5. Ant Colony Optimization

Vittorio Maniezzo, Luca Maria Gambardella, Fabio de Luigi

5.1 Introduction

Ant Colony Optimization (ACO) is a paradigm for designing metaheuristic algorithms for combinatorial optimization problems. The first algorithm which can be classified within this framework was presented in 1991 [21, 13] and, since then, many diverse variants of the basic principle have been reported in the literature. The essential trait of ACO algorithms is the combination of a priori information about the structure of a promising solution with a posteriori information about the structure of previously obtained good solutions.

Metaheuristic algorithms are algorithms which, in order to escape from local optima, drive some basic heuristic: either a constructive heuristic starting from a null solution and adding elements to build a good complete one, or a local search heuristic starting from a complete solution and iteratively modifying some of its elements in order to achieve a better one. The metaheuristic part permits the low-level heuristic to obtain solutions better than those it could have achieved alone, even if iterated. Usually, the controlling mechanism is achieved either by constraining or by randomizing the set of local neighbor solutions to consider in local search (as is the case of simulated annealing [46] or tabu search [33]), or by combining elements taken by different solutions (as is the case of evolution strategies [11] and genetic [43] or bionomic [56] algorithms).

The characteristic of ACO algorithms is their explicit use of elements of previous solutions. In fact, they drive a constructive low-level solution, as GRASP [30] does, but including it in a population framework and randomizing the construction in a Monte Carlo way. A Monte Carlo combination of different solution elements is suggested also by Genetic Algorithms [40], but in the case of ACO the probability distribution is explicitly defined by previously obtained solution components.

The particular way of defining components and associated probabilities is problem-specific, and can be designed in different ways, facing a trade-off between the specificity of the information used for the conditioning and the number of solutions which need to be constructed before effectively biasing the probability distribution to favor the emergence of good solutions. Different applications have favored either the use of conditioning at the level of decision variables, thus requiring a huge number of iterations before getting a precise distribution, or the computational efficiency, thus using very coarse conditioning information.

The chapter is structured as follows. Section 2 describes the common elements of the heuristics following the ACO paradigm and outlines some of the variants proposed. Section 3 presents the application of ACO algorithms to a number of different combinatorial optimization problems and it ends with a wider overview of the problem attacked by means of ACO up to now. Section 4 outlines the most significant theoretical results so far published about convergence properties of ACO variants.

5.2 Ant Colony Optimization

ACO [1, 24] is a class of algorithms, whose first member, called Ant System, was initially proposed by Colorni, Dorigo and Maniezzo [13, 21, 18]. The main underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result. The collective behavior emerging from the interaction of the different search threads has proved effective in solving combinatorial optimization (CO) problems.

Following [50], we use the following notation. A combinatorial optimization problem is a problem defined over a set $\mathbf{C} = c_1, \ldots , c_n$ of basic *components*. A subset **S** of components represents a *solution* of the problem; $\mathbf{F} \subseteq 2^{\mathbf{C}}$ is the subset of *feasible solutions*, thus a solution **S** is feasible if and only if $\mathbf{S} \in \mathbf{F}$. A *cost function z* is defined over the solution domain, $z : 2^{\mathbf{C}} \rightarrow \mathbf{R}$, the objective being to find a minimum cost feasible solution S*, i.e., to find S*: $S^* \in \mathbf{F}$ and $z(S^*) \leq z(\mathbf{S})$, $\forall \mathbf{S} \in \mathbf{F}$.

Given this, the functioning of an ACO algorithm can be summarized as follows (see also [27]). A set of computational concurrent and asynchronous agents (a colony of ants) moves through states of the problem corresponding to partial solutions of the problem to solve. They move by applying a stochastic local decision policy based on two parameters, called *trails* and *attractiveness*. By moving, each ant incrementally constructs a solution to the problem. When an ant completes a solution, or during the construction phase, the ant evaluates the solution and modifies the trail value on the components used in its solution. This pheromone information will direct the search of the future ants.

Furthermore, an ACO algorithm includes two more mechanisms*: trail evaporation* and, optionally, *daemon actions*. Trail evaporation decreases all trail values over time, in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants, such as the invocation of a local optimization procedure, or the update of global information to be used to decide whether to bias the search process from a non-local perspective [27].

More specifically, an *ant* is a simple computational agent, which iteratively constructs a solution for the instance to solve. Partial problem solutions are seen as *states*. At the core of the ACO algorithm lies a loop, where at each iteration, each ant *moves* (performs a *step*) from a state ι to another one ψ, corresponding to a more complete partial solution. That is, at each step σ, each ant k computes a set $\mathbf{A}_k{}^{s}(\iota)$ of feasible expansions to its current state, and moves to one of these in probability. The probability distribution is specified as follows. For ant k, the probability $\mathrm{p}_{iy}{}^{k}$ of moving from state ι to state ψ depends on the combination of two values:

- the *attractiveness* η_{iy} of the move, as computed by some heuristic indicating the *a priori* desirability of that move;
- the *trail level* τ_{iy} of the move, indicating how proficient it has been in the past to make that particular move: it represents therefore an *a posteriori* indication of the desirability of that move.

Trails are *updated* usually when all ants have completed their solution, increasing or decreasing the level of trails corresponding to moves that were part of "good" or "bad" solutions, respectively.

The general framework just presented has been specified in different ways by the authors working on the ACO approach. The remainder of Section 2 will outline some of these contributions.

5.2.1 Ant System

The importance of the original Ant System (AS) [13, 21, 18] resides mainly in being the prototype of a number of ant algorithms which collectively implement the ACO paradigm. AS already follows the outline presented in the previous subsection, specifying its elements as follows.

The move probability distribution defines probabilities $\mathrm{p}\iota\psi\mathrm{k}$ to be equal to 0 for all moves which are infeasible (i.e., they are in the tabu list of ant k, that is a list containing all moves which are infeasible for ants k starting from state ι), otherwise they are computed by means of formula (5.1), where α and β are user-defined parameters ($0 \le \alpha,\beta \le 1$):

$$\mathrm{p}_{iy}^{\mathrm{k}} = \begin{cases} \dfrac{t_{iy}^{a} + h_{iy}^{b}}{\sum\limits_{(iz) \notin tabu_k} \left(t_{iz}^{a} + \cdot h_{iz}^{b} \right)} & \text{if } (iy) \notin \mathrm{tabu}_{\mathrm{k}} \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

In formula 5.1, tabu$_k$ is the tabu list of ant k, while parameters α and β specify the impact of trail and attractiveness, respectively.

After each iteration t of the algorithm, i.e., when all ants have completed a solution, trails are updated by means of formula (5.2):

$$\tau_{iy}\ (\tau) = \rho\ \tau_{iy}\ (\tau - 1) + \Delta\tau_{iy} \qquad (5.2)$$

where $\Delta\tau_{iy}$ represents the sum of the contributions of all ants that used move ($\iota\psi$) to construct their solution, ρ, $0 \le \rho \le 1$, is a user-defined parameter called *evaporation coefficient,* and $\Delta\tau_{iy}$ represents the sum of the contributions of all ants that used move ($\iota\psi$) to construct their solution. The ants' contributions are proportional to the quality of the solutions achieved, i.e., the better solution is, the higher will be the trail contributions added to the moves it used.

For example, in the case of the TSP, moves correspond to arcs of the graph, thus state ι could correspond to a path ending in node i, the state ψ to the same path but with the arc (*ij*) added at the end and the move would be the traversal of arc (*ij*). The quality of the solution of ant k would be the length L_k of the tour found by the ant and formula (5.2) would become $\tau_{ij}(t)=\rho\ \tau_{ij}(t\text{-}1)+\Delta\tau_{ij}$, with

$$\Delta t_{ij} = \sum_{k=1}^{m} \Delta t_{ij}^{k} \qquad (5.3)$$

where m is the number of ants and Δt_{ij}^{k} is the amount of trail laid on edge (*ij*) by ant k, which can be computed as

$$\Delta t_{\mathrm{ij}}^{\mathrm{k}} = \begin{cases} \dfrac{\mathrm{Q}}{\mathrm{L_k}} \text{if ant } k \text{ uses arc (ij) in its tour} \\ \quad 0 \quad \text{otherwise} \end{cases} \qquad (5.4)$$

Q being a constant parameter.

The ant system simply iterates a main loop where m ants construct in parallel their solutions, thereafter updating the trail levels. The performance of the algorithm depends on the correct tuning of several parameters, namely: α, β, relative importance of trail and attractiveness, ρ, trail persistence, $\tau_{ij}(0)$, initial trail level, *m,* number of ants, and Q, used for defining to be of high quality solutions with low cost. The algorithm is the following.

```
1. {Initialization}
   Initialize $\tau_{iy}$ and $\eta_{iy}$, " (iy ).

2. {Construction}
   For each ant k (currently in state i) do
       repeat
           choose in probability the state to move into.
           append the chosen move to the k-th ant's set tabu$_k$.
```

```
        until ant k has completed its solution.
    end for

3.  {Trail update}
    For each ant move ($iy$) do
        compute $\Delta\tau_{iy}$
        update the trail matrix.
    end for

4.  {Terminating condition}
    If not(end test) go to step 2
```

5.2.2 Ant Colony System

AS was the first algorithm inspired by real ants behavior. AS was initially applied to the solution of the traveling salesman problem but was not able to compete against the state-of-the art algorithms in the field. On the other hand he has the merit to introduce ACO algorithms and to show the potentiality of using artificial pheromone and artificial ants to drive the search of always better solutions for complex optimization problems. The next researches were motivated by two goals: the first was to improve the performance of the algorithm and the second was to investigate and better explain its behavior. Gambardella and Dorigo proposed in 1995 the Ant-Q algorithm [35], an extension of AS which integrates some ideas from Q-learning, and in 1996 Ant Colony System (ACS) [36, 25] a simplified version of Ant-Q which maintained approximately the same level of performance, measured by algorithm complexity and by computational results. Since ACS is the base of many algorithms defined in the following years we focus the attention on ACS other than Ant-Q. ACS differs from the previous AS because of three main aspects:

Pheromone

In ACS once all ants have computed their tour (i.e. at the end of each iteration) AS updates the pheromone trail using all the solutions produced by the ant colony. Each edge belonging to one of the computed solutions is modified by an amount of pheromone proportional to its solution value. At the end of this phase the pheromone of the entire system evaporates and the process of construction and update is iterated. On the contrary, in ACS only the best solution computed since the beginning of the computation is used to *globally update* the pheromone. As was the case in AS, global updating is intended to increase the attractiveness of promising route but ACS mechanism is more effective since it avoids long convergence time by directly concentrate the search in a neighborhood of the best tour found up to the current iteration of the algorithm.

In ACS, the final evaporation phase is substituted by a *local updating* of the pheromone applied during the construction phase. Each time an ant moves from the current city to the next the pheromone associated to the edge is modified in the following way: $t_{ij}(t) = r \cdot t_{ij}(t-1) + (1-r) \cdot t_0$ where $0 \le \rho \le 1$ is a parameter (usually set at 0.9) and t_0 is the initial pheromone value. t_0 is defined as $t_0=(n \cdot L_{nn})^{-1}$, where L_{nn} is the tour length produced by the execution of one ACS iteration without the pheromone component (this is equivalent to a probabilistic nearest neighbor heuristic). The effect of local-updating is to make the desirability of edges change dynamically: every time an ant uses an edge this becomes slightly less desirable and only for the edges which never belonged to a global best tour the pheromone remains t_0. An interesting property of these local and global updating mechanisms is that the pheromone $t_{ij}(t)$ of each edge is inferior limited by t_0. A similar approach was proposed with the Max-Min-AS (MMAS, [70]) that explicitly introduces lower and upper bounds to the value of the pheromone trials.

State Transition Rule

During the construction of a new solution the state transition rule is the phase where each ant decides which is the next state to move to. In ACS a new state transition rule called *pseudo-random-proportional* is introduced. The *pseudo-random-proportional* rule is a compromise between the *pseudo-random* state choice rule typically used in Q-learning [WQ92] and the *random-proportional* action choice rule typically used in Ant System. With the pseudo-random rule the chosen state is the best with probability q_0 (*exploitation*) while a random state is chosen with probability $1-q_0$ (*exploration*). Using the AS random-proportional rule the next state is chosen randomly with a probability distribution depending on h_{ij} and t_{ij}. The ACS *pseudo-random-proportional* state transition rule provides a direct way to balance between exploration of new states and exploitation of a priori and accumulated knowledge. The best state is chosen with probability q_0 (that is a parameter $0 \le q_0 \le 1$ usually fixed to 0.9) and with probability $(1-q_0)$ the next state is chosen randomly with a probability distribution based on h_{ij} and t_{ij} weighted by a (usually equal to 1) and b (usually equal to 2) .

$$s = \begin{cases} \arg\max\limits_{(ij)\notin tabu_k} \left\{ t_{ij}^{\,a} \cdot h_{ij}^{\,b} \right\} & \text{if } q \le q_0 \quad \text{(exploitation)} \\ \text{AS rule 2.1} & \text{otherwise} \quad \text{(exploration)} \end{cases} \tag{5.5}$$

Hybridization and performance improvement

ACS was applied to the solution of big symmetric and asymmetric traveling salesman problems (TSP/ATSP) [36],[25]. For these purpose ACS incorporates an advanced data structure known as *candidate list* [60]. A candidate list is a static data structure of length *cl* which contains, for a given city *i*, the *cl* preferred cities to be visited. An ant in ACS first uses candidate list with the state transition rules to choose the city to move to. If none of the cities in the candidate list can be visited the ant chooses the nearest available city only using the heuristic value h_{ij}. ACS for TSP/ATSP has been improved by incorporating local optimization heuristic (*hybridization*): the idea is that each time a solution is generated by the ant it is taken to its local minimum by the application of a local optimization heuristic based on an edge exchange strategy, like 2-opt, 3-opt or Lin-Kernighan [48]. The new optimized solutions are considered as the final solutions produced in the current iteration by ants and are used to globally update the pheromone trails.

This ACS implementation combining a new pheromone management policy, a new state transition strategy and local search procedures was finally competitive with state-of-the-art algorithm for the solution of TSP/ATSP problems [5]. This opened a new frontier for ACO based algorithm. Following the same approach that combines a constructive phase driven by the pheromone and a local search phase that optimizes the computed solution, ACO algorithms were able to break several optimization records, including those for routing and scheduling problems that will be presented in the following paragraphs.

5.2.3 ANTS

ANTS is an extension of the AS proposed in [50], which specifies some underdefined elements of the general algorithm, such as the attractiveness function to use or the initialization of the trail distribution. This turns out to be a variation of the general ACO framework that makes the resulting algorithm similar in structure to tree search algorithms. In fact, the essential trait which distinguishes ANTS from a tree search algorithm is the lack of a complete backtracking mechanism, which is substituted by a probabilistic (*Non-deterministic*) choice of the state to move into and by an incomplete (*Approximate*) exploration of the search tree: this is the rationale behind the name ANTS, which is an acronym of *Approximated Non-deterministic Tree Search*. In the following, we will outline two distinctive elements of the ANTS algorithm within the ACO framework, namely the attractiveness function and the trail updating mechanism.

Attractiveness

The attractiveness of a move can be effectively estimated by means of lower bounds (upper bounds in the case of maximization problems) on the cost of the completion of a partial solution. In fact, if a state ι corresponds to a partial problem solution it is possible to compute a lower bound on the cost of a complete solution containing ι. Therefore, for each feasible move ι,ψ, it is possible to compute the lower bound on the cost of a complete solution containing ψ: the lower the bound the better the move. Since a large part of research in ACO is devoted to the identification of tight lower bounds for the different problems of interest, good lower bounds are usually available.

When the bound value becomes greater than the current upper bound, it is obvious that the considered move leads to a partial solution which cannot be completed into a solution better than the current best one. The move can therefore be discarded from further analysis. A further advantage of lower bounds is that in many cases the values of the decision variables, as appearing in the bound solution, can be used as an indication of whether each variable will appear in good solutions. This provides an effective way of initializing the trail values. For more details see [50].

Trail update

A good trail updating mechanism avoids stagnation, the undesirable situation in which all ants repeatedly construct the same solutions making any further exploration in the search process impossible. Stagnation derives from an excessive trail level on the moves of one solution, and can be observed in advanced phases of the search process, if parameters are not well tuned to the problem.

The trail updating procedure evaluates each solution against the last *k* solutions globally constructed by ANTS. As soon as *k* solutions are available, their moving average $\bar{z}$ is computed; each new solution z_{curr} is compared with $\bar{z}$ (and then used to compute the new moving average value). If z_{curr} is lower than $\bar{z}$, the trail level of the last solution's moves is increased, otherwise it is decreased. Formula (5.6) specifies how this is implemented:

$$\Delta t_{ij} = t_0 \cdot \left(1 - \frac{z_{curr} - LB}{\bar{z} - LB}\right) \tag{5.6}$$

where $\bar{z}$ is the average of the last *k* solutions and LB is a lower bound on the optimal problem solution cost. The use of a dynamic scaling procedure permits discrimination of a small achievement in the latest stage of search, while avoiding focusing the search only around good achievement in the earliest stages.

One of the most difficult aspects to be considered in metaheuristic algorithms is the trade-off between exploration and exploitation. To obtain good results, an agent should prefer actions that it has tried in the past and found to be effective in producing desirable solutions (exploitation); but to discover them, it has to try actions not previously selected (exploration). Neither exploration nor exploitation can be pursued exclusively without failing in the task: for this reason, the ANTS algorithm integrates the stagnation avoidance procedure to facilitate exploration with the probability definition mechanism based on attractiveness and trails to determine the desirability of moves.

Based on the elements described, the ANTS algorithm is as follows.

```
1.    Compute a (linear) lower bound LB to the problem
```

```
     Initialize $\tau_{iy}$ ($\forall\iota,\psi$) with the primal variable values
2.   For k=1,m (m= number of ants) do
       repeat
2.1      compute $\eta_{iy}$ $\forall(\iota\psi)$
2.2      choose in probability the state to move into
2.3      append the chosen move to the $k$-th ant’s tabu list
       until ant $k$ has completed its solution
2.4    carry the solution to its local optimum
     end for
3.   For each ant move ($\iota\psi$),
     compute $\Delta\tau_{iy}$ and update trails by means of (5.6)
4.   If not(end_test) goto step 2.
```

It can be noted that the general structure of the ANTS algorithm is closely akin to that of a standard tree search procedure. At each stage we have in fact a partial solution which is expanded by branching on all possible offspring; a bound is then computed for each offspring, possibly fathoming dominated ones, and the current partial solution is selected from among those associated to the surviving offspring on the basis of lower bound considerations. By simply adding backtracking and eliminating the MonteCarlo choice of the node to move to, we revert to a standard branch and bound procedure. An ANTS code can therefore be easily turned into an exact procedure.

5.3 Significant problems

In the following of this section we will present applications of ACO algorithms to some significant combinatorial optimization problems. This is to give the reader an idea of what is involved by the use of an ACO algorithm for a problem: even though the last subsection presents an overview of recent application the list is by no means exhaustive, as it becomes readily evident by searching the web under the keywords “ant colony optimization”.

5.3.1 Sequential ordering problem

The first ACO applications were devoted to solve the symmetric and the asymmetric traveling salesman problem. Given a set of cities $V = \{v_1, \dots , v_n\}$, a set of edges $A = \{(i,j) : i,j \in V\}$ and a cost $d_{ij} = d_{ji}$ associated with edge $(i,j) \in A$, the TSP is the problem of finding a minimal length closed tour that visits each city once. In case d_{ij} ? d_{ji} for at least one edge (i,j) than the TSP becomes an Asymmetric TSP (ATSP). The first algorithm that applies an ACO based algorithm to a more general version of the ATSP problem is *Hybrid Ant System for the Sequential Ordering Problem* (HAS-SOP, [34]). HAS-SOP was intended to solve the sequential ordering problem with precedence constraints (SOP). The SOP in an *NP*-hard combinatorial optimization problem first formulated by Escudero [29] to design heuristics for a production planning system. The SOP models real-world problems like production planning [29], single-vehicle routing problems with pick-up and delivery constraints [64], and transportation problems in flexible manufacturing systems [2]. The SOP can be seen as a general case of both the ATSP and the pick-up and delivery problem [47]. It differs from ATSP because the first and the last nodes are fixed, and in the additional set of precedence constraints on the order in which nodes must be visited. It differs from the pick-up and delivery problem because this is usually based on symmetric TSPs, and because the pick-up and delivery problem includes a set of constraints between nodes with a unique predecessor defined for each node, in contrast to the SOP where multiple precedences can be defined.

HAS-SOP combines a constructive phase (ACS-SOP) based on the ACS algorithm [36] with a new local search procedure called SOP-3-exchange. SOP-3-exchange is based on a lexicographic search heuristic due to [64], on a new labeling procedure and on a new data structure called *don't push stack* inspired by the *don't look bit* [5] both introduced by the authors. SOP-3-exchange is the first local search able to handle multiple precedence constraints in constant time.

ACS-SOP implements the constructive phase of HAS-SOP but differs from ACS in the way the set of feasible nodes is computed and in the setting of one of the algorithm's parameters that is made dependent on the problem dimensions. ACS-SOP generates feasible solutions that does not violate the precedence constraints with a computational cost of order $O(n^2)$ like the traditional ACS heuristic.

A set of experiments based on the TSPLIB data shows that HAS-SOP algorithm is more effective than other existing methods for the SOP. HAS-SOP was compared against the two previous most effective algorithms: a branch-and-cut algorithm [2] that proposed a new class of valid inequalities and Maximum Partial Order/Arbitrary Insertion (MPO/AI), a genetic algorithm by Chen and Smith [17].

To better understand the role of the constructive ACS-SOP phase and the role of the SOP-3-exchange local search MPO/AI was also coupled with the SOP-3-exchange local search. Experimental results shows that MPO/AI alone is better than ACS-SOP due to the use of a simple local search embedded in its crossover operator. On the contrary, the combination between constructive phase and local search shows that HAS-SOP is better than both MPO/AI alone and MPO/AI + SOP-3-exchange. This is probably due to the fact that MPO/AI generates solutions that are already optimized and therefore the SOP-3-exchange procedure quickly gets stuck. On the contrary, ACS-SOP solution is a very effective starting point for the SOP-3-exchange local search therefore the HAS-SOP hybridization is very effective. Currently HAS-SOP is the best known method to solve the SOP and was able to improve 14 over 22 best known results in the TSPLIB data set.

5.3.2 Vehicle routing problems

A direct extension of the TSP, the first problem AS was applied to, are Vehicle routing problems (VRPs). These are problems where a set of vehicles stationed at a depot has to serve a set of customers before returning to the depot, and the objective is to minimize the number of vehicles used and the total distance traveled by the vehicles. Capacity constraints are imposed on vehicle trips, as well as possibly a number of other constraints deriving from real-world applications, such as time windows, backhauling, rear loading, vehicle objections, maximum tour length, etc. The basic VRP problem is the Capacitated VRP (CVRP): AS_{rank}, the rank-based version of AS, was applied to this problem by Bullnheimer, Hartl and Strauss [7, 8] with good results. These authors used various standard heuristics to improve the quality of VRP solutions and modified the construction of the tabu list considering constraints on the maximum total tour length of a vehicle and on its capacity.

Following these results, and the excellent ones obtained by ACS with TSP, SOP and QAP problems, ACS was applied to a VRP version more close to actual logistic practice, called VRPTW. VRPTW is defined as the problem of minimizing time and costs in case a fleet of vehicles has to distribute goods from a depot to a set of customers. The VRPTW minimizes a multiple, hierarchical objective function: the first objective is to minimize the number of tours (or vehicles) and the second objective is to minimize the total travel time. A solution with a lower number of tours is always preferred to a solution with a higher number of tours even if the travel time is higher. This hierarchical objectives VRPTW is very common in the literature and in case problem constraints are very tight (for example when the total capacity of the minimum number of vehicles is very close to the total volume to deliver or when customers time windows are narrow), both objectives can be antagonistic: the minimum travel time solution can include a number of vehicles higher than the solution with minimum number of vehicles (see e.g. Kohl et al., [45]).

To adapt ACS to a multiple objectives the Multiple Ant Colony System for the VRPTW (MACS-VRPTW [38]) has been defined. MACS-VRPTW is organized with a hierarchy of artificial ACS colonies designed to hierarchically optimize a multiple objective function: the first ACS colony (ACS-VEI) minimizes the number of vehicles while the second ACS colony (ACS-TIME) minimizes the traveled distances. Both colonies use independent pheromone trails but they collaborate by exchanging information through mutual pheromone updating. In the MACS-VRPTW algorithm both objective functions are optimized simultaneously: ACS-VEI tries to diminish the number of vehicles searching for a feasible solution with always one vehicle less than the previous feasible solution.

```
0.  MACS-VRPTW algorithm
1.  {Initialization}
    Initialize $\psi^{gb}$ the best feasible solution: lowest number
    of vehicles and shortest travel time.
2.  {Main loop}
    Repeat
2.1 Vehicles ← #active_vehicles($\psi^{gb}$) /* The actual number of used vehicles is computed */
2.3 Activate ACS-VEI(Vehicles - 1)  /* ACS-VEI searches for a feasible solution with always one vehicle less by maximising the num. of visited customers */
2.4 Activate ACS-TIME(Vehicles)     /* ACS-TIME is a traditional ACS colony that minimises the travel time */
    While ACS-VEI and ACS-TIME are active
       Wait for an improved solution $y$ from ACS-VEI or ACS-TIME
2.5    $\psi^{gb}$ ← $y$
       if #active_vehicles($\psi^{gb}$) < Vehicles then
2.6       kill ACS-TIME and ACS-VEI
    End While
    until a stopping criterion is met
```

ACS-VEI is therefore different from the traditional ACS applied to the TSP. In ACS-VEI the current best solution is the solution (usually unfeasible) with the highest number of visited customers, while in ACS the current best solution is the shortest one. On the contrary, ACS-TIME is a more traditional ACS colony: ACS-TIME, optimizes the travel time of the feasible solutions found by ACS-VEI. As in HAS-SOP, ACS-TIME is coupled with a local search procedure that improves the quality of the computed solutions. The local search uses data structure similar to the data structure implemented in HAS-SOP [36] and is based on the exchange of two sub-chains of customers. One of this sub-chain may eventually be empty, implementing a more traditional customer insertion.

Experimentally has been shown that the performance of the system increases in case the best solution ψ^{gb} calculated in ACS-TIME is used, in combination with the ACS-VEI best solution $\psi^{ACS-VEI}$, to update the pheromone in ACS-VEI equation (5.7).

$$t_{ij}(t) = r \cdot t_{ij}(t-1) + (1-r) \big/ L_y^{\mathrm{ACS\text{-}VEI}} \qquad \forall (i,j) \in y^{\mathrm{ACS\text{-}VEI}} \tag{5.7}$$

$$t_{ij}(t) = r \cdot t_{ij}(t-1) + (1-r) \big/ L_y^{gb} \qquad \forall (i,j) \in y^{gb}$$

MACS-VRPTW has been experimentally proved to be most effective than the best known algorithms in the field such as the the tabu search of Rochat and Taillard [61], the large neighbourhood search of Shaw [71] and the genetic algorithm of Potvin and Bengio [58]. MACS-VRPTW was also able to improve many results in the Solomon problem set both decreasing the number of vehicle or the travelled time.

MACS-VRPTW introduces a new methodology for optimising multiple objective functions. The basic idea is to coordinate the activity of different ant colonies, each of them optimizing a different objective. These colonies work by using independent pheromone trails but they collaborate by exchanging information. This is the first time a multi-objective function minimization problem is solved with a multiple ant colony optimization algorithm.

5.3.3 Quadratic Assignment Problem

The quadratic assignment problem (QAP) is the problem of assigning *n* facilities to *n* locations so that the assignment cost is minimized, where the cost is defined by a quadratic function. The QAP is considered one of the hardest CO problems, and can be solved to optimality only for small instances. Several ACO applications dealt with the QAP, starting using AS [MC94] and then by means of several of the more advanced versions [54], [66]. The limited effectiveness of AS was in fact improved using a well-tuned local optimizer [53], but several other systems previously introduced were also adapted to the QAP. For example, two efficient techniques are the MMAS-QAP algorithm [71] and HAS-QAP [39]. For the testing of QAP solution algorithms, Taillard [75] proposed to categorize instances into four groups: (*i*) unstructured, uniform random (*ii*) unstructured, grid distance, (*iii*) real-world and (*iv*) real-world-like. Both MMAS-QAP and HAS-QAP have been applied to problem instances of type *i* and *iii*. The performances of these two heuristic approaches are strongly dependent on the type of problem. Comparisons with some of the best heuristics for the QAP have shown that HAS-QAP performs well as far as real-world, irregular and structured problems are concerned. On the other hand, on random, regular and unstructured problems the performance of this technique is less competitive.

This problem-dependency was not shown by ANTS, which was also applied to QAP. In order to apply ANTS to QAP (or any other problem), it is necessary to specify the lower bound to use and what is a move in the problem context (step 2.2). The application described in [50] made the following choices.

As for the lower bound, since there is currently no lower bound for QAP, which is both tight and efficient to compute, the LBD bound was used, which can be computed in O(n) but which is unfortunately on the average quite far from the optimal solution.

As for the moves, it was declared that a move corresponds to the assignment of a facility to a location, thus adding a new component to the partial solution corresponding to the state from which the move originated. Some considerations on the move structure were used to improve the computational effectiveness of the resulting algorithm.

ANTS was tested on instances up to n=40 and showed to be effective on all instance types; moreover its direct transposition into an exact branch and bound was also effective when compared to other exact algorithms.

5.3.4 Other problems

This section outlines some of the more recent applications of ACO approaches to problems other than those listed in the previous ones. This variety is well represented in the many diverse conference with tracks entirely dedicated to ACO and most notably in ANTS conference series, entirely dedicated to algorithms inspired by the observation of ants' behavior (ANTS'98, ANTS'2000 and ANTS'2002). Many different applications have been presented: from plan merging to routing problems, from driver scheduling to search space sharing, from set covering to nurse scheduling, from graph coloring to dynamic multiple criteria balancing problems. A large part of the relevant literature can be accessed online from [1].

Moreover, several introductory overviews have been published. We refer the reader to [23], [24] and [52] for other overviews on ACO.

Among the problems not in the list above, a prominent role is played by the TSP. In fact, TSP has been and in many cases still is the first testbed for ACO variants, and more in general for most combinatorial optimization metaheuristics [68]. It was already on this problem that the limited effectiveness of the first variants emerged, and this fostered the design of improved approaches modifying some algorithm element and possibly hybridizing the framework with greedy local search or with other approaches, such as genetic algorithms or tabu seach [69], [42], [73]. These variants were then applied to other problems, for example MAX-MIN ant system was applied to the flow shop problem in [63], a problem then faced also with other ACO modifications [10], whereas in [8] a rank-based approach for the TSP is described or in [14] a so-called best-worst variant.

More recently, different authors ([76], [77], [44]) have tackled the TSP with hybrid variants, mainly using tabu search, but also, in the case of large TSP instances, also with genetic evolution and nearest neighbor search, in order to improve both efficiency and efficacy. Moreover, variations of the basic TSP, such as the orienteering problem [49] or the probabilistic TSP where clients have to be visited with a certain probability [4] have also been studied.

Scheduling problems provide another common area for testing the effectiveness of ACO algorithms. An ACO approach for the job-shop scheduling is presented in [12], whereas applications to real-world scheduling cases have been recently described in [3] and [62].

More recently, the maturity of the field is showed by the fact that ACO approaches began to be proposed also for problems which are not standard combinatorial optimization testbed, but which are more directly connected to actual practice. For example, the problem of searching and clustering records of large databases is faced by means of ACO in [59], while an algorithm for document clustering is described in [80]. Even more theoretical problems linked to spatial data analysis were tackled with ACO techniques in [74] and [37].

Finally, a recent interesting research branch of ACO, not directly related to combinatorial optimization, is about telecommunication. In fact, the area of packet switching communications appear to be a promising field for ACO-related routing approaches [19, 20]. Whereas a standard optimization version of the frequency assignment problem was described in [51], an application to wavelength allocation was presented in [57], while techniques for path adaptive search are described in [79], [22], [9], [81] and an application to a satellite network in [67]. Moreover, applications directly related to communication Quality of Service (QoS) have been presented in [28], and more recently in [15], while an application which optimizes communication systems with GPS techniques is described in [16].

5.4 Convergence proofs

Recently, some works appeared which provide theoretical insight into the convergence properties of ant colony algorithms. All proofs refer to simplified versions of actually used systems, and do not provide direct guidelines for real-world usage, but they are of interest for the ascertainment of general properties of the systems used.

The first such proofs was proposed by Gutjahr [31], who worked on an ACO variant called Graph-Based Ant System (GBAS). The name derives from the analysis being carried on a so-called *construction graph*, which is a graph assigned to an instance of the optimization problem under consideration, encoding feasible solutions by “walks” on the graph. The objective function value of the walk is equal to the objective function value of the corresponding feasible solution of the original problem. It is always possible to design a construction graph for any given combinatorial optimization problem instance, with a number of nodes linear in the number of bits needed for the representation of a solution, and a number of arcs quadratic in this number of bits. Gutjahr proved that, under the conditions listed below, the solutions generated in each iteration of this Graph-based Ant System converge with a probability that can be made arbitrarily close to 1 to the optimal solution of the given problem instance. Essential conditions are: (*i*) there is only one optimal walk in W, i.e., the optimal solution is unique, and it is encoded by only one walk in W; (*ii*) along the optimal walk w*, the desirability values satisfy η_{kl}.(u) > 0 for all arcs (*k*,*l*) of w* and the corresponding partial walks u of w*; (*iii*) a version of what is called *elitist strategy* is used, where only the best walks are rewarded: walks that are dominated by another already traversed walk do not get pheromone increments anymore. Especially the first of these conditions is quite restrictive.

Stützle and Dorigo [65] propose another convergence proof. They consider both the MAX-MIN Ant System and the ACS outlined in Section 3, and they show that in this case it is possible to prove that allowing more and more iterations the cost of the best solution found converges with probability equal to 1 to the optimal cost. This a property already guaranteed by random search alone, and it does not get lost imposing a minimum trail value. Moreover, the authors show that it is possible to compute a lower bound for the probability of the current best solution to be optimal.

Finally, Gutjahr [32] in a recent paper builds upon these results context of ACO and shows that for a particular ACO algorithm, a time-dependent modification of GBAS, the current solutions converge to an optimal solution with probability exactly one. More specifically, he shows that by using suitable parameter schemes, it can be guaranteed that the optimal paths get attractors of the stochastic dynamic process realized by the algorithm. This improves all previous results and proves a property of the same strength of the tightest one so far obtained in the whole metaheuristic area, which was that obtained by Hajek [41] for Simulated Annealing.

5.5 Conclusions

Ant Colony Optimization has been and continues to be a fruitful paradigm for designing effective combinatorial optimization solution algorithms. After more than ten years of studies, both its application effectiveness and its theoretical groundings have been demonstrated, making ACO one of the most successful paradigm in the metaheuristic area.

This overview tries to propose the reader both introductory elements and pointers to recent results, obtained in the different directions pursued by current research on ACO.

No doubt new results will both improve those outlined here and widen the area of applicability of the ACO paradigm.

References

1. M. Dorigo, Ant colony optimization web page, http://iridia.ulb.ac.be/mdorigo/ACO/ACO.html.
2. N. Ascheuer, Hamiltonian path problems in the on-line optimization of flexible manufacturing systems. Ph.D.Thesis, Technische Universität Berlin, Germany, 1995
3. M. E. Bergen, Canstraint-based assembly line sequencing, *Lecture Notes in Computer Science*, 2001
4. L. Bianchi, L.M. Gambardella, M.Dorigo. An ant colony optimization approach to the probabilistic traveling salesman problem. *In Proceedings of PPSN-VII, Seventh Inter-*

national Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science. Springer Verlag, Berlin, Germany, 2002

5. E. Bonabeau, M. Dorigo, G. Theraulaz, *Nature*, Volume 406, Number 6791, Pag. 39 - 42 (2000)
6. J.L. Bentley, Fast algorithms for geometric traveling salesman problem, *ORSA Journal on Computing*, vol. 4, pp. 387–411, 1992.
7. B. Bullnheimer, R.F. Hartl, and C. Strauss, Applying the ant system to the vehicle routing problem, In: Voss S., Martello S., Osman I.H., Roucairol C. (eds.) *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Boston, 1999.
8. B. Bullnheimer, R.F. Hartl, and C. Strauss, A new rank-based version of the ant system: a computational study, *Central European Journal of Operations Research* 7 (1) (1999), 25–38.
9. C. N. Bendtsen, T. Krink, Phone routing using the dynamic memory model, in *Proceedings of the 2002 congress on Evolutionary Computation*, Honolulu, USA
10. C. Blum, M. Sampels, Ant colony optimization for FOP shop scheduling: a case study on different pheromone representations, *in Proceedings of the 2002 congress on Evolutionary Computation*, Honolulu, USA
11. T. Bäck and H.-P. Schwefel, An overview of evolutionary algorithms for parameter optimization, *Evolutionary Computation* 1(1), (1993), 1-23.
12. M. den Besten, T. Stützle, M. Dorigo, Ant colony optimization for the total weighted tardiness problem, *Parallel Problem Solving from Nature: 6th international conference,* September 2000. Springer Verlag.
13. A. Colorni, M. Dorigo, and V. Maniezzo, Distributed optimization by ant colonies, Proceedings of ECAL'91, *European Conference on Artificial Life*, Elsevier Publishing, Amsterdam, 1991.
14. O. Cordon, I. Fernandez de Viana, F. Herrera, L. Moreno, A new ACO model integrating evolutionary computation concepts: the best-worst ant system, in *Proceedings of ANTS2000 –from ant colonies to artificial ants*, Universitè Libre de Bruxelles
15. C. Chao-Hsien, G. JunHua, H. Xiang Dan, G. Qijun, A heuristic ant algorithm for solving QoS multicast routing problem, in *Proceedings of the 2002 congress on Evolutionary Computation*, Honolulu, USA
16. D. Camara, A.A.F. Loureiro, A GPS/ant-like routing algorithm for ad hoc networks, in *2000 IEEE Wireless Communications and Networking Conference*, Chicago, USA
17. S. Chen, S. Smith., Commonality and genetic algorithms. *Technical Report CMU-RI-TR-96-27*, The Robotic Institute, Carnegie Mellon University, Pittsburgh, PA, USA, 1996
18. M. Dorigo, Optimization, learning and natural algorithms*, Ph.D. Thesis*, Politecnico di Milano, Milano, 1992.
19. G. di Caro and M. Dorigo, Antnet: distributed stigmergetic control for communications networks, *Journal of Artificial Intelligence Research*, 9 (1998), 317-365.
20. G. di Caro and M. Dorigo, Mobile agents for adaptive routing, *Proceedings of HICSS-31*, 1998.
21. M. Dorigo, V. Maniezzo, and A. Colorni, The ant system: an autocatalytic optimizing process, *Technical Report TR91-016*, Politecnico di Milano (1991).
22. G. di Caro and M. Dorigo, AntNet: distributed stigmergetic control for communications network , *Journal of Artificial Intelligence Research (JAIR)*, Vol. 9, Pag. 317-365, 1998

23. M. Dorigo and G. Di Caro (1999). The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo and F. Glover, editors, *New Ideas in Optimization*, McGraw-Hill, 11-32.
24. M. Dorigo, G. Di Caro & L.M. Gambardella (1999). Ant Algorithms for Discrete Optimization. *Artificial Life*, 5(2):137-172.
25. M. Dorigo and L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transaction on Evolutionary Computation* 1 (1997), 53--66.
26. M. Dorigo, V. Maniezzo, and A. Colorni, The ant system: optimization by a colony of cooperating agents, *IEEE Transactions on Systems, Man, and Cybernetics-Part B* 26(1) (1996), 29--41.
27. M. Dorigo, T. Stützle. The ant colony optimization metaheuristic: Algorithms, applications and advances. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*. Kluwer Academic Publishers, To appear in 2002.
28. G. Di Caro, Vasilakos T., Ant-SELA: Ant-agents and stochastic automata learn adaptive routing tables for QoS routing in ATM networks, *ANTS'2000 - From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms*, Brussels, Belgium, September 2000.
29. L.F. Escudero, An inexact algorithm for the sequential ordering problem. *European Journal of Operational Research* 37 (1988), 232–253.
30. T.A. Feo and M.G.C. Resende, Greedy randomized adaptive search procedures, *Journal of Global Optimization* 6 (1995), 109--133.
31. W.J. Gutjahr, A graph-based Ant System and its convergence. *Future Generation Computer Systems*. 16, 873 - 888, 2000.
32. W.J. Gutjahr: ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters* 82(3): 145-153 (2002).
33. F. Glover, Tabu search, *ORSA Journal on Computing* 1 (1989), 190--206.
34. L.M. Gambardella and M. Dorigo, An ant colony system hybridized with a new local search for the sequential ordering problem, *INFORMS Journal on Computing* 12 (2000), no. 3, 237--255.
35. L.M. Gambardella and M. Dorigo, Ant-Q: a reinforcement learning approach to the travelling salesman problem, *Proceedings of the Twelfth International Conference on Machine Learning*, ML-95, Palo Alto, CA, Morgan Kaufmann, Palo Alto, California, USA, 1995.
36. L.M Gambardella and M. Dorigo M, Solving Symmetric and Asymmetric TSPs by Ant Colonies , *Proceedings of the IEEE Conference on Evolutionary Computation*, ICEC96, Nagoya, Japan, May 20-22, 1996, pp. 622-627.
37. Y. Gabriely, E. Rimon " Spanning-tree based coverage of continuous areas by a mobile robot", in Proceedings of the IEEE international conference on Robotics and Automation, 2001, Seoul, South Korea
38. L.M. Gambardella, E. Taillard, and G. Agazzi, Ant colonies for vehicle routing problems, vol. *New Ideas in Optimization*, McGraw-Hill, London, 1999.
39. L.M. Gambardella, E. Taillard, and M. Dorigo, Ant colonies for the quadratic assignment problem, *Journal of the Operational Research Society* 50 (1999), 167--176.
40. J.H. Holland, Adaptation in natural and artificial systems, University of Michigan Press, 1975.
41. B. Hajek, Cooling schedules for optimal annealing, *Math. of OR*, 13, pag. 311–329, 1988.

42. H.M. Botee, E. Bonabeau, Evolving ant colony optimization, (SFI Working Paper Abstract, 1999)
43. C. Hurkens and S. Tiourine, Upper and lower bounding techniques for frequency assignment problems, *Technical Report* 95-34, T.U. Eindhoven (1995).
44. T. Kaji, Approach by ant tabu agents for Traveling Salesman Problem, 2001 IEEE *International Conference on System, Man and Cybernetics*
45. N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, F. Soumis, K-Path Cuts for the Vehicle Routing Problem with Time Windows, *Technical Report IMM-REP-1997-*12, Technical University of Denmark, 1997.
46. S. Kirkpatrik, C.D. Gelatt, and M.P. Vecchi, Optimization by simulated annealing, *Science* 220 (1983), 671--680.
47. G.A.P. Kindervater, Savelsbergh. M.W.P. Vehicle routing: handling edge exchanges, E. H. Aarts, J. K. Lenstra, eds. *Local Search in Combinatorial Optimization.* John Wiley & Sons, Chichester, UK. (1997), 311–336.
48. S. Lin and B.W. Kernighan, “An effective heuristic algorithm for the traveling salesman problem,” *Operations Research*, vol. 21, pp. 498–516, 1973.
49. Yun-Chia Liang, S. Kulturel-Konak, A.E. Smith “Meta heurustic for the orienteering problem”, in *Proceedings of the 2002 congress on Evolutionary Computation*, Honolulu, USA
50. V. Maniezzo, Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, *INFORMS Journal of Computing* 11(4) (1999), 358--369.
51. V. Maniezzo, A. Carbonaro (2000), An ANTS Heuristic for the Frequency Assignment Problem, *Future Generation Computer Systems*; 16, North-Holland/Elsevier, Amsterdam, pag 927 – 935.
52. V. Maniezzo, A.Carbonaro (2001), Ant Colony Optimization: an overview, in C.Ribeiro (eds.) *Essays and Surveys in Metaheuristics*, Kluwer, pag.21-44.
53. V. Maniezzo and A. Colorni, The ant system applied to the quadratic assignment problem, *IEEE Trans. Knowledge and Data Engineering* 11(5) (1999), 769--778.
54. V. Maniezzo and A. Carbonaro, A bionomic approach to the capacitated p-median problem, *Future Generation Computer Systems* 16(8) (2000), 927--935.
55. V. Maniezzo and R. Montemanni, An exact algorithm for the radio link frequency assignment problem, *Technical Report* CSR99-02 (1999).
56. V. Maniezzo, A. Mingozzi, and R. Baldacci, A bionomic approach to the capacitated p-median problem, *Journal of Heuristics* 4(3) (1998), 263--280.
57. G. Navarro Varela, M. C. Sinclair, Ant Colony Optimization for virtual wavelength – path routing and wavelength allocations, in *Proceeding of the Congress on Evolutionary Computation* (CEC ’99), Washington DC, USA
58. J.Y. Potvin and S. Bengio, The vehicle routing problem with time windows - part {II: genetic search, *Informs Journal of Computing* 8 (1996), 165--172.
59. R.S. Parpinelli, H.S. Lopes, A.A. Freitas, Data mining with ant colony optimization algorithm, in *IEEE trasactions on Evolutionary Computation*, Volume 6, August 2002
60. G. Reinelt, *The traveling salesman: computational solutions for TSP applications*. Springer-Verlag, 1994.
61. Y. Rochat and E.D. Taillard, Probabilistic diversification and intensification in local search for vehicle routing, *Journal of Heuristics* 1 (1995), 147--167.

62. H. Shyh-Jier, Enhancement of hydroelectric generation scheduling using ant colony system based organization approach, in *IEEE Transactions of Energy Conversion*, Volume 16, September 2001
63. T. Stützle, An Ant Approach to the Flow Shop Problem, *Proceedings of EUFIT'98* , Aachen, pag. 1560-1564, 1998.
64. M.W.P. Savelsbergh, An efficient implementation of local search algorithms for constrained routing problems. *European Journal of Operational Research* **47** (1990), 75–85.
65. T. Stützle and M. Dorigo, A Short Convergence Proof for a Class of ACO Algorithms, *IEEE Transactions on Evolutionary Computation*, 6 (4), 2002 (in press).
66. T. Stützle and M. Dorigo, Aco algorithms for the quadratic assignment problem, *New Ideas in Optimization*, McGraw-Hill, London, 1999, pp. 3--50.
67. E. Siegel, B. Denby, S. Le Hégarat-Mascle, Application of ant colony optimizatio to adaptive routing in a leo telecommunications satellite network, submitted *to IEEE Trasactions on Networks*, july 2000
68. T. Stützle, A .Grün, S. Linke, M. Rüttger, A comparison of nature inspired heuristic on the traveling salesman problem, In Deb et al, editors, Proceedings of PPSN-VI, *Sixth International Conference on Parallel Problem Solving from Nature*, volume 1917 of LNCS, pages 661-670, 2000
69. T. Stützle, H. H. Hoos, MAX-MIN ant system, *Future Generation Computer Systems*, Vol. 16 (2000)
70. T. Stützle and H. Hoos, Improvements on the ant system: Introducing $max-min$ ant system, Proceedings of ICANNGA'97, *Int. Conf. on Artificial Neural Networks and Genetic Algorithms*, Springer Verlag, Vienna, 1997.
71. P. Shaw, Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming* (CP '98), M. Maher and J.-F. Puget (eds.), Springer-Verlag, 1998, 417-431.
72. T. Stützle and H. Hoos, Ant system and local search for combinatorial optimization problems, S. Voss, S. Martello, I.H. Osman and C. Roucairol editors, vol. *Meta-Heuristics: Advanced and Trends in Local Search Paradigms for Optimization*, Kluwer, Boston, 1998.
73. T. Stützle, H. Hoos The MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems: Towards Adaptive Tools for Combinatorial Global Optimization In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, pages 313-329, 1998.
74. M. Schreyer, G. R. Raidl, Letting ants labeling point feature, in *Proceedings of the 2002 congress on Evolutionary Computation*, Honolulu, USA
75. E.D. Taillard, Comparison of iterative searches for the quadratic assignment problem, *Location Science*, 3, 1995, pag. 87-105.
76. C. Tsai, C. Tsai, A new approach for solving large traveling salesman problem using evolutionary ant rules, *in Proceeding of the 2002 International Joint Conference on Neural Networks*
77. C. Tsai, C. Tsai, C. Tseng, A new approach for solving large traveling salesman problem, in *Proceeding of the 2002 Congress of Evolutionary Computation*
78. C.J. Watkins and P. Dayan, Q-learning, *Machine Learning*, 8:1992, 279--292.

79. O. Wittnr, B. E. Helvik, Cross-entropy guided ant-like agents finding dependable primary/backup path patterns in networks, in *Proceedings of the 2002 congress on Evolutionary Computation*, Honolulu, USA
80. B. Wu, Y. Zheng, S. Liu, Z. Shi, CSIM: a document clustering algorithm based on swarm intelligence, in *Proceedings of the 2002 congress on Evolutionary Computation, Honolulu*, USA
81. W. Ying, X. Jianying, Ant colony optimization for multicast routing, in *the 2000 IEEE Asia-Pacific Conference on Circuits and Systems*, Tianjin, China