Chapter 14

SWARM INTELLIGENCE

Daniel Merkle and Martin Middendorf
Department of Computer Science
University of Leipzig, Germany

14.1 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 (see e.g. 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 (Camazine et al., 2001). Examples where insights into the behavior of natural swarms has influenced the design of algorithms and systems in computer science include the following (see Bonabeau et al., 1999; Middendorf, 2002 for more information):

- Collective transport of ants has inspired the design of controllers of robots for doing coordinated work Kube and Bonabeau, 2000.

- Brood sorting behavior of ants motivated several clustering and sorting algorithms (e.g., Handl and Meyer, 2002; Lumer and Faieta, 1994).

- The path-finding and orientation skills of the desert ant *Cataglyphis* were used as an archetype for building a robot orientation unit Lambrinos et al., 2000.

- Models for the division of labor between members of an ant colony were used to regulate the joint work of robots (e.g. Agassounoun et al., 2001; Goldberg and Mataric, 2000).

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).

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 a pheromone trail 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 Kennedy et al., 2001. 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.

14.2 ANT COLONY OPTIMIZATION

A renowned 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. 14.2). 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.

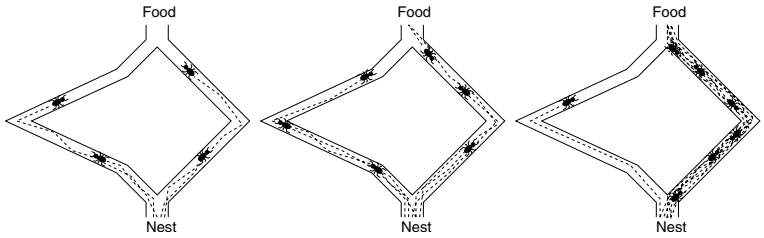

Figure 14.1. Double bridge experiment.

Similar to the experiment with branches 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), see Dorigo, 1992; Dorigo et al., 1991, and initiated the field of ACO. In recent years this field of research has become quite rich and ACO algorithms have now been designed for various application problems and different types of combinatorial optimization problems including dynamic and multi-objective optimization problems (see Cordón et al., 2002; Maniezzo et al., 2001; Stützle and Dorigo, 2002b for other overviews). Some papers treat the theory of ACO and modeling of ACO algorithms (Gutjahr, 2000; Gutjahr, 2002; Merkle and Middendorf, 2002a; Stützle and Dorigo, 2002a). 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

In what follows 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.

14.2.1 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, the TSP being a well known example. After 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, and stopping criterion.

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 contains 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.

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 the 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

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 about 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, for example by setting $\eta_{ij} := 1/d_{ij}$. The probability distribution when using a heuristic is

$$p_{ij} := \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{z \in S} \tau_{iz}^{\alpha} \cdot \eta_{iz}^{\beta}} \quad \forall j \in S \tag{14.1}$$

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

In order to better 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. (14.1).

Table 14.1. 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)

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 $\Delta > 0$:

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

Stopping Criterion The ACO algorithm executes a number of iterations until a specified stopping criterion has been met. 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.

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 14.1.

14.2.2 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 that was proposed recently is population-based ACO (P-ACO) (see Guntsch and Middendorf, 2002b). One idea of P-ACO is to transfer less and only the most important information from one iteration to the next. This is done in the form of a small population of good solutions. In this section we describe the differences between P-ACO and standard ACO for permutation problems. It was shown that both approaches show a similar performance on the TSP Guntsch and Middendorf, 2002b. A scheme of a P-ACO algorithm for the TSP is given in the following (compare with the scheme of ACO-TSP).

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

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. This matrix is called the pop-

ulation 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.

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.

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 $\tau_{\text{init}} > 0$ and is increased, if there are corresponding solutions in the population:

$$\tau_{ij} := \tau_{\text{init}} + \zeta_{ij} \cdot \Delta \tag{14.2}$$

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 $\tau_{\text{init}}, \tau_{\text{init}} + \Delta, \ldots, \tau_{\text{init}} + k \cdot \Delta$ (when using an elitist solution $\tau_{\text{init}} + (k+1) \cdot \Delta$ 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.

14.2.3 Example 3: ACO for a Scheduling Problem

In this section the ACO approach is applied to a scheduling permutation problem which is 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. 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:

A Pheromone information

- Pheromone encoding

B Solution construction

- Pheromone evaluation
- Adaptation of heuristics

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 $\sum_{j=1}^{n} w_j T_j$.

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 next to 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. This pheromone matrix is of type place×item whereas the pheromone matrix used for the TSP is of type item×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.

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, see Bauer et al., 1999). An ant that has to decide which job is on the next place i in the permutation considers all values τ_{ij}, $j \in S$ which indicate how good the selectable jobs have performed on 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):

$$p_{ij} = \frac{\left(\sum_{l=1}^{i} \tau_{lj}\right)^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{z \in S} (\sum_{l=1}^{i} \tau_{lz})^{\alpha} \cdot \eta_{iz}^{\beta}} \quad \forall j \in S \tag{14.3}$$

So far the potential of global evaluation has not been fully recognized because nearly all ACO algorithms use only local evaluation (see Bautista and Pereira, 2002; Merkle et al., 2002; Merkle and Middendorf, 2002b; Merkle and Middendorf, 2003a for other applications or other global evaluation methods).

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 see Merkle and Middendorf, 2003a; another investigation of ACO on simple problems is Stützle

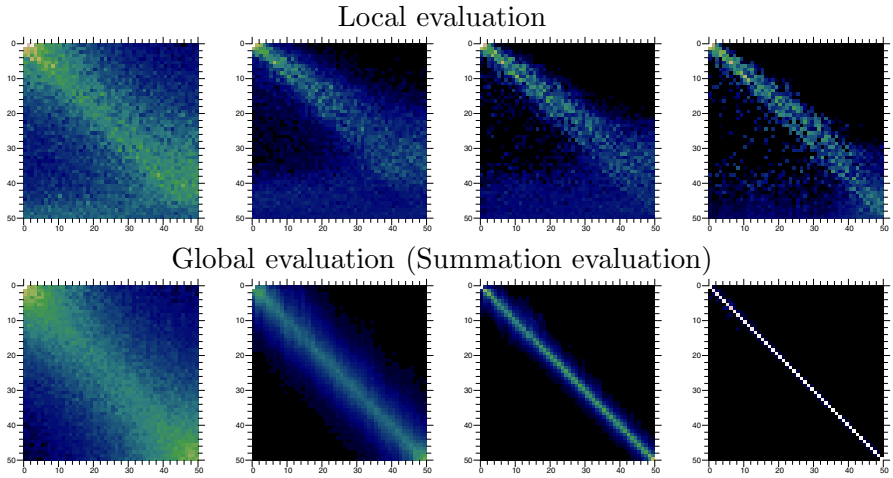

Figure 14.2. Comparison between SMTWTP-ACO with local evaluation and summation evaluation: Pheromone matrices averaged over 25 runs in iterations 800, 1500, 2200, and 2900: brighter gray colors indicate higher pheromone values.

and Dorigo, 2001). It consists of 50 jobs where job i, $i \in [1:50]$ has processing time $p_i = 1$, due date $d_i = i$, and weight $w_i = 1$. Clearly, to place job i, $i = 1, \ldots, n$ on place i is the only optimal solution with costs 0. Figure 14.2 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. 14.2 show a very symmetric behavior and do not have the undesired property that some of the jobs with small number are scheduled very late.

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 (MDD) rule, i.e.

$$\eta_{ij} = \frac{1}{\max\{\mathcal{T} + p_j, d_j\}} \tag{14.4}$$

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

Table 14.2. 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).

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

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): for example,

$$\eta_{ij} = \frac{1}{\max\{\mathcal{T} + p_j, d_j\} - \mathcal{T}} \tag{14.5}$$

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, 2004 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 14.2 compares the behavior of the algorithm using non-adapted heuristic (14.4) and local pheromone evaluation with the algorithms that use one or both of adapted heuristic (14.5) and global pheromone evaluation. The results clearly show that using an adapted heuristic (14.5) or the global pheromone evaluation improves the results significantly and using both is best.

14.2.4 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.

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 is 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., 1998.

- *Elitist solution pheromone update.* It can be advantageous to enforce the influence of the best solution π^e 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 of this method is that often a decision which can lead to bad solutions can also lead to a very good solution. In that case there is a danger that the corresponding pheromone value decreases too fast so that the very good solution is not found by the ants.

- *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 done (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 then 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).

Other ACO Variants Several variants of ACO algorithms which do not regard 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. Only if no city in the candidate list is selectable is one of the other cities from the selection set S chosen.

- *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 in 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. In Stützle and Hoos, 1997 it was suggested to increase the pheromone values proportionately to their difference to the maximum pheromone

value. A temporary reduction of α to negative values was proposed in Randall and Tonkes, 2002.

- *Changing α, β values.* In Merkle et al., 2002 it was proposed to reduce the value of β during a run to increase the influence of the pheromone at later stages of the algorithm. See "Stagnation recovery" for changing α values.

- *Repelling pheromone.* Some experiments with pheromone that let the ants avoid choosing an edge have been conducted in Kawamura et al., 2000; Montgomery and Randall, 2002 in order to enforce ants (or different colonies of ants) to search in different regions of the search space. A similar idea has also been applied for an ant-based network routing algorithm Amin et al., to appear; Schoonderwoerd et al., 1996.

- *Moving direction.* The use of ants that "move in different directions" can improve the optimization behavior Michels and Middendorf, 1999. For an example for a permutation problem this could mean that some ants decide first which item is on place one of the permutation and other ants decide first which item is on the last place. One aspect is that the ants should make important decisions early Merkle and Middendorf, 2001a. For some permutation problems where an unwanted bias in the decision of the ants can occur it can be advantageous to let the ants decide randomly about the sequence in which the places of the permutation are fixed (see Merkle and Middendorf, 2001a for details).

- *Local improvement of solutions.* The use of local optimization strategies to improve the solutions that have been found by the ants has been applied quite successfully for many ACO algorithms (e.g., Dorigo and Gambardella, 1997; Stützle et al., 2000; Stützle and Dorigo, 1999). 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) as (i), but the pheromone update is done according to the solution after local improvement.

14.2.5 Some Promising Areas for Future Application of ACO

An important area of research for ACO that is often underestimated in its practical importance is to gain a deeper understanding how the use of different pheromone models influences the optimization behavior. Other promising fields, like multiobjective optimization, dynamic and probabilistic optimization, hybrid algorithms, and theoretical aspects, cannot be covered in this introductory tutorial.

Pheromones and Optimization Behavior of ACO A few works which consider this aspect have already been mentioned. Some recent papers have focused on the investigation of pheromone models. In Dorigo et al., 2002b pheromone update rules for ACO are systematically derived based on the stochastic gradient ascent algorithm and cross-entropy method. A deterministic model for ACO is proposed in Merkle and Middendorf, 2002a and used to explain the dynamic change of pheromone values based on fixed-point analysis. In Blum and Sampels, 2002b; Merkle and Middendorf, 2001a; Merkle and Middendorf, 2001b it is investigated how the pheromone model can introduce a strong bias to some regions of the search space.

14.3 PARTICLE SWARM OPTIMIZATION

The roots of the metaheuristic that is described in this section lie 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 lets 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 (PSO) 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 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 (neighboring) 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

An active field of research on PSO has developed, with the main use of PSO being for continuous function optimization. An increasing number of works have begun to investigate the use of PSO algorithms as function optimizers embedded into more complex application contexts. Examples are the use of PSO for neural network training (van den Bergh and Engelbrecht, 2000; Conradie et al., 2002), gene clustering Xiao et al., 2003, power systems (Yoshida et al., 2000), and multimodal biometric systems (Veeramachaneni and Osadciw, 2003). Some works apply PSO also to discrete problems, such as subset problems (Kennedy and Eber-

hart, 1997; Ko and Lin, 2004) and permutation problems (see Hu et al., 2003).

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 function optimization. A subset problem is addressed in the second example to illustrate how PSO can be applied to other types of optimization problems.

14.3.1 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 $\vec{x}_i = (x_{i1}, \ldots, x_{iD})$ and the velocity of the particle by the vector $\vec{v}_i = (v_{i1}, \ldots, v_{iD})$. 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 $\vec{p}_i = (p_{i1}, \ldots p_{iD})$ and is called *pBest*. The index of the particle with the so far best found position in the swarm is denoted by g and $\vec{p}_g$ 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 $f(\vec{x}_i) < f(\vec{p}_i)$ then set $\vec{p}_i = \vec{x}_i$. If $f(\vec{p}_i) < f(\vec{p}_g)$ 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:

$$v_{id} = w \cdot v_{id} + c_1 \cdot r_1 \cdot (p_{id} - x_{id}) + c_2 \cdot r_2 \cdot (p_{gd} - x_{id}) \tag{14.6}$$

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 near 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 $[0, 1]$.

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	$f_5(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$

Table 14.3. Test functions.

After velocity update the new position of the particle i is then 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 14.3. 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 dimensions D of 10–100.

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 $w = 0.729$, $c1 = c2 = 1.494$). It can be seen from Fig. 14.3 (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. 14.3 (right).

14.3.2 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. Unlike many permutation problems like the TSP, 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

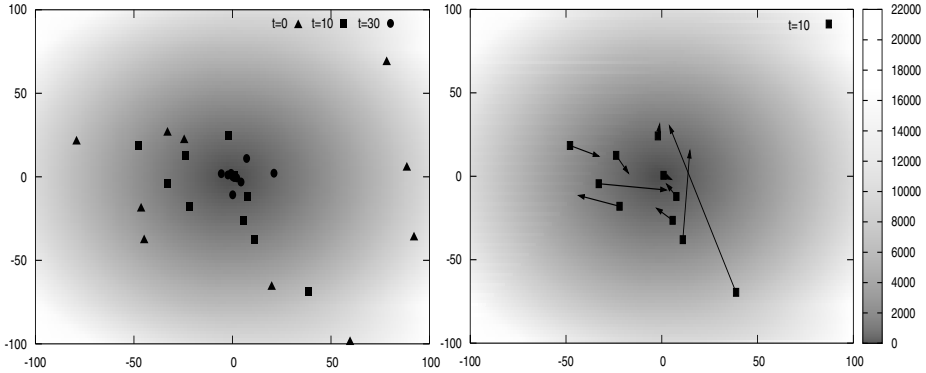

Figure 14.3. Swarm on the two-dimensional Sphere function; particles positions at iterations $t \in \{0, 10, 30\}$ (left); particles positions and velocity vectors at iteration $t = 10$ (right).

financial ratios that are generated from the company's 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 an individual of the swarm 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. 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. (14.6). 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 often used is

$$\text{sig}(v_{id}) = \frac{1}{1 + \exp(-v_{id})} \tag{14.7}$$

To determine then the ith bit of the position vector of particle d a random number r_{id} is drawn from the interval $[0, 1]$ and the ith bit is set to one if $r_{id} < sig(v_{id})$ and otherwise it is set to zero.

The results of a comparative study between the discrete binary PSO and a genetic algorithm 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 genetic algorithm. Another application of discrete binary PSO to data mining can be found in Sousa et al., 2003. The problem was to select a good subset of a given set of classification rules so that certain data can be classified accordingly.

14.3.3 Advanced Features of PSO

Several 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. Some of them are reviewed in this section.

- *Adaptive PSO.* In Clerc, 2002 a version of PSO has been proposed where most values of the algorithms parameters are adapted automatically at run time. One example 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. Experimental results reported in Parsopoulos and Vrahatis, 2002 have shown that an approach to use a second PSO during run time for determining the best parameter values for the first PSO were not successful. A certain evolutionary algorithm (Differential Evolution algorithm) performed better for this purpose.

- *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 is proposed in 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):

$$v_{id} = w \cdot v_{id} + \sum_{j \in N_i} \frac{c \cdot r_j \cdot (p_{jd} - x_{id})}{|N_i|} \tag{14.8}$$

- *Maximum velocity.* A parameter $v_{\max}$ is introduced for some PSO algorithms to restrict the size of the elements v_{ij} of the velocity vector so that $v_{ij} \in [-v_{\max}, v_{\max}]$. 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; Kennedy et al., 2001 the formula for velocity update becomes

    $$v_{id} = K \cdot (v_{id} + c_1 \cdot r_1 \cdot (p_{id} - x_{id}) + c_2 \cdot r_2 \cdot (p_{gd} - x_{id})) \tag{14.9}$$

    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 (see Kennedy and Eberhart, 1999)

$$K = \frac{2}{|2 - c - \sqrt{c^2 - 4c}|} \tag{14.10}$$

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 more quickly reach 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. A particle that has not found a better solution for a longer time span is replaced to the global best solution in order to start searching around the global best solution *gBest*. To prevent too many particles searching around *gBest* it was suggested to use a maximum number of particles that can switch to *gBest* or to make it more difficult to switch to *gBest* for those particles that are far away from *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 too early convergence to a small area so that the particles become too similar some methods have been proposed to keep the diversity of the swarm high enough. A common measure for the diversity of the swarm S in PSO is the "distance-to-average-point"

$$\text{diversity}(S) := \frac{1}{|S|} \cdot \sum_{i=1}^{|S|} \sqrt{\sum_{j=1}^{D} (p_{ij} - \bar{p}_j)} \tag{14.11}$$

where $\bar{p}$ is the average vector of all vectors $\vec{p}_i$. 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 performed to introduce additional "chaos" to the system:

$$\begin{aligned}&\textbf{if } rand() < c_v \textbf{ then}\\ &\quad v_{id} = rand() \cdot v_{\max,d} \text{ \{chaos for velocity\}}\\ &\textbf{if } rand() < c_l \textbf{ then}\\ &\quad x_{id} = rand(l_d, u_d) \text{ \{chaos for location\}}\end{aligned}$$

where $c_v, c_l \in [0, 1]$ are parameters which control the probability to add chaos, $rand(a, b)$ is random number that is uniformly distributed in (a, b) ($rand()$ is a shortcut for $rand(0, 1)$) and l_d, u_d are lower and upper bounds for the location in dimension d. Observe, that if $rand() < c_l$ 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 introduced 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. A problem is to choose the direction in which the bouncing particles move away. Three strategies for defining the bouncing direction have been studied: (i) random bouncing, where the particles move in a random direction, (ii) physical bouncing, in which the particles bounce like physical objects, and (iii) velocity line-bouncing, in which the particles do not change their direction but move with increased speed. 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 to collapse 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 two 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 $d_{high} > d_{low}$ 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 to detect such a situation and then to react 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.

14.3.4 Some Promising Areas for Future Application of PSO

Complex multimodal functions that possess multiple, 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 complex multimodal functions are: (i) to find good neighborhood relations between the particles that might lead to an increased optimization efficiency and (ii) to investigate how several swarms can work co-operatively. We briefly review some works in both areas. Other interesting application and research areas, such as multiobjective and dynamic optimization, hybrid algorithms, and theoretical aspects of PSO, cannot be covered by this introductory tutorial.

Neighborhood Relations One possible advantage of neighborhood relations between the particles is an increased efficiency of PSO

because the particles have to react only with respect to their neighbors. Another advantage is that the introduction of neighborhood relations can support the specialization of subsets of particles to different parts of the search space. A neighborhood scheme is explored in Suganthan, 1999 that is defined by a particle's actual position so that a certain number of the closest other particles are considered to be neighbors. A different approach is to define the neighborhood independently from the particles' positions. In Kennedy and Mendes, 2003 several such fixed neighborhoods are examined. For example, the particles are mapped onto a grid that defines the neighborhood. A hierarchical PSO where the particles of the swarm are arranged in a dynamic hierarchy that depends on the quality of the actual positions of the particles and defines the neighborhood structure was proposed in Janson and Middendorf, 2003.

Co-operative Swarms Co-operative swarms have been introduced 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 co-operative 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. Co-operation 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 others 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 dependences between the component vectors are not too strong. Another motivation to use co-operative 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 e.g. 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. (14.6). Test results have shown that this velocity update modification can be advan-

tageous, especially for multimodal functions like the Rastrigin function (see Table 14.3).

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 well known observation from ecology that coexisting species can survive because they occupy different niches, which roughly means that they have different tasks. Various niching techniques have been developed for genetic algorithms but meanwhile some authors have used niching also for PSO. A niching technique for PSO that aims to find all good local minima was proposed in 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 in 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. (14.6). 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. At creation time the subswarm consists only of two particles, the founding particle and its closest neighbor in the search space. Each subswarm is assigned a search region (initially all positions that are not further away from the founding particle as its closest neighbor). Several rules are used to define how other particles can enter a subswarm or how subswarms with intersecting regions are merged.

14.4 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 develop practical experience without too much 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: for example, scheduling problems in the case of ACO, and continuous function optimization in the case of PSO. Clearly, such 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 may help the newcomer arrive at a deeper understanding of Swarm Intelligence.

- Read papers which apply Swarm Intelligence methods to problems that are similar to the problem you want to solve. No less important is to also study other good papers to 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 problem. 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. the α, β, and ρ parameters for ACO and the w, c_2, and c_2 parameters for PSO).
- Consider the optimization behavior of your algorithm at different numbers of iterations. Then you can discover, 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 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 (see Chapter 19) 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 co-operative swarm are used.

14.5 CONCLUSIONS

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 an 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. A number of new application areas is emerging in which Swarm Intelligence will play its part. One promising concept are hybrid methods where swarm intelligence algorithms work in line with other metaheuristics. Might this tutorial also 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
  - E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems,* 1999, Oxford University Press, New York.
  - Kennedy, J., R. C. Eberhart and Y. Shi, *Swarm Intelligence,* 2001, Morgan Kaufmann, San Mateo, CA.

  Recent overview papers are Cordón et al., 2002; Maniezzo et al., 2001; Stützle and Dorigo, 2002b on ACO and van den Bergh, 2002 on PSO.

- The following book will become the ultimate reference book for ACO: M. Dorigo and T. Stützle, *Ant Colony Optimization,* 2004, MIT Press, Boston, MA.

- A valid source of information for optimization techniques in general that contains four chapters on ACO and one chapter on PSO is: D. Corne, M. Dorigo, and F. Glover (eds), *New Ideas in Optimisation,* 1999, McGraw-Hill, New York.

- Special issues of journals that are devoted to Swarm Intelligence are

  - Special section on Ant Algorithms and Swarm Intelligence in *IEEE Transactions on Evolutionary Computation* 6(4), M. Dorigo, L. Gambardella, M. Middendorf and T. Stützle, guest editors, 2002.
  - Special issue on Ant Colony Optimization, *Mathware & Soft Computing* 9, O. Cordon, F. Herrera and T. Stützle, guest editors, 2002.
  - Special issue on Ant Algorithms, *Future Generation Computer Systems Journal* 16(8), M. Dorigo, G. Di Caro, and T. Stützle, guest editors, 2000.

- A valuable source of recent research papers are the proceedings of the following workshop series that focus on Swarm Intelligence: International Workshop on Ant Colony Optimization and Swarm Intelligence (ANTS); IEEE Swarm Intelligence Symposium (the latest proceedings are Dorigo et al., 2002a and Eberhart et al., 2003, respectively).

References

Agassounoun W., Martinoli, A. and Goodman, R., 2001, A scalable, distributed algorithm for allocating workers in embedded systems, in: *Proc. 2001 IEEE Systems, Man and Cybernetics Conference,* pp. 3367–3373.

Amin, K. A., Mikler, A. R. and Iyengar, P. V., Dynamic agent population in agent-based distance vector routing, *J. Neural Parallel Sci. Comput.* **11**:127–142.

Bauer, A., Bullnheimer, B., Hartl, R. F. and Strauss, C., 1999, An ant colony optimization approach for the single machine total tardiness problem, in: *Proc. CEC'99*, IEEE Press, Piscataway, NJ, pp. 1445–1450.

Bautista J. and Pereira, J., 2002, Ant algorithms for assembly line balancing, in: *Proc. 3rd Int. Workshop on Ant Algorithms,* M. Dorigo et al., eds, Lecture Notes in Computer Science, Vol. 2463, Springer, Berlin, pp. 65–75.

van den Bergh, F., 2002, An analysis of particle swarm optimizers, *Ph.D. Thesis,* Department of Computer Science, University of Pretoria, South Africa.

van den Bergh, F. and Engelbrecht, A. P., 2000, Cooperative learning in neural networks using particle swarm optimizers, *S. African Comput. J.* **26**:84–90.

Blackwell, T. M. and Bentley, P. J., 2002, Dynamic search with charged swarms, *Proc. Genetic and Evolutionary Computation Conference 2002 (GECCO 2002),* Morgan Kaufmann, San Mateo, CA, pp. 19–26.

Blum C. and Sampels, M., 2002a, Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations, in: *Proc. 2002 Congress on Evolutionary Computation (CEC'02),* pp. 1558–1563.

Blum C. and Sampels, M., 2002b, When model bias is stronger than selection pressure. *Proc. 7th Int. Conference on Parallel Problem Solving from Nature (PPSN VII),* Lecture Notes in Computer Science, Vol. 2439, Springer, Berlin, pp. 893–902.

Bonabeau, E., Dorigo, M. and Theraulaz, G., 1999, *Swarm Intelligence: From Natural to Artificial Systems,* Oxford University Press, Oxford.

Brits, R., Engelbrecht, A. P. and van den Bergh, F., 2002, A niching particle swarm optimizer, in: *Proc. 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL 2002),* pp. 692–696.

Bullnheimer, B., Hartl, R. F. and Strau, C. A., 1999, New rank based version of the Ant System: a computational study, *Central Eur. J. Oper. Res. Econ.* **7**:25–38.

Camazine, S., Franks, N. R., and Deneubourg, J.-L., 2001, *Self-Organization in Biological Systems,* Princeton Studies in Complexity, Princeton University Press, Princeton, NJ.

Clerc, M., 1999, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: *Proc. Congress of Evolutionary Computation*, IEEE Press, Piscataway, NJ, pp. 1951–1957.

Clerc, M., 2002, Think locally, act locally—a framework for adaptive particle swarm optimizers, *IEEE J. Evol. Comput.*, submitted.

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

Conradie, A., Miikkulainen, R., and Aldrich, C., 2002, Adaptive control utilizing neural swarming, *Proc. Genetic and Evolutionary Computa-*

tion Conference (GECCO 2002), Morgan Kaufmann, San Mateo, CA, pp. 60–67.

Cordón, O., Fernandez, I., Herrera, F. and Moreno, L., 2000, A new ACO model integrating evolutionary computation concepts: the best–worst ant system, in: *Proc. 2nd Int. Workshop on Ant Algorithms,* pp. 22–29.

Cordon, O., Herrera, F. and Stützle, T., 2002, A review on the ant colony optimization metaheuristic: basis, models and new trends, *Mathware Soft Comput.***9**:141–175.

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

Dorigo, M., 1992, Optimization, learning and natural algorithms (in Italian), *Ph.D. Thesis,* Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo, M., and Di Caro, G., 1999, The ant colony optimization meta-heuristic, in: *New Ideas in Optimization,* D. Corne, M. Dorigo, and F. Glover, eds, McGraw-Hill, New York, pp. 11–32.

Dorigo, M., Di Caro, G. and Sampels, M., eds., 2002a, *Proc. 3rd Int. Workshop on Ant Algorithms (ANTS 2002),* Lecture Notes in Computer Science, Vol. 2463, Springer, Berlin.

Dorigo, M. and Gambardella, L. M., 1997, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evol. Comput.* **1**:53–66.

Dorigo, M., Maniezzo, V. and Colorni, A., 1991, Positive feedback as a search strategy, *Technical Report* 91-016, Politecnico di Milano, Italy.

Dorigo, M., Maniezzo, V., and Colorni, A., 1996, The ant system: optimization by a colony of cooperating agents, *IEEE Trans. Syst., Man Cybernet.* B **26**:29–41.

Dorigo, M., Zlochin, M., Meuleau, N. and Birattari, M., 2002b, Updating ACO pheromones using stochastic gradient ascent and cross-entropy methods, in: *Proc. of the EvoWorkshops 2002,* S. Cagnoni et al., eds., Lecture Notes in Computer Science, Vol. 2279, Springer, Berlin, pp. 21–30.

Eberhart, R. C., Kennedy, J. and Shi, Y., eds, 2003, *Proc. Swarm Intelligence Symposium* (Indianapolis, IN).

Gambardella, L. M., Taillard, E. and Dorigo, M., 1999, Ant colonies for the quadratic assignment problem, *J. Oper. Res. Soc.* **50**:167–76.

Goldberg, D. and Mataric, M. J., 2000, Robust behavior-based control for distributed multi-robot collection tasks, *USC Institute for Robotics and Intelligent Systems Technical Report* IRIS-00-387.

Goss, S., Aron, S., Deneubourg, J. L. and Pasteels, J. M., 1989, Self-organized shortcuts in the Argentine ant, *Naturwissenschaften* **76**:579–581.

Guntsch M. and Middendorf, M., 2002a, Applying population based ACO to dynamic optimization problems, in: *Proc. 3rd Int. Workshop (ANTS 2002)*, Lecture Notes in Computer Science, Vol. 2463, Springer, Berlin, pp. 111–122.

Guntsch M., and Middendorf, M., 2002b, A population based approach for ACO, in: *Applications of Evolutionary Computing—Proc. EvoWorkshops 2002*, Lecture Notes in Computer Science, Vol. 2279, Springer, Berlin, pp. 72–81.

Gutjahr, W., 2000, A graph-based Ant System and its convergence, *Future Generation Comput. Syst.* **16**:873–888.

Gutjahr, W., 2002, ACO algorithms with guaranteed convergence to the optimal solution. *Inform. Process. Lett.* **82**:145–153.

Handl, J. and Meyer, B., 2002, Improved ant-based clustering and sorting in a document retrieval interface, in: *Proc. 7th Int. Conference Parallel Problem Solving from Nature—PPSN VII,* J. J. Merelo Guervos et al., eds, Lecture Notes in Computer Science, Vol. 2439, Springer, Berlin, pp. 913–923.

Hu, X., Eberhart, R. and Shi, Y., 2003, Swarm intelligence for permutation optimization: a case study on n-queens problem, in: *Proc. IEEE Swarm Intelligence Symposium 2003* (Indianapolis, IN).

Janson, S. and Middendorf, M., 2003, A hierarchical particle swarm optimizer, in: *Proc. Congress on Evolutionary Computation (CEC 2003),* IEEE Press, Piscataway, NJ, pp. 770–776.

Kawamura, H., Yamamoto, M., Suzuki, K., and Ohucke, A., 2000, Multiple ant colonies algorithm based on colony level interactions, *IEICE Trans. Fundamentals* A **83**:371-379.

Kennedy, J. 2000. Stereotyping: improving particle swarm performance with cluster analysis, in: *Proc. IEEE Int. Conference on Evolutionary Computation,* pp. 1507–1512.

Kennedy J. and Eberhart, R. C., 1995, Particle swarm optimization, in: *Proc. IEEE Int. Conference on Neural Networks,* pp. 1942–1948.

Kennedy, J. and Eberhart, R. C., 1997, A discrete binary version of the particle swarm algorithm. *Proc. 1997 Conf. on Systems, Man, and Cybernetics*, Piscataway, NJ: IEEE Service Center, 4104-4109.

Kennedy, J. and Eberhart, R. C., 1999, The particle swarm: social adaption in information processing systems, in: D. Corne et al., eds, *New Ideas in Optimization,* McGraw-Hill, New York, pp. 379–387.

Kennedy, J., Eberhart, R. C., and Shi, Y., 2001, *Swarm Intelligence,* Morgan Kaufmann, San Francisco, CA.

Kennedy, J. and Mendes, R., 2003, Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms, in: *Proc. IEEE Int. Workshop on Soft Computing in Industrial Applications.*

Ko, P.-C. and Lin, P.-C., 2004, A hybrid swarm intelligence based mechanism for earning forecast, in: *Proc. 2nd Int. Conference on Information Technology and Applications (ICITA 2004).*

Krink, T., Vesterstrøm, J. S. and Riget, J., 2002, Particle swarm optimisation with spatial particle extension, in: *Proc. 4th Congress on Evolutionary Computation (CEC-2002)*, pp. 1474–1479.

Kube, C. R. and Bonabeau, E., 2000, Cooperative transport by ants and robots, *Robot. Autonomous Syst.* **30**:85–101.

Lambrinos, D., Möller, R., Labhart, T., Pfeifer, R., and Wehner, R., 2000, A mobile robot employing insect strategies for navigation, *Robot. Autonomous Syst.* **30**:39–64.

Lumer, E. D. and Faieta, B., 1994, Diversity and adaptation in populations of clustering ants, in: *Proc. 3rd Int. Conference on Simulation of Adaptive Behavior: From Animals to Animats 3 (SAB 94)*, MIT Press, Cambridge, MA, pp.501–508.

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

Maniezzo, V., and Carbonaro, A., 2001, Ant colony optimization: an overview, in: *Essays and Surveys in Metaheuristics,* C. Ribeiro, ed., Kluwer, Dordrecht, pp. 21–44.

Merkle, D. and Middendorf, M., 2001a, A new approach to solve permutation scheduling problems with ant colony optimization, in: *Applications of Evolutionary Computing: Proc. EvoWorkshops 2001,* Lecture Notes in Computer Science, Vol. 2037, E. J. W. Boers et al., eds, Springer, Berlin, pp. 484–493.

Merkle, D. and Middendorf, M., 2001b, On solving permutation scheduling problems with ant colony optimization, *Int. J. Syst. Sci.* submitted.

Merkle, D. and Middendorf, M., 2002a, Modelling the dynamics of ant colony optimization algorithms, *Evol. Comput.* **10**:235–262.

Merkle D. and Middendorf, M., 2002b, Ant colony optimization with the relative pheromone evaluation method, in: *Applications of Evolutionary Computing: Proc. EvoWorkshops 2001*, Lecture Notes in Computer Science, Vol. 2279, Springer, Berlin, pp. 325–333.

Merkle, D., and Middendorf, M., 2003a, On the behavior of ACO algorithms: studies on simple problems, in: *Metaheuristics: Computer Decision-Making,* M. G. C. Resende and J. Pinho de Sousa, eds, Kluwer, Dordrecht, pp. 465–480.

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

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

Michels, R. and Middendorf, M., 1999, An ant system for the shortest common supersequence problem, in: *New Ideas in Optimization,* D. Corne, M. Dorigo and F. Glover, eds, McGraw-Hill, New York, pp. 51–61.

Middendorf, M., 2002. Ant colony optimization, in: *Tutorial Proc. Genetic and Evolutionary Computation Conference (GECCO-2002).*

Montgomery, J., and Randall, M., 2002, Anti-pheromone as a tool for better exploration of search space, in: *Proc. 3rd Int. Workshop ANTS 2002*, Lecture Notes in Computer Science, Vol. 2463, Springer, Berlin, pp. 100–110.

Parsopoulos, K. E. and Vrahatis, M. N., 2001, Modification of the particle swarm optimizer for locating all the global minima, in: *Artificial Neural Networks and Genetic Algorithms,* V. Kurkova et al., eds, Springer, Berlin, pp. 324–327.

Parsopoulos, K. E. and Vrahatis, M. N., 2002, Recent approaches to global optimization problems through particle swarm optimization, *Nat. Comput.* **1**:235–306.

Parsopoulos, K. E., Tasoulis, D. K. and Vrahatis, M. N., 2004, Multiobjective optimization using parallel vector evaluated particle swarm optimization, in: *Proc. IASTED Int. Conference on Artificial Intelligence and Applications.*

Randall, M. and Tonkes, E., 2002, Intensification and diversification strategies in ant colony optimisation, *Complex. Int.* **9**.

Riget, J. and Vesterstrøm, J. S., 2002, A diversity-guided particle swarm optimizer—the ARPSO, *Technical Report* 2002-02, Department of Computer Science, University of Aarhus.

Schoonderwoerd, R., Holland, O., Bruten, J. and Rothkrantz, L., 1996, Ant-based load balancing in telecommunications networks, *Adapt. Behav.* **5**:169–207.

Stützle, T., den Besten, M. and Dorigo, M., 2000, Ant colony optimization for the total weighted tardiness problem, in: *Proc. 6th Int. Conference on Parallel Problem Solving from Nature (PPSN-VI)*, Lecture Notes in Computer Science, Vol. 1917, Deb et al., eds, Springer, Berlin, pp. 611–620.

Stützle, T. and Dorigo, M., 1999, ACO algorithms for the traveling salesman problem, in: *Evolutionary Algorithms in Engineering and Computer Science,* K. Miettinen et al., eds., Wiley, New York, pp. 163–183.

Stützle, T. and Dorigo, M., 2001, An experimental study of the simple ant colony optimization algorithm, in: *Proc. 2001 WSES Int. Conf. on Evolutionary Computation (EC'01).*

Stützle, T. and Dorigo, M., 2002a, A short convergence proof for a class of ACO algorithms. *IEEE Trans. Evol. Comput.* **6**:358–365.

Stützle, T. and Dorigo, M., 2002b, The ant colony optimization metaheuristic: algorithms, applications, and advances, in: *Handbook of Metaheuristics,* F. Glover and G. Kochenberger, eds, Kluwer, Dordrecht.

Stützle, T. and Hoos, H., 1997, Improvements on the ant system: Introducing MAX(MIN) ant system, in: *Proc. Int. Conference on Artificial Neutral Networks and Genetic Algorithms,* Springer, Berlin, pp. 245–249.

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

Suganthan, P. N., 1999, Particle swarm optimizer with neighborhood optimizer, in: *Proc. of the Congress on Evolutionary Computation (CEC 1999)*, pp. 1958–1961.

Sousa, T., Neves, A., and Silva, A., 2003, Swarm optimization as a new tool for data mining, in: *Proc. Workshop on Nature Inspired Distributed Computing (NIDISC)—at IPDPS'03,* p. 144.

Veeramachaneni, K., and Osadciw, L., 2003, Adaptive multimodal biometric fusion algorithm using particle swarm, in: *Proc. SPIE Aerosense 2003* (Orlando, FL), pp. 211–222.

Vesterstrøm, J. S., Riget, J. and Krink, T., 2002, Division of labor in particle swarm optimisation, in: *Proc. 4th IEEE Congress on Evolutionary Computation (CEC-2002),* pp. 1570–1575.

Xiao, X., Dow, E. R., Eberhart, R., Ben Miled, Z. and Oppelt, R. J., 2003, Gene clustering using self-organizing maps and particle swarm optimization, in: *Online Proc. 2nd IEEE Int. Workshop on High Performance Computational Biology (HICOMB 2003).*

Xie, X.-F., Zhang, W.-J. and Yang, Z.-L., 2002, A dissipative particle swarm optimization, in: *Proc. 4th IEEE Congress on Evolutionary Computation (CEC-2002)* (Honolulu, HI).

Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., and Nakanishi, Y., 2000, A particle swarm optimization for reactive power and voltage control considering voltage security assessment, *IEEE Trans. Power Syst.* **15**:1232–1239.

OR-Library, 2004, http://mscmga.ms.ic.ac.uk/jeb/orlib/wtinfo.html