle et al. 2000) that most state-of-the-art ACO algorithms use local improvement strategies. Two variants of the use of local improvement strategies exist: (i) to determine how much pheromone is updated for a solution, the quality or rank of the solution is computed after the local improvement has been applied but the actual pheromone

update is done according to the original solution before the local improvement, (ii) the same as (i) but the pheromone update is done according to the solution after local improvement.

8.2.5 Promising Areas for Future Applications of ACO

An important area of research for practical applications of ACO is to create hybrid algorithms that combine properties of ACO with other search heuristics. Other promising fields like multiobjective optimization, dynamic and probabilistic optimization, hybrid algorithms, parallel and hardware algorithms, and theoretical aspects cannot be covered in this introductory tutorial.

8.2.5.1 Hybrid ACO

The aim of hybrid search heuristics is to combine the advantages of different types of search heuristics. In recent years hybrids between ACO and the following metaheuristics have been designed: genetic algorithms (GAs), particle swarm optimization (PSO), simulated annealing (SA), scatter search, path relinking, greedy randomized adaptive local search procedure (GRASP), tabu search, and others. The *International Workshop on Hybrid Metaheuristics* is particularly devoted to the study of this research field (see Blesa et al. 2009 for the latest proceedings).

8.3 Particle Swarm Optimization

The roots of the metaheuristic that is described in this section lay in computing models that have been created by scientists in the last two decades to simulate bird flocking and fish schooling. The coordinated search for food which let a swarm of birds land at a certain place where food can be found was modeled with simple rules for information sharing between the individuals of the swarm. These studies inspired Kennedy and Eberhart to develop a method for function optimization that they called particle swarm optimization (Kennedy and Eberhart 1995). A PSO algorithm maintains a population of particles (the swarm), where each particle represents a location in a multidimensional search space (also called problem space). The particles start at random locations and search for the minimum (or maximum) of a given objective function by moving through the search space. The analogy to reality (in the case of a search for a maximum) is that the function measures the quality or amount of the food at each place and the particle swarm searches for the place with the best or most food. The movements of a particle depend only on its velocity and the locations where good solutions have already been found by the particle itself or other (neighbored) particles in the swarm. This is again in analogy to

bird flocking where each individual makes its decisions based on cognitive aspects (modeled by the influence of good solutions found by the particle itself) and social aspects (modeled by the influence of good solutions found by other particles). Note that, unlike many deterministic methods for continuous function optimization, PSO uses no gradient information.

In a typical PSO algorithm each particle keeps track of the coordinates in the search space which are associated with the best solution it has found so far. The corresponding value of the objective function (fitness value) is also stored. Another "best" value that is tracked by each particle is the best value obtained so far by any particle in its topological neighborhood. When a particle takes the whole population as its neighbors, the best value is a global best. At each iteration of the PSO algorithm the velocity of each particle is changed towards the personal and global best (or neighborhood best) locations. But also some random component is incorporated into the velocity update. A scheme for a PSO algorithm is given below.

PSO scheme:
Initialize location and velocity of each particle
repeat
for each particle
evaluate objective function f at the particles location
endfor
for each particle
update the personal best position
endfor
update the global best position
for each particle
update the velocity
compute the new location of the particle
endfor
until stopping criterion is met

PSO has become a very active field of research with several hundreds of publications per year. The main use of PSO is as function optimizer that is often embedded into a more complex application context. An extensive overview on different fields of applications is given in Poli (2008). Most applications of PSO are in the fields of image and video analysis, control, distribution networks, power systems and plants (Alrashidi and El-Hawary 2009; del Valle et al. 2008), electronics and electromagnetics, design signal processing (Merkle and Middendorf 2008), biomedicine, communication networks, data mining, fuzzy systems and neural networks. Some works have also been done to apply PSO to discrete problems, e.g. scheduling problems.

In the following we describe in more detail how the PSO scheme can be applied to optimization problems. The first example considers the typical use of PSO for continuous optimization. A subset problem is addressed in the second example to illustrate how PSO can be applied to other types of optimization problems.

8.3.1 Example 1: Basic PSO and Continuous Function Optimization

In order to describe the PSO algorithm for function optimization we need some notation. Let f be a given objective function over a D-dimensional problem space. The location of a particle $i \in \{1,\ldots,m\}$ is represented by a vector $\mathbf{x}_i = (x_{i1},\ldots,x_{iD})$ and the velocity of the particle by the vector $\mathbf{v}_i = (v_{i1},\ldots,v_{iD})$. Let l_d and u_d be lower and upper bounds for the particles coordinates in the dth dimension, $d \in [1:D]$. The best previous position of a particle is recorded as $\mathbf{p}_i = (p_{i1},\ldots p_{iD})$ and is called *pBest*. The index of the particle with the so far best found position in the swarm is denoted by g and $\mathbf{p}_g$ is called *gBest*.

At each iteration of a PSO algorithm after the evaluation of function f the personal best position of each particle i is updated, i.e. if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$ then set $\mathbf{p}_i = \mathbf{x}_i$. If $f(\mathbf{p}_i) < f(\mathbf{p}_g)$ then i becomes the new global best solution, i.e. set $g = i$. Then the new velocity of each particle i is determined during the update of velocity in every dimension $d \in [1:D]$ as follows:

$$v_{id} = w \cdot v_{id} + c_1 \cdot r_1 \cdot (p_{id} - x_{id}) + c_2 \cdot r_2 \cdot (p_{gd} - x_{id}), \tag{8.6}$$

where

- Parameter w is called the *inertia weight*, it determines the influence of the old velocity; the higher the value of w the more the individuals tend to search in new areas; typical values for w are slightly smaller than 1.0;
- c_1 and c_2 are the *acceleration coefficients*, which are also called the cognitive and the social parameter respectively, because they are used to determine the influence of the local best position and the global best position respectively; typical values are $c_1 = c_2 = 2$;
- r_1 and r_2 are random values uniformly drawn from $[0,1]$.

After velocity update the new position of particle i is determined by

$$x_{id} = x_{id} + v_{id}.$$

If there is a maximum range for the location in dimension d, i.e. $x_g \in [l_d, u_d]$, then the particle is reflected.

The behavior of PSO algorithms is usually studied and compared on a set of standard test functions. Examples of the most prominent test functions are given in Table 8.3. These functions represent different types of functions, e.g. the variables in Sphere and Rastrigin are uncorrelated which is not the case for the other functions in the table. Most of these functions are typically used for 10–100 dimensions.

As an example we consider a test run of the standard PSO with a swarm of size $m = 10$ on the two-dimensional Sphere function (the PSO parameters used are $w = 0.729$, $c_1 = c_2 = 1.494$). It can be seen from Fig. 8.3 (left) that the swarm proceeds from initial random positions at iteration $t = 0$ towards the single minimum value of the Sphere function. The velocity vectors of the particles at iteration $t = 10$ are shown in Fig. 8.3 (right).

Table 8.3 Test functions

Sphere
$f_1(x) = \sum_{i=1}^{D} x_i^2$
Rastrigin
$f_2(x) = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$
Rosenbrock
$f_3(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$
Schaffer's f6
$f_4(x) = 0.5 - \frac{(\sin\sqrt{x_1^2+x_2^2})^2 - 0.5}{(1+0.001(x_1^2+x_2^2))^2}$
Griewank
$f_5(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$

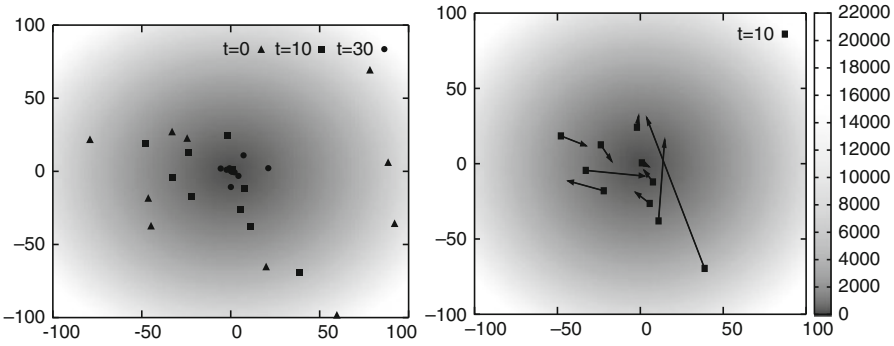

Fig. 8.3 Swarm on the two-dimensional Sphere function; particle positions at iterations $t \in \{0, 10, 30\}$ (*left*); particle positions and velocity vectors at iteration $t = 10$ (*right*)

8.3.2 Example 2: Discrete Binary PSO for Subset Problems

Subset problems are a broad class of optimization problems where the aim is to find a good subset of a given set of items. For many practical problems additional restrictions will be given so that not all subsets of the given set are valid. A difference to many permutation problems like the TSP is that subset problems allow solutions of different sizes. As an example subset problem we consider a problem from the financial sector where the earnings of a company have to be forecast. The forecast is based on financial ratios that are generated from the companies results and other economic indicators from the last quarters. We assume that a forecast method is given that computes for each financial ratio a forecast and the final forecast is the average of the forecasts for all given values. Since many different financial ratios are in use, e.g. the book value per share or the total growth rate of a company, the problem is to select a not-too-large subset of these so that the forecast method gives good forecasts when applied to the selected financial ratios.

To solve a subset problem with PSO a particle can be encoded by a D-dimensional vector where $D = |M|$ equals the size of the given set M (in the example M is the

set of all considered financial ratios). Each dimension represents one binary bit that determines whether the corresponding item respectively the corresponding financial ratio is selected to be member of the subset. The crucial part in the design of the PSO algorithm is to connect the continuous movement of the particles to the discrete solution space.

In the so-called discrete binary PSO algorithm this is done as follows (Kennedy and Eberhart 1997). Similar as for the continuous PSO, the position of a particle corresponds to a solution and the velocity has an influence on the new position. But how the position is computed is different. Since the solution space is discrete and a particle should not stay at the same place, a random component is used in the computation of the new position. The idea is to let a high velocity in one dimension give a high probability that the corresponding bit of the position vector is one.

Formally, the velocity of a particle is determined exactly as in Eq. (8.6). In order to determine the probabilities for the computation of the position vector a function is used that maps a velocity value onto the interval $[0,1]$. A function which is often used is $\text{sig}(v_{id}) = 1/(1+\exp(-v_{id}))$. To determine the dth bit of the position vector of particle i a random number r_{id} is drawn from the interval $[0,1]$ and the dth bit is set to one if $r_{id} < \text{sig}(v_{id})$ and otherwise it is set to zero.

The results of a comparative study between the discrete binary PSO and a GA for the financial ratio selection problem are presented in Ko and Lin (2004). It was shown for a problem of dimension $D = 64$ that PSO is faster and gives better results than the GA.

8.3.3 Advanced Features of PSO

Many variations of the velocity update in PSO and extensions of the standard PSO for a better control of the behavior of the swarm have been proposed in the literature (see Sedighizadeh and Masehian 2009 for a taxonomy of the different PSO variants). Only some extensions can reviewed in this section.

- *Adaptive PSO*. In Clerc (2002) a version of PSO was proposed where most values of the algorithms parameters are adapted automatically at run time. One example parameter is the swarm size that varied during execution. A particle is removed when it is the worst (with respect to the best solution found so far) of a neighborhood of particles and the best particle in its neighborhood has improved significantly since its creation. Other rules have been implemented for creating new particles. A PSO that combines ideas of various other adaptive PSOs and has several layers of adaptation has been proposed by Ritscher et al. (2010).
- *Neighborhood best velocity update*. Several PSO algorithms establish a neighborhood relation between particles. In that case, instead of using the global best position *gBest* for velocity update for each particle the best position of the particles in its neighborhood is used. This position is called neighborhood best and is denoted by *lBest*. A PSO variant where all particles in the neighborhood of a particle have an influence on its velocity was proposed by Kennedy and Mendes (2003).

The following formula describes such an-all-neighborhood-velocity-update of particle i with neighborhood N_i in dimension $d \in [1:D]$ (note that a particle is included in its own neighborhood):

$$v_{id} = w \cdot v_{id} + \sum_{j \in N_i} \frac{c \cdot r_j \cdot (p_{jd} - x_{id})}{|N_i|}. \tag{8.7}$$

A Hierarchical PSO where particles are particle within a hierarchy and neighborhood is defined by the place within the hierarchy has been proposed in Janson and Middendorf (2005).

- *Simplified PSO*. A PSO which does not use the personal best positions for velocity update was suggested by Kennedy (1997) and has been called *social only* PSO. A similar simplified PSO was investigated by Pedersen and Chipperfield (2010) and was shown to perform well for optimizing artificial neural network problems.
- *Maximum velocity*. A parameter $v_{\max}$ is introduced for some PSO algorithms to restrict the size of the elements v_{id} of the velocity vector so that $v_{id} \in [-v_{\max}, v_{\max}]$. Hence, when the velocity of a particle becomes larger than $v_{\max}$ during velocity update it is set to $v_{\max}$. A typical range for values of $v_{\max}$ is $[0.1 \cdot x_{max}, 1.0 \cdot x_{\max}]$. Observe that such values for $v_{\max}$ do not restrict the possible locations of a particle to $[-x_{\max}, x_{\max}]$.
- *Queen particle*. The addition of a queen particle, which is always located at the swarm's actual gravity center was proposed in Clerc (1999). Since the gravity center of the swarm might often be near a possible optimum, it is reasonable to evaluate the objective function at this location. Experimental results have shown that it depends on the type of function whether the introduction of a queen particle is advantageous.
- *Convergence enforcement*. Several authors have considered the problem of how to improve the rate of convergence of PSO.
  - A constriction coefficient K was introduced in Clerc and Kennedy (2002) to reduce undesirable explosive feedback effects where the average distance between the particles grows during an execution. With the constriction coefficient as computed in Kennedy and Eberhart (1999) and Kennedy et al. (2001) the formula for velocity update becomes

$$v_{id} = K \cdot (v_{id} + c_1 \cdot r_1 \cdot (p_{id} - x_{id}) + c_2 \cdot r_2 \cdot (p_{gd} - x_{id})).$$

Note that the constriction factor is just another way of choosing parameters w, c_1 and c_2. It can be shown that the swarm converges when parameter K is determined as (Kennedy and Eberhart 1999)

$$K = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|}$$

with $c = c_1 + c_2$, $c > 4$.

- Parameter w can be decreased over time during execution to diminish the diversity of the swarm and to reach faster a state of equilibrium. A linear decrease of w from a maximum value $w_{\max}$ to a minimum value $w_{\min}$ is used by several authors (e.g. Kennedy et al. 2001). Typical values are $w_{\max} = 0.9$ and $w_{\min} = 0.4$.
- In Vesterstrøm et al. (2002) the concept of division of labor and specialization was applied to PSO. Specialization to a task in this approach means for a particle to search near the global best position *gBest*. A particle that has not found a better solution for a longer time span is replaced by *gBest* in order to start searching around it. To prevent too many particles searching around *gBest* it was suggested to use a maximum number of particles that can switch to *gBest*. Note that a similar approach to let particles that have not found good solutions jump to the place of good particles is realized in the hybrid PSO, described below.

- *Controlling diversity*. To prevent the swarm from converging too early to a small area so that the particles become too similar, some methods have been proposed to maintain the diversity of the swarm. A common measure for the diversity of the swarm S in PSO is the "distance to average point":

$$\text{diversity}(S) := \frac{1}{|S|} \cdot \sum_{i=1}^{|S|} \sqrt{\sum_{j=1}^{D} (p_{ij} - \bar{p}_j)}$$

where $\bar{p}$ is the average vector of all *pBest* vectors $\mathbf{p}_i$. In order to make the diversity measure independent of the range of the search space some authors use the measure $\text{diversity}(S)/|L|$ where $|L|$ is the length of the longest diagonal in the search space. Some methods to keep the diversity of the swarm high enough are described in the following.

- Xie et al. (2002) proposed adding an additional random element to the movement of the particles. In the new algorithm called dissipative PSO (DPSO), immediately after velocity update and determination of the new position of the particles the following computations are done to introduce additional "chaos" to the system:

---

if $\text{rand}() < c_v$ **then**
  $v_{id} = \text{rand}() \cdot v_{\max,d}$ {chaos for velocity}
if $\text{rand}() < c_l$ **then**
  $x_{id} = \text{rand}(l_d, u_d)$ {chaos for location}

---

where $c_v, c_l \in [0,1]$ are parameters which control the probability to add chaos, $\text{rand}(a,b)$ is random number that is uniformly distributed in (a,b) ($\text{rand}()$ is a shortcut for $\text{rand}(0,1)$) and l_d, u_d are lower and upper bounds for the location in dimension d. Observe, that if $\text{rand}() < c_l$ the dth dimension of the new position is a random location within the search area.

  - The idea of using particles that have a spatial extension in the search space was established in Krink et al. (2002) to hinder particles from coming too close to each other and forming too dense clusters. In this variation of PSO particles that come too close to each other bounce away. Preliminary experimental results show that bouncing can be advantageous for complex objective functions. For objective functions with a single optimum, clustering is not a problem and bouncing is not advantageous. A similar approach to hinder the swarm from collapsing uses an analogy to electrostatic energy. In this approach, so-called charged particles experience a repulsive force when they come too close to each other (Blackwell and Bentley 2002). Swarms with different ratios of charged particles have been studied.
  - A strategy to explicitly control the diversity is to have two different phases of the PSO algorithm that can increase (repulsion phase) or reduce (attraction phase) the diversity of the swarm (Riget and Vesterstrøm 2002). Two threshold values have been introduced that are used to determine when an exchange between the two phases should take place. When the diversity becomes lower than the threshold d_{low} the algorithm switches to the repulsion phase and when the diversity becomes larger than threshold $d_{\text{high}} > d_{\text{low}}$ the algorithm changes to the attraction phase. The only thing that happens when the phase of the algorithm changes is that every velocity vector is changed so that it points in the opposite direction. The authors have shown by experiments that nearly all improvements of the global best solutions were found during the attraction phases. Therefore, they propose to do no function evaluations during the repulsion phase to reduce the run time of the algorithm.

- *Stagnation recovery*. For multi-modal functions there is the danger of premature convergence of standard PSO which results in suboptimal solutions. Stagnation recovery means detecting such a situation and reacting accordingly.

  - Re-initialization of the swarm is proposed in Clerc (1999) when the diameter of the area that is actively searched by the swarm has become too small. The new swarm is initialized around the previous best position.

8.3.4 Promising Areas for Future Applications of PSO

Complex multimodal functions that possess multiple and possibly similarly good local optimal solutions occur in many applications. Often in such applications it is not enough to know just a single of these local optimal solutions but several or all of them are needed. Two areas of future PSO research that are relevant for optimizing such complex multimodal functions are: (i) to find additional operations on the particles that can improve the optimization efficiency and (ii) to investigate how several swarms can work cooperatively. We briefly review some works in both areas. Other interesting application and research areas like multiobjective and dynamic op-

timization, hybrid algorithms, and theoretical aspects of PSO cannot be covered in this introductory tutorial.

8.3.4.1 Operations on Particles

Several operations on particles other than velocity update have been proposed in recent years. Examples are: (i) mutation operators that change the position of a particle, (ii) interpolation operators that try to find a new good position between the positions of several other particles, or (iii) operations that try to make particles behave similarly to particles in quantum mechanics. Algorithms using the latter type of operations are called quantum PSO algorithms (QPSOs) (Sun et al. 2004). In a QPSO the state of a particle is described by a wavefunction, instead of a position and a velocity. The wavefunction determines the probability of a particle appearing at a certain position.

8.3.4.2 Cooperative Swarms

Cooperative swarms have been introduced in order to divide the work between several swarms. One motivation is that it can be very difficult for a single swarm to solve problems with large dimension D. An example is the Cooperative Swarm Optimizer (CPSO) or Split Swarm that uses a set of swarms and splits the work equally between them in the following way (van den Bergh and Engelbrecht 2000). The vector to be optimized is split across the swarms so that each swarm optimizes a different part of the vector, i.e. the swarms optimize with respect to different dimensions of the search space. Cooperation between the swarms is established in that for every evaluation of a new position of some particle its partial vector is combined with one partial vector from each of the other swarms so that the quality of the resulting position is best. Experiments with CPSO for function minimization and neural network learning have shown that it is good for problems where the dependencies between the component vectors are not too strong. Another motivation to use cooperative swarms is for solving multiobjective problems where several functions have to be optimized so that the swarms optimize with respect to different functions. A problem is then to find good methods for exchanging information about the best positions between the swarms (see for example Parsopoulos et al. 2004).

A PSO method that intends to form subswarms of particles searching for the same local minimum is proposed in Kennedy (2000). A standard k-means cluster method was used to divide the swarm into several clusters of individuals. For velocity update then each particle i uses the center of its cluster instead of its personal best position p_{id} (see Eq. (8.6)). Test results have shown that this velocity update modification can be advantageous, especially for multimodal functions like the Rastrigin function (see Table 8.3).

Niching techniques can also be used to promote the formation of subswarms around different local optima. Niching is a concept that is inspired by the famous

observation known from ecology that coexisting species can survive because they occupy different niches, which roughly means that they have different tasks. A niching technique for PSO that aims to find all good local minima was proposed by Parsopoulos and Vrahatis (2001). It uses a function *stretching* method that changes the objective function during execution as follows. Assume that a position x has been found where the objective function f to be minimized has a small value. Then f is transformed with the aim to remove local minima that are larger than $f(x)$ and a subswarm is created that searches for a local minimum near x on the transformed function. In addition, a second transformation is applied to f which increases the function values in the neighborhood of x. This function is then used by the main swarm which will be repelled from the area around x and searches for a different local minimum. Another niching approach for PSO was proposed by Brits et al. (2002). The niching PSO starts with particles that move according to the so-called cognition-only model where velocity update is done only according to the personal best position of an individual (i.e. $c_2 = 0$ in Eq. (8.6)). The particles then basically perform local search. When the quality of a particle has not changed significantly for several iterations it is assumed that is has reached the region of a local minimum. To search for this minimum a subswarm is formed. Several rules are used to define how other particles can enter a subswarm or how subswarms with intersecting regions are merged.

8.4 Tricks of the Trade

For newcomers to the field of swarm intelligence it is an advantage that ACO and PSO algorithms are relatively easy to implement so that one can devise one's own practical experience without too much initial effort. Often even the standard form of ACO and PSO algorithms that do not use many problem-specific features work reasonably well for different types of optimization problems. This is especially true for certain types of problems, e.g. scheduling problems in the case of ACO and continuous function optimization in case of PSO. Clearly, such an early success should not lead to the illusion that swarm intelligence is a field where good algorithms can be obtained more or less for free, because principles are used that have been inspired by successful strategies which occur in nature. The following hints might be helpful for the newcomer to get a deeper understanding of swarm intelligence:

- Read those papers where swarm intelligence methods have been applied to problems that are similar to the problem you want to solve. Not less important is the study of good papers where you can learn about specific aspects of swarm intelligence methods or where carefully designed state-of-the-art algorithms are described.
- Preferably, do not start with too complicated an algorithm that you do not understand. Critically evaluate every step of your algorithm.
- Investigate how your algorithm behaves on different types of problem instances and try to verify your explanations. Test your algorithm on benchmark instances,

if available, to make comparisons with the works of other researchers easier. Ideally, use random instances and real-world instances for the tests. Random instances have the advantage that their properties can be characterized by their generation method. A disadvantage is that they are often too artificial to reflect important characteristics of real-world problems. In addition, carefully designed artificial problem instances can sometimes help to study special aspects of the behavior of algorithms.
- Investigate how robust your algorithm is with respect to changes of the parameters (e.g. parameters α, β, and ρ for ACO and parameters w, c_2 and c_2 for PSO).
- Consider the optimization behavior of your algorithm at different numbers of iterations. Then you can find out for example whether the algorithm converges too early.

For ACO the following hints should be considered:

- It is important to use pheromone information so that the ants are guided to good solutions. Two connected aspects are important here: (i) the pheromone should be used to encode properties of a solution that are most relevant in the sense that they can be used to characterize the good solutions, and (ii) the pheromone information should be interpreted by the ants in the best possible way.
- Find a solution construction process so that the ants make important decisions early on and so that they can use a good (deterministic) heuristic. Such heuristics can be found in the literature for many problems.

For PSO the following hint should be considered:

- For function optimization it is important to understand the characteristics of the search landscape of the application functions. When there is basically a single valley in the search space, a single swarm where convergence is enforced might work. But for search landscapes with many valleys a more sophisticated approach might be necessary, where the diversity of the swarm is controlled, stagnation recovery mechanisms are introduced, or several cooperative swarm are used.

8.5 Conclusion

The field of swarm intelligence with the vision to learn from the behavior of natural swarms for the development of new methods in optimization has produced with ACO and PSO two successful metaheuristics that have found a increasing number of applications in the last few years. The basic principles of swarm intelligence methods and a selection of example applications have been explained in this tutorial. New algorithmic principles, e.g. HBO, arise for swarm intelligence and there are also a number of new application areas where swarm intelligence will play its part. One promising concept are hybrid methods where swarm intelligence algorithms work in line with other metaheuristics. Hopefully, this tutorial may be a starting point for the reader to further explore the field of swarm intelligence.

Sources Of Additional Information

- Good introductory books that cover various aspects of Swarm Intelligence are: (i) Bonabeau, Dorigo, and Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, 1999, Oxford University Press, (ii) Kennedy, Eberhart, and Shi, *Swarm Intelligence*, 2001, Morgan Kaufmann.
  Recent overview papers are Blum (2005) and Dorigo and Blum (2005) on ACO and Banks et al. (2007) and Banks et al. (2008) on PSO.
- The following book is the ultimate reference book for ACO: Dorigo and Stützle, *Ant Colony Optimization*, 2004, MIT Press. A good reference book on PSO which also covers other parts of Swarm Intelligence is: Engelbrecht, *Fundamentals of Computational Swarm Intelligence*, 2005, Wiley. Anther recent book on PSO is: Parsopoulos and Vrahatis, *Particle Swarm Optimization and Intelligence: Advances and Applications*. 2010, Information Science Publishing (IGI Global).
- Two journals that are devoted entirely to the field of Swarm Intelligence are: *Swarm Intelligence*, Springer, and the *International Journal of Swarm Intelligence Research*, IGI Global.
- Recent special issues of journals that are devoted to Swarm Intelligence are
  - Special Issue on Swarm Intelligence Theory, *Theoretical Computer Science* 411(21), Bonabeau et al., guest editors, 2010.
  - Special Issue on Swarm Intelligence, *Natural Computing*, Bonabeau et al., guest editors, 2010.
  - Special Issue on Particle Swarm Optimization, *Swarm Intelligence* 3(4), Poli et al., guest editors, 2009.
  - Special Issue on Swarm Intelligence, *IEEE Transactions on Evolutionary Computation* 13(4), Engelbrecht et al., guest editors, 2009.
  - Special Issue on Ant Colony Optimization, *Swarm Intelligence* 3(1), Doerner et al., guest editors, 2009.
- A valuable source of recent research papers are the proceedings of the following conference series that focus on Swarm Intelligence: (i) International Conference on Ant Colony Optimization and Swarm Intelligence (ANTS), (ii) IEEE Swarm Intelligence Symposium (SIS) (the latest proceedings are Dorigo et al. 2008 and Kennedy and Shi 2009, respectively). Another new conference on the topic is the *International Conference on Swarm Intelligence* (ICSI).

References

Alrashidi MR, El-Hawary ME (2009) A survey of particle swarm optimization applications in electric power systems. IEEE Trans Evol Comput 13:913–918

Banks A, Vincent J, Anyakoha C (2007) A review of particle swarm optimization. Part I: background and development. Nat Comput 6:467–484

Banks A, Vincent J, Anyakoha C (2008) A review of particle swarm optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. Nat Comput 7:109–124

Bauer A, Bullnheimer B, Hartl RF, Strauss C (1999) An ant colony optimization approach for the single machine total tardiness problem. In: Proceedings of the CEC 1999, Washington, DC. IEEE, Piscataway, pp 1445–1450

Blackwell TM, Bentley PJ (2002) Dynamic search with charged swarms. In: GECCO 2002, New York. Morgan Kaufmann, San Mateo, pp 19–26

Blesa MJ, Blum C, Di Gaspero L, Roli A, Sampels M, Schaerf A (eds) (2009) In: 6th international workshop hybrid metaheuristics, Udine. LNCS 5818. Springer, Berlin

Blum C (2005) Ant colony optimization: introduction and recent trends. Phys Life Rev 2:353–373

Blum C, Sampels M (2002a) Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations. In: Proceedings of the CEC 2002, Honolulu, pp 1558–1563

Blum C, Sampels M (2002b) When model bias is stronger than selection pressure. In: Proceedings of the PPSN VII, Granada. LNCS 2439. Springer, Berlin, pp 893–902

Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, New York

Brits R, Engelbrecht AP, van den Bergh F (2002) A niching particle swarm optimizer. In: Proceedings of the SEAL 2002, Singapore, pp 692–696

Bullnheimer B, Hartl RF, Strauss CA (1999) New rank based version of the ant system: a computational study. Cent Eur J Oper Res Econ 7:25–38

Christensen A, O'Grady R, Dorigo M (2009) From fireflies to fault tolerant swarms of robots. IEEE Trans Evol Comput 13:754–766

Clerc M (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the CEC, Washington, DC. IEEE, Piscataway, pp 1951–1957

Clerc M (2002) Think locally, act locally—a framework for adaptive particle swarm optimizers. IEEE J Evol Comput 3:1951–1957

Clerc M, Kennedy J (2002) The particle swarm—explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evol Comput 6:58–73

Cordón O, Fernandez I, Herrera F, Moreno L (2000) A new ACO model integrating evolutionary computation concepts: the best-worst ant system. In: Proceedings of the 2nd international workshop on ant algorithms, Brussels, pp 22–29

del Valle Y, Venayagamoorthy GK, Mohagheghi S, Hernandez J-C, Harley RG (2008) Particle swarm optimization: basic concepts, variants and applications in power systems. IEEE Trans Evol Comput 12:171–195

Deneubourg J-L, Aron S, Goss S, Pasteels JM (1990) The self-organizing exploratory pattern of the Argentine ant. J Insect Behav 32:159–168

Diwold K, Beekman M, Middendorf M (2011) Honeybee optimisation. In: Panigrahi BK et al (eds) Handbook of swarm intelligence—concepts, principles and application. Springer, Berlin, pp 295–328

Dorigo M (1992) Optimization, learning and natural algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano
Dorigo M, Blum C (2005) Ant colony optimization theory: a survey. Theor Comput Sci 344:243–278
Dorigo M, Di Caro G (1999) The ant colony optimization meta-heuristic. In: Corne D et al (eds) New ideas in optimization. McGraw-Hill, New York, pp 11–32
Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1:53–66
Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy. Technical report 91-016, Politecnico di Milano
Dorigo M, Maniezzo V, Colorni A (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern B 26:29–41
Dorigo M, Birattari M, Blum C, Clerc M, Stützle T, Winfield AFT (eds) (2008) In: Proceedings of the ANTS 2008, Brussels. LNCS 5217. Springer, Berlin
Gambardella LM, Taillard E, Dorigo M (1999) Ant colonies for the quadratic assignment problem. J Oper Res Soc 50:167–176
Goss S, Aron S, Deneubourg JL, Pasteels JM (1989) Self-organized shortcuts in the Argentine ant. Naturwissenschaften 76:579–581
Guntsch M, Middendorf M (2002a) Applying population based ACO to dynamic optimization problems. In: Proceedings of the 3rd international workshop ANTS 2002, Brussels. LNCS 2463. Springer, Berlin, pp 111–122
Guntsch M, Middendorf M (2002b) A population based approach for ACO. In: Proceedings of the EvoWorkshops 2002 on applications of evolutionary computing, Kinsale. LNCS 2279. Springer, Berlin, pp 72–81
Gutjahr WJ (2011) Ant colony optimization: recent developments in theoretical analysis. In: Auger A, Doerr B (eds) Theory of randomized search heuristics. World Scientific, Singapore, pp 225–254
Handl J, Meyer B (2002) Improved ant-based clustering and sorting in a document retrieval interface. In: Merelo Guervos JJ et al (eds) Proceedings of the PPSN VII, Granada. LNCS 2439. Springer, Berlin, pp 913–923
OR-Library (2012). http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html
Janson S, Middendorf M (2005) A hierarchical particle swarm optimizer and its adaptive variant. IEEE Syst Man Cybern B 32:1272–1282
Karaboga D, Akay B (2009) A survey: algorithms simulating bee swarm intelligence. Artif Intell Rev 31:61–85
Kawamura H, Yamamoto M, Suzuki K, Ohucke A (2000) Multiple ant colonies algorithm based on colony level interactions. IEICE Trans Fundam 83A:371–379
Kennedy J (1997) The particle swarm: social adaptation of knowledge. In: Proceedings of the CEC, Indianapolis, pp 303–308
Kennedy J (2000) Stereotyping: improving particle swarm performance with cluster analysis. In: Proceedings of the CEC, La Jolla, pp 1507–1512
Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Perth, pp 1942–1948
Kennedy J, Eberhart RC (1997) A discrete binary version of the particle swarm algorithm. Proc Conf Syst Man Cybern 5:4104–4109. IEEE, Piscataway

Kennedy J, Eberhart RC (1999) The particle swarm: social adaption in information processing systems. In: Corne D et al (eds) New ideas in optimization. McGraw-Hill, New York, pp 379–387

Kennedy J, Mendes R (2003) Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. In: Proceedings of the IEEE international workshop on soft computing in industrial applications, New York

Kennedy J, Shi Y (eds) (2009) In: Proceedings of the 2009 IEEE Swarm Intelligence Symposium, Nashville, IEEE

Kennedy J, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann, San Francisco

Ko P-C, Lin P-C (2004) A hybrid swarm intelligence based mechanism for earning forecast. In: Proceedings of the ICITA 2004, Harbin

Krink T, Vesterstrøm JS, Riget J (2002) Particle swarm optimisation with spatial particle extension. In: Proceedings of the CEC 2002, Honolulu, pp 1474–1479

Labella TH, Dorigo M, Deneubourg J-L (2006) Division of labour in a group of robots inspired by ants' foraging behaviour. ACM Trans Auton Adapt Syst 1:4–25

Lumer ED, Faieta B (1994) Diversity and adaptation in populations of clustering ants. In: Proceedings of the SAB 1994, Brighton. MIT, Cambridge, pp 501–508

Maniezzo V (1999) Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. Inf J Comput 11:358–369

Merkle D, Middendorf M (2002) Ant colony optimization with the relative pheromone evaluation method. In: Proceedings of the EvoWorkshops 2001, Como. LNCS 2279. Springer, Berlin, pp 325–333

Merkle D, Middendorf M (2003a) On the behavior of ACO algorithms: studies on simple problems. In: Resende MGC, Pinho de Sousa J (eds) Metaheuristics: computer decision-making. Kluwer, Dordrecht, pp 465–480

Merkle D, Middendorf M (2003b) An ant algorithm with global pheromone evaluation for scheduling a single machine. Appl Intell 18:105–111

Merkle D, Middendorf M (2005) On solving permutation scheduling problems with ant colony optimization. Int J Syst Sci 36:255–266

Merkle D, Middendorf M (2008) Swarm intelligence and signal processing. IEEE Signal Process Mag 25:152–158

Merkle D, Middendorf M, Schmeck H (2002) Ant colony optimization for resource-constrained project scheduling. IEEE Trans Evol Comput 6:333–346

Michels R, Middendorf M (1999) An ant system for the shortest common supersequence problem. In: Corne D, Dorigo M, Glover F (eds) New ideas in optimization. McGraw-Hill, New York, pp 51–61

Montgomery J, Randall M (2002) Anti-pheromone as a tool for better exploration of search space. In: Proceedings of the ANTS 2002, Brussels. LNCS 2463. Springer, Berlin, pp 100–110

Oliveira SM, Hussin MS, Stützle T, Roli A, Dorigo M (2011) A detailed analysis of the population-based ant colony optimization algorithm for the TSP and the QAP. In: GECCO (Companion), Dublin, pp 13–14

Parsopoulos KE, Vrahatis MN (2001) Modification of the particle swarm optimizer for locating all the global minima. In: Kurkova V et al (eds) Artificial neural networks and genetic algorithms. Springer, Berlin, pp 324–327

Parsopoulos KE, Tasoulis DK, Vrahatis MN (2004) Multiobjective optimization using parallel vector evaluated particle swarm optimization. In: Proceedings of the IASTED international conference on artificial intelligence and applications, Innsbruck

Pedersen MEH, Chipperfield AJ (2010) Simplifying particle swarm optimization. Appl Soft Comput 10:618–628

Poli R (2008) Analysis of the publications on the applications of particle swarm optimisation. J Artif Evol Appl 1:1–10

Riget J, Vesterstrøm JS (2002) A diversity-guided particle swarm optimizer—the ARPSO. Technical report no 2002-02, University of Aarhus

Ritscher T, Helwig S, Wanka R (2010) Design and experimental evaluation of multiple adaptation layers in self-optimizing particle swarm optimization. In: Proceedings of the CEC 2010, Barcelona, pp 1–8

Sedighizadeh D, Masehian E (2009) Particle swarm optimization methods, taxonomy and applications. Int J Comput Theor Eng 1:1793–8201

Stützle T, Hoos H (1997) Improvements on the ant system: introducing MAX(MIN) ant system. In: Proceedings of the international conference on artificial neutral networks and genetic algorithms. Springer, Berlin, pp 245–249

Stützle T, Hoos H (2000) MAX-MIN ant system. Future Gener Comput Syst J 16:889–914

Stützle T, den Besten M, Dorigo M (2000) Ant colony optimization for the total weighted tardiness problem. In: Deb et al (eds) Proceedings of the PPSN-VI, Paris. LNCS 1917. Springer, Berlin, pp 611–620

Sumpter DJT (2009) Collective animal behavior. Princeton University Press, Princeton

Sun J, Feng B, Xu W (2004) Particle swarm optimization with particles having quantum behavior. In: IEEE Proceeding of the CEC, San Diego, pp 325–331

van den Bergh F, Engelbrecht AP (2000) Cooperative learning in neural networks using particle swarm optimizers. S Afr Comput J 26:84–90

Vesterstrøm JS, Riget J, Krink T (2002) Division of labor in particle swarm optimisation. In: Proceedings of the CEC 2002, Honolulu, pp 1570–1575

Xie X-F, Zhang W-J, Yang Z-L (2002) A dissipative particle swarm optimization. In: Proceedings of the CEC 2002, Honolulu