Chapter 9
Ant Colony Based Algorithms for Dynamic Optimization Problems

Guillermo Leguizamón and Enrique Alba

Abstract. The use of metaheuristic approaches to deal with dynamic optimization problems has been largely studied, being evolutionary techniques the more widely used and assessed techniques. Nevertheless, successful applications coming from other nature-inspired metaheuristics, e.g., ant algorithms, have also shown their applicability in dynamic optimization problems, but received a limited attention until now. Different from perturbative techniques, ant algorithms use a set of agents which evolve in an environment to construct one solution. They cooperate by means of asynchronous communications based on numerical information laid on an environment. This environment is often modeled by a graph which constitutes a formalism with a great expressiveness, specially well-suited for dynamic optimization problems. A solution could be a structure like a subgraph, a route, a short path, a spanning tree, or even a partition of vertices. In this chapter we present a general overview of the more relevant works regarding the application of ant colony based algorithms for dynamic optimization problems. We will also highlight the mechanisms used in different implementations found in the literature, and thus show the potential of this kind of algorithms for research in this area.

9.1 Introduction

Metaheuristic techniques have widely proved to be suitable approaches for dynamic environments. In this regard, it should be noticed that Evolutionary Algorithms (AEs) are without any doubt the pioneer and more widely used metaheuristic [30].

Guillermo Leguizamón
Universidad Nacional de San Luis,
Av. Ejército de Los Andes 950 (5700), San Luis, Argentina
e-mail: legui@unsl.edu.ar

Enrique Alba
Universidad de Málaga, Campus de Teatinos (3.2.12)
Málaga - 29071, Spain
e-mail: eat@lcc.uma.es

However, Ant Colony Optimization (ACO) metaheuristic has also shown to be an efficient candidate to deal with this kind of problem [35]. In a broad sense, ACO [17] refers to a class of algorithms whose design is mainly based on the foraging behavior of real ants, being the more representative ACO algorithms those designed for solving a certain type of combinatorial optimization problem: those problems for which a solution is obtained by simulating a walk through a construction graph. One of the more studied problems under the above (original) vision of ACO algorithms is the Traveling Salesperson Problem (TSP), a well-known and classical NP-complete problem that owns important features that can be easily exploited to show the applicability of this metaheuristic. Several ACO algorithms were designed since the publication of the first ACO, the so-called Ant System (AS) by Dorigo et al. [16]. These algorithms include (see Figure 9.1 where each algorithm name includes the main keyword that respectively defines it) [17]: elitist-AS (an AS with an elitist strategy for updating the pheromone trail levels), AS rank (a rank-based version of Ant System), *MAX-MIN* Ant System (an AS that incorporates a mechanism to control the pheromone levels), the Ant Colony System (ACS) (a more advanced ACO algorithm with a modified transition rule, with local and global pheromone update), the Best-Worst Ant System (an extended AS characterized by integrating some components taken from evolutionary computation), ANTS (an AS that uses lower bounds on the completion of a partial solution to derive the heuristic desirability), and ANT-Q (a version of an AS that combines concepts of the reinforcement learning theory). Many of them were initially applied to TSP and also to the Quadratic Assignment Problem (QAP) [34]. After that, many other variants of these algorithms have been proposed in the literature. However, not all of them strictly follow the standard design principles given for the ACO metaheuristic in [17]. Instead, these algorithms follow in many different ways the metaphor of the ants behavior (foraging or some other) as a general framework which let the researcher broaden the field of application of this nature-inspired approach. In a more general sense, algorithms that follow in some way the above mentioned metaphor could be called Ant Colony Based algorithms (ACB).

Ant colony based algorithms have proven to be successfully applied to many different real world and academic problems that include combinatorial optimization problems, continuous domains, and also dynamic optimization problems (DOPs), the kind of problems and applications considered in the current chapter.

The rest of the chapter is organized as follows. The next section describes the ACO metaheuristic as the more representative ant colony based algorithm. Section 9.3 gives a general description of the type of dynamic problems usually found in the literature, as well as in real-world applications. Section 9.4 presents the mechanisms used in ACB algorithms to deal with dynamic problems, and a survey of the works in the literature. The last section contains some conclusions and outlines some future challenges.

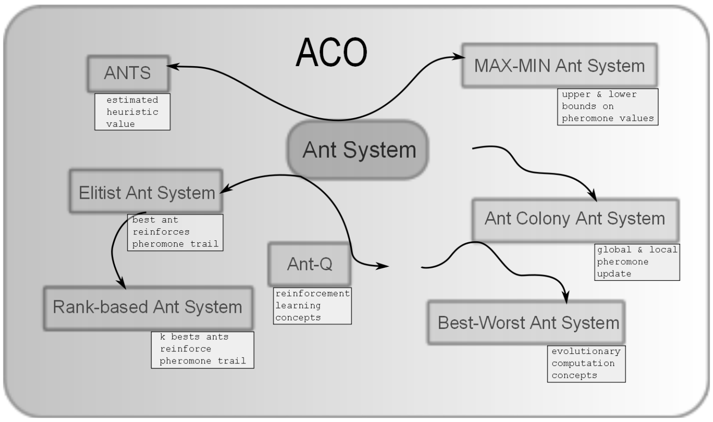
ACO
ANTS
estimated heuristic value
MAX-MIN Ant System
upper & lower bounds on pheromone values
Ant System
Elitist Ant System
best ant reinforces pheromone trail
Ant Colony Ant System
global & local pheromone update
Ant-Q
reinforcement learning concepts
Rank-based Ant System
k bests ants reinforce pheromone trail
Best-Worst Ant System
evolutionary computation concepts

Fig. 9.1 The most representative and widely used algorithms based on the ACO metaheuristic.

9.2 Ant Colony Optimization

As mentioned earlier in this chapter, ant colony based algorithms involve a broad class of optimization algorithms designed under the metaphor of real ants behavior. More precisely, ant colonies are social insect societies that can be considered distributed systems composed of simple interacting individuals. From the interaction between those individuals, a complex and highly structured social organization arises. Different types of ant colonies self-organize according to a particular behavior, e.g., foraging, division of labor, brood sorting, and cooperative transport. In the case of foraging ants, as well for the remaining behaviors, the activities are coordinated through indirect communication known as stigmergy [23]. A foraging ant deposits a chemical substance (pheromone trail) on the ground to communicate to other ants the desirability of following a particular path. At the same time, as more intense is the pheromone trail sensed on the ground, more amount of the chemical substance is deposited by a particular ant. From this autocatalytic or positive feedback process emerges a self-organized system in which the shortest paths connecting the nest and the food source remain candidates to follow by the whole colony. In addition to the above, it is worth noticing that an evaporation process occurs in the environment which helps the colony to keep the exploration capabilities during the search of alternative paths to the food source.

The foraging ants behavior just described as well as other types of behavior are taken as useful metaphors to design optimization and search algorithms under different names. In this work we adopted the definition of the Ant Colony Optimization (ACO) metaheuristic, as given by Dorigo and Stützle [17], which is the classical and more widely used formulation for this type of algorithms. Nevertheless, other ant colony based algorithms that not strictly follow the definition of the ACO

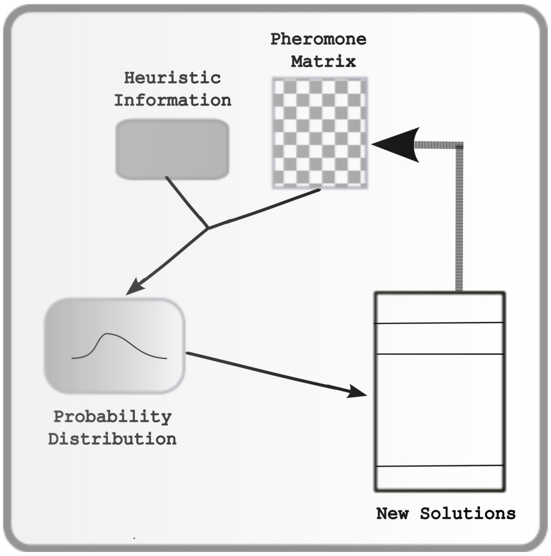

Fig. 9.2 A general overview of the behavior of an ACO algorithm: pheromone trail plus heuristic information are used to find the probabilistic distribution to generate new solutions. These new solutions are then used to update pheromone trails to bias the search for the next iterations.

metaheuristic will be also described in further sections. It is well-known that ACO is one of the most representative metaheuristics derived from the broad concept known as swarm intelligence, where the behavior of social insects is the main source of inspiration. As a typical swarm intelligence approach, the ACO metaheuristic is mainly characterized by its distributiveness, flexibility, capacity of interaction among simple agents, and its robustness.

ACO algorithms (in the standard version) generate solutions for an optimization problem by a construction mechanism where the selection of the solution component to be added at each step is probabilistically influenced by pheromone trails and (in most of the cases) heuristic information. Thus, the solution construction process is mainly influenced by pheromone trails and heuristic information[1] from which a probabilistic model is evolved to guide to exploration of the search space.

This means that the construction process probabilistically builds step by step the problem solutions and the probabilistic model has a feedback for its modification based on the solutions found. Figure 9.2 displays a general overview of an ACO algorithm. Heuristic information plus pheromone values are used to find a probability distribution over the search space. Initial pheromone values are in general set to a constant value, thus, at the first iterations the algorithm is highly explorative.

[1] The use of heuristic information is not mandatory, if used, this information is also taken into account to build a probabilistic model.

The probability distribution is then used to sample new solutions to the problem at hand. According to some criterion, all or a subset of these new solutions are involved in the pheromone update process. As the cycle displayed in Figure 9.2 is repeated, the pheromone values will bias the search to specific regions of the search space as it directly influences the probability distribution.

Although the application domains of ACO algorithms are certainly diverse, the more and well-known field is that related to combinatorial optimization problems like TSP [15, 16], QAP [21, 44], and Vehicle Routing Problem (VRP) [12, 41]. Thus, we present in the following some important considerations when applying this kind of algorithm to discrete problems as the mentioned above.

First of all, it is necessary to define an appropriate problem representation. In the jargon of ACO metaheuristic, this means to properly define:

(i) the construction graph and the way this represents the different problem components and connections among them,
(ii) the definition (if any) of the problem information to be exploited,
(iii) the behavior of the artificial ants, in the sense of how each ant will walk through the construction graph to build the corresponding solutions.

Algorithm 9.1. ACO algorithm

```
1: Init();
2: while not (termination-condition) do
3:     Build-Sols-Step-by-Step();
4:     Pheromone-Update();
5:     Daemon-Actions(); // Optional step
6: end while
```

A general design of an ACO algorithm (as showed in Algorithm 9.1) includes a set of four main activities (or steps) that define this iterative search technique. It must be noticed that variations of the way these activities are implemented define the kind of ACO Algorithm. For example, a variation on the activity Pheromone-Update() will mainly define Algorithm 9.1 as an Ant System (AS), elitist-AS, AS-rank, *MAX-MIN* Ant System, or an ACS (in this last case, Build-Sols-Step-by-Step() activity also involves a local pheromone update step). Also, any other ant colony based algorithm not necessarily fitting exactly in the canonical definition of those algorithms could be included in the family of ACO algorithms. Nevertheless, the activities in Algorithm 9.1 can be described in a general way in the following. To do that we assume an Ant System applied to TSP, thus the problem components are cities and connections (routes) between them. The connections have an associated value, e.g., distance between cities or cost to travel from one city to another:

- **Init():** As in any typical population-based algorithm, some basic tasks need to be done before starting the exploration of the search space. In this case, the initialization of pheromone trail matrix which at time 0 is:

$$\tau_{ij}(0) = \tau_0, \text{ for } i,j \in \{1,\ldots,n\}, \tag{9.1}$$

where n represents the problem size and τ_0 is a small constant value (e.g., the following value is suggested for TSP in [17]: $\tau_0 = m/C^{nn}$, where m is the number of ants, and C^{nn} is the length of a tour generated by the nearest-neighbor heuristic). Also the heuristic values (represented by η symbol) are calculated (when available and used) and any other structure, necessary to complete the problem representation, is initialized.

- **Build-Sols-Step-by-Step():** This activity involves the release of an independent colony of artificial ants in charge of incrementally building a solution to the problem. Each ant, at each step of the construction process, makes a local stochastic decision about the next component to be included in the solution under construction. For example, for an Ant System applied to TSP, the decision of adding city j (problem component) to the solution under construction when city i was the last visited is given by:

$$p_{ij}^{k} = \begin{cases} \frac{\tau_{ij}(t)^{\alpha}\eta_{ij}^{\beta}}{\sum_{h\in\mathscr{N}^k(i)} \tau_{ih}(t)^{\alpha}\eta_{ih}^{\beta}} & \text{if } j \in \mathscr{N}^k(i) \\ 0 & \text{otherwise,} \end{cases} \tag{9.2}$$

where α and β are the parameters that, respectively represent the importance of the pheromone trail ($\tau_{ij}(t)$) and the heuristic information (η_{ij}), and $\mathscr{N}^k(i)$ represents the set of cities that can be visited by ant k, i.e., the feasible or unvisited cities.

- **Pheromone-Update():** The acquired experience achieved at each iteration by the colony is considered in this activity. High quality solutions will positively affect the amount of pheromone trail, i.e., those edges that are part of solutions found will receive an increased amount of pheromone trail according to the goodness of these solutions. This is known as the global pheromone update.

  As in Nature [22], a process of pheromone evaporation takes place (usual implementations of this metaheuristic decrease the amount of pheromone trail for all edges in the construction graph) [9]. Thus, the amount of pheromone corresponding to those edges that are not part of any solution at the current iteration will show a gradually diminishing pheromone intensity. It should be noticed that some ACO algorithms, such as ACS [15], apply a local pheromone update rule which does not depend on the solution quality. Instead, a fixed amount is deposited as soon as an edge in the construction graph is selected to make the move (the next component added to the solution under construction). The following equation is a possible way of (global) updating pheromone values:

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}, \tag{9.3}$$

  where ρ is the evaporation rate and $\Delta\tau_{ij}$ represents the amount of pheromone trail deposited in edge (i,j). That amount is calculated according to the quality of the solutions, found by the whole colony, that include edge (i,j).

- **Daemon-Actions():** As single ants cannot carry out centralized actions, many ACO algorithms include some specific activities called *daemon actions*. Examples of these activities are: activation of a local search procedure or a collection of global information (e.g., use of a set of the best ranked solutions) that could be employed to modify some entries in the pheromone trail matrix.

Based on the above description, several types of ACO algorithms can be obtained [17]. In addition, many other algorithms that not necessarily follow the standard design can be considered either for discrete problems [9], continuous domains [43], and dynamic optimization problems [28] as we will show in the next sections.

9.3 Dynamic Optimization Problems (DOPs)

Dynamic optimization problems (DOPs) involve any problem definition for which at least one of its components varies with time. Thus, we can find problems where the objective function changes over time or some problem constraints depend on environmental conditions. These situations include many real-world tasks for which changes in the environment affect the applied optimization process, as this has to react to the new environmental conditions.

In this section we present the more relevant concepts involved in DOPs that are usually considered when applying metaheuristic techniques[2] for solving them. A classical reference to the use of Evolutionary Algorithms (EAs) for dynamic problems is given in Branke [6] in which a widely referenced classification of dynamic problems is presented, and the fundamentals of possible mechanisms to deal with are also analyzed.

In order to obtain a fairly self-contained chapter we succinctly describe some important concepts regarding the mechanism to deal with certain dynamic optimization problems. The interested reader can find a good source of information in this regard in Branke [6, 30] and Morrison [37]. In addition, a complementary source of information can be found in Leguizamón et al. [33] where alternative metaheuristic techniques to deal with dynamic optimization problems under a unified perspective are described.

Any change in a dynamic problem can be seen as the activation of a new optimization problem (replacing the previous one) for which a new solution must be provided as the quality of the current solution could be no longer acceptable for the new environment. Therefore, the adaptation of the current solution to the new problem will be the main objective when a change occurs. The most primitive mechanism to deal with dynamic problems is restarting from scratch, i.e., the optimization algorithm does not consider any previous experience or information that could be helpful under the new conditions. However, this approach is impractical for many reasons and alternative mechanisms should be taken into account that consider in different ways the past experience on the search. This is particularly desirable when

[2] We are assuming through this chapter population-based metaheuristic techniques.

the solution for the new problem should be not too different from the solution previously found for the (probably related) old problem. Also, the past experience is a valuable element, as many dynamic problems are defined as a sequence of static instances of a basic problem with slight variations from one instance to another, resulting in a sequence that defines the complete dynamic problem. In that regard, it is worth noticing that the quantity and quality of the past experience considered for an optimization algorithm will determine the capacity of such algorithm to adapt to a possibly continuously changing environment. In addition, the control of the population diversity is a key factor, as the adaptation to the changes could be harder if the algorithm rapidly losses diversity.

The use of explicit or implicit memory to remember past experience is the typical approach implemented in metaheuristic techniques to guide the exploration of the search space. However, some metaheuristics implement by definition some sort of memory of the past experience as a mechanism to bias the search during the incoming iterations; also this "natural" memory can be used to deal with dynamic problems. This is the case of ant colony based algorithms, as will be described in the next section.

Many types of dynamic features can be found in real-world problems. From earlier application of evolutionary algorithms to dynamic problems (see for example, Abdunnaser [1], Bianchi [7], Branke [6], and Psaraftis [31]) the following elements and features are commonly considered when dealing with DOPs:

- the problem can change with time in such a way that future scenarios are not completely known, yet the problem is completely known up to the current moment;
- a solution that is optimal or near optimal at a certain time may reduce its quality in the future, or may even become infeasible;
- the goal of the optimization algorithm is to track the shifting optima through time as closely as possible;
- solutions cannot be determined ahead of time but should be found in response to the incoming information; and
- solving the problem entails setting up a strategy that specifies how the algorithm should react to environmental changes, e.g., to solve the problem from scratch or adapt some parameters of the algorithm at every change.

Besides the described features, DOPs can be classified in different ways depending on the sources of dynamism and its effects on the objective function. Simple questions will help us to determine the nature of the change: i) *what?* (i.e., aspects of change), ii) *when?* (i.e., frequency of change), and iii) *how?* (i.e., severity of the change, effect of the algorithm over the scenario, and presence of patterns).

Different descriptions for dynamic problems are given in the literature [6, 37] and some of them were proposed having in mind certain type of metaheuristic algorithm that could be used for solving them [8, 11, 28]. Nevertheless, before applying any algorithm to solve a particular dynamic problem, the dynamic nature of the problem must be taken into account as well as the capability of the chosen algorithm to react to the changes. Next section discusses different ant colony based algorithms and the way they were applied to dynamic problems.

9.4 Solving DOPs with ACB algorithms

Different metaheuristic approaches have been applied to dynamic optimization problems, being EAs the more widely used search technique to deal with this kind of problems [6, 8, 28]. However, other well-known metaheuristics have also been increasingly and successfully applied to DOPs. Some of them include Nature inspired metaheuristics like the ACO and Particle Swarm Optimization (PSO). For example, a recent short survey by Hendtlass et al. [28] examines some representative works and methodologies to deal with DOPs by using Ant Colony Optimization, Particle Swarm Optimization, and Extremal Optimization. Besides describing the applications based in these Nature inspired metaheuristics, the authors also analyzed some limitations of the presented algorithms. Other short review in this regard (see the technical report from Angus [2]) focuses on the ACO metaheuristic and describes some relevant and related works. Besides the previous surveys on this topic, we give here a unified and broad perspective of the different ant colony based algorithms to deal with DOPs.

There are two concepts closely related in any algorithm dealing with DOPs: 1) the mechanisms implemented to avoid stagnation and hence 2) the capacity of the algorithm *to react to the changes*. Particularly in the ant colony based algorithms the pheromone structure τ presented in Section 9.2 is the key algorithm component that should mainly be taken into account, as this represents, on one hand, the memory of the algorithm. On the other hand, the pheromone structure is built in the solution search space by a graph and that dynamics modify this graph valuated by variable amounts of pheromone. Thus, in any ant based algorithm an appropriate strategy must be defined to let the algorithm adapt to the changes by modifying the pheromone values, e.g., by resetting all or part of the pheromone values to reduce, in some extent, the learnt experience. This is a classical strategy for increasing an explorative behavior as the new scenario has been detected.

The seminal works of the ACO metaheuristic to deal with dynamic optimization problems (see [24, 26, 27, 36]) were mostly devoted to the TSP, QAP, and VRP. All the dynamic versions of these problems have the following characteritic: the changes are produced by adding/eliminating problem components, i.e., cities in TSP, locations in QAP, and orders in VRP. However, other alternatives are also possible like changing the problem data, e.g., distance between cities in TSP, cost of assignment in QAP, or cost of delivery between the depot and customers in VRP.

In the following we present in three sections a short review of literature and a global description of the respective mechanism used to deal with certain types of DOPs in the past. The sections are divided according to the following criteria: Section 9.4.1 presents the so-called standard versions of ACO algorithms, Section 9.4.2

describes the class of population-based ACO (P-ACO) algorithms, and finally in Section 9.4.3 some general algorithms based in the metaphor of ants behavior are presented.

9.4.1 Standard ACO Algorithms

One of the seminal works regarding this type of algorithm is the proposal by Guntsch et al. [27]. Authors there investigate several strategies for pheromone updating in reaction to changes on the problem instance. In this case, the problem instance is changed by adding/eliminating cities.

Regarding dynamic TSP, Angus and Hendtlass [3, 4] study another strategy to modify the pheromone matrix when a change occurs. In this case, the strategy consists in normalizing the pheromone values in a way that the past memory is maintained but avoiding extreme values.

Another proposal to solve dynamic TSP is presented by Eyckelhof and Snoek [18]. They study two different ways of modifying pheromone trails: local and global. In the dynamic TSP studied, the distance between cities is seen as the time to travel from one city to another one. Thus, the changes are obtained by modifying the traveling times and hence, traffic jams could be produced in the paths as the ants are walking while solving the problem. The strategy proposed logarithmically smooths the pheromone values (called *shaking* process) maintaining the relative ordering among them before and after the modification. The global shaking produces a modification on all over the edges while the local one only changes the pheromone values around a certain distance where the traffic jam was produced. The strategy also avoids to assign pheromone values below a certain lower bound.

Montemanni et al. [36] investigate the Dynamic Vehicle Routing Problem (DVRP) through an Ant Colony System based algorithm. In the studied version of the DVRP new orders can arrive at any time and they have to be dynamically incorporated in the constantly evolving schedule. In the particular case of [36] new orders can be assigned after the vehicle left the depot. The mechanism for pheromone updating is inspired by the work by Guntsch and Middendorf [24, 25]. The strategy adopted evaporates the old pheromone values and at the same time increases the amount of pheromone values by a constant amount.

The Binary Ant Algorithm (BAA) proposed by Fernandes et al. [20] is designed by using a particular construction graph to work on binary dynamic environments. The main characteristic of BAA is that it stresses the role of the negative feedback (i.e., give more relevance to the evaporation processes). BAA was tested on two dynamic continuous functions: Oscillatory Royal Road and Dynamic Schaffer.

In summary, several strategies have been proposed in the literature to deal with DOPs by applying the classical ACO algorithms (i.e., those which more closely follow the principle of using a construction graph as earlier defined in [17]):

- Global pheromone modification:
  - Increase the values proportionally to their difference to the maximum pheromone value.
  - The new pheromone values are a combination of the old values and an increment of a constant small pheromone value. Those values are regulated by a parameter $0 \leq \gamma_r \leq 1$ that controls the relative importance of both values:

$$\tau_{i,j}^{new} = (1-\gamma_r) \cdot \tau_{i,j}^{old} + \gamma_r \cdot \tau_0, \tag{9.4}$$

  where τ_0 is a small constant value (usually used in the pheromone matrix initialization process); and $\tau_{i,j}^{new}$ and $\tau_{i,j}^{old}$ represent, respectively the pheromone value on edge (i, j) before and after the change in the environment.

- Local pheromone modification:
  - η-strategy (based on the heuristic information) and τ-strategy (based on pheromone information). They both refer to connection problems, thus, problem components can be inserted/deleted. Consequently, only the edges connecting the problem components must be added/eliminated from the construction graph will influence (locally) the pheromone values.
  - Combined strategies based on η-strategy and τ-strategy.
  - Normalize the pheromone values regarding the maximum pheromone value all over the current edges, i.e., $\tau_{ij}(t+1) = \tau_{ij}(t)/\tau_{i,max}(t)$. The term "current edges" is used here as the edges directly connecting component i with some other component and $\tau_{i,max}(t)$ indicates the maximum pheromone value in the neighborhood of component i. Thus, the normalization is local with respect to each component i of the problem instance.

- Local and global pheromone modification:
  - Using a mechanism to limit the lower pheromone values (similar to the *MAX-MIN* Ant System) and smoothing the pheromone values keeping the relative order before and after the change (this promote exploration without losing information of the past experience). The modification can be applied locally or globally (depending on an *ad-hoc* parameter).

- Other:
  - Keeping elitist ant: in this case the best-so-far ant is modified after a change has occurred (e.g., adding/eliminating solution components). Thus, the eliminated components are deleted from the elitist solution, whereas the new added components (if any) are located in the places left in the solution. The new components could be added, for example, by using some heuristic procedure. This is a possible way of remembering the past experience that will influence the pheromone values when the usual step of pheromone updating takes place.
  - Choosing only the ants with a quality value equal to or below[3] the population's average fitness to generate new solutions. This strategy is combined with a quick evaporation of pheromone trails.

[3] For a maximization problem the value is above or equal to the average.

Please notice that the above strategies could be adapted or enhanced in many ways to deal with unseen dynamic problems. In this regards, the approach to use and adapt the pheromone matrix is the more important issue in order to achieve an efficient mechanism to react to the changes.

9.4.2 Population Based ACO Algorithms

Population based ACO algorithms (P-ACO) were first proposed by Guntsch and Middendorf [26] as an alternative ACO in which a set of solutions is transferred between the current iteration and the next one instead of the pheromone trail values (or pheromone matrix). Thus the set of transferred solutions is used to calculate the new pheromone values used later for building the new set of solutions. Although not applied to dynamic problems, FIFO-Queue ACO ([26]) could be considered as a preliminary work of P-ACO in this direction as the authors claimed about its applicability to create new metaheuristic algorithms, as well as to solve dynamic problems, since P-ACO could rapidly adapt the pheromone values as the environment changes in comparison with standard ACO algorithms. In Guntsch and Middendorf [25] FIFO-Queue was studied on dynamic versions of QAP and TSP, where different strategies for updating the population were investigated. More recently, Ho and Ewe [29] proposed a P-ACO algorithm where three different mechanisms to adapt the pheromone values are investigated to solve the dynamic load-balanced clustering problem in ad hoc networks. Although the proposed mechanisms are novel, they only represent slight variations of the original P-ACO [26]. For that reason, we describe first the main features of P-ACO, as proposed by Guntsch and Middendorf [26] and then we give some highlights of the variations of P-ACO according to Ho and Ewe [29].

It is important to note that P-ACO algorithms also adopt the principle of the construction graph as in the standard ACO described in the previous section. However, we have decided to describe them in a section apart. The reason for that is because they differ from the standard ACO algorithms. On one side, ACO algorithms include problem-based strategies in the pheromone updating process. On the other side, P-ACO algorithms use the solutions themselves to define the strategies to calculate the new pheromone values. Another interesting feature of P-ACO is that this algorithm does not use any problem information to handle the changes. This makes it a convenient approach for different types of dynamic as the basic information needed is that provided by the fitness function.

The following strategies are the alternatives implemented in [26] to manipulate the set of solutions maintained (i.e., the population) in order to influence in different ways the pheromone values:

1. *Age strategy:* Add the best solution found in the current iteration and remove the oldest one.
2. *Quality strategy:* Add the best solution found in the current iteration and remove the worst one.

3. *Prob strategy:* Add the best solution found in the current iteration and remove a solution randomly chosen. To do that a distribution probability is created in a way that bad solutions have more chances to be removed; however, all solutions are candidate to be eliminated.

Combinations of two of the above three strategies also exist, as used and tested in [26]. It must be noticed that the above strategies were designed for controlled changes where the number of cities (problem components for the general case) that are added is the same as the number of deleted ones, i.e., the problem size is kept constant.

Under the population-based ACO the *KeeepElite* strategy, first defined in [27], can also be applied to repair the solutions in the population regarding the changes that took place in the environments, i.e., deleted problem components are eliminated from the elite solution whereas the new inserted ones are added based on some criterion.

Instances considered in [26] include two additional parameters regarding the dynamic features:

- c, which indicates the severity of the change, i.e., the number of components deleted (respectively inserted) from (to) the problem instance.
- t, the time window that controls the occurrence of the change.

No overall definitive conclusion about the use of the different strategies in P-ACO algorithms can be achieved according to the results presented in [26]. However, strategy *Prob* was the worst performer for all the scenarios considered in the experimental study. Also, a small number of solutions in the population (size of the population $k = 3$) was enough to perform fairly well on all the problems (TSP and QAP) and considered instances.

Three variations of P-ACO (see Ho and Ewe [29]) were studied when solving the dynamic load-balanced clustering problem in ad hoc networks. The changes here operate on the problem structure. The variations presented are intended to adapt the pheromone values as closely as possible to reflect the new problem structure after a change has occurred by:

i) Applying a repairing process (based on the new problem structure) to the solutions in the population.
ii) Adapting the parameters that control the importance of the pheromone and heuristic values: parameters α and β as used in standard ACO algorithms (see Equation 9.2). Thus, more importance is given to the heuristic values (they are recalculated when the problem changes) and less importance to the pheromone values (forgetting past experience). The adapted parameters are used by a percentage of ants of the whole colony.
iii) Combining two approaches to build new solutions: greedy and pheromone based. Thus, a percentage of ants have a greedy behavior since they disregard the accumulated pheromone values and only consider the new problem structure. The remaining ants follow the usual steps to find a solution based on the accumulated pheromone values.

The above P-ACO variations were called, respectively P-ACO, PAdapt, and GreedyAnts. These algorithms were tested on 12 instances representing 4 ad hoc networks. The best performer algorithms were P-ACO and GreedyAnts. The authors highlight that all the algorithms experience some difficulties to react to the first change, however, they improved the respective capacity of reaction in subsequent changes of the problems structure.

To finish this section we would like to remark that only a few applications of P-ACO for dynamic problems were found in the literature. Nevertheless, it is interesting to note, as claimed by Guntsch and Middendorf [26], that P-ACO has a potential to solve other dynamic problems.

9.4.3 Other Ants Based Algorithms

In addition to the standard ACO algorithms (Section 9.4.1) and Population-based ACO algorithms (Section 9.4.2), there exist other alternatives to implement algorithms based on the metaphor of ants behavior which do not completely fit in the mentioned two classes. In this section we describe some representative works in this regard in order to show the reader the potential of following the ant behavior metaphor to solve unseen dynamic problems. An earlier ant-based method to deal with dynamic problems is the proposal by Schoonderwoerd et al. [42] where an ant-based load balancing approach is applied to telecommunication networks. This methodology considers the ants as mobile agents that can travel though the network with similar abilities as the real ants, as they can deposit certain amount of pheromone according to the distance between two pair of nodes and the congestion found during the journey.

The implemented mechanism to route the call is fairly simple as the pheromone value deposited on the route connecting two nodes is used to calculate a probability distribution that will influence the decision maker to route a call. Based on the experimental study accomplished, the authors claim that good load balance can be reached due to the emergent organization of the proposed ant-based methodology (thus, other related problems could also be solved).

Another methodology where the ants are seen as mobile agents is the AntNet [14], maybe one of the more widely known ant based algorithms different from standard ACO algorithms. This algorithm was designed as an alternative approach to the adaptive learning tables in communication networks, an intrinsically dynamic problem. AntNet follows the core ideas of the ACO metaheuristic: a set of independent ants (agents) try to find an optimal or near-optimal path by indirect communications.

Interestingly, there are two types of ants (*forward* and *backward*) which are distributed on the network (represented by small packages) to find paths from a source node to a destination node (the task of forward ants) and propagate the collected information on the routing tables (the task of backwards ants). In short, during their travel ants collect information about the network traffic which is later used to adapt certain values (resembling pheromone values) used by the decision maker at the time of routing the actual data packages. AntNet was thoroughly tested on real and artificial IP datagram networks and achieved superior performance with respect to the other algorithms tested for comparison. In the same mobile agent approaches as AntNet and Schoonderwoerd's ABC, Pigné and Guinand [39] consider the problem of mobile ad hoc networks where, prior to the throughput and the quality of service, the problem of energy consuming is the biggest issue in these wireless networks, composed of small handheld devices. The authors propose a bi-objective model where the length of communication paths is minimized and the selection of robust (unlikely to fail) links is maximized. The approach is decentralized and only relies on local rules for heuristics and pheromone updates.

In Cicirello and Smith [13] the so-called Ant Colony Control (AC^2) is proposed to an adaptive and dynamic shop floor scheduling problem. AC^2 is a decentralized algorithm conformed by a set of artificial ants which use indirect communication to make all shop routing decisions. This is done by altering and reacting to a dynamically changing common environment through the use of simulated pheromone trails. Thus, the amount of pheromones will be used to control the reaction of the algorithm to the changes, i.e., the decision about to which processing shop a job should be assigned. Another interesting feature of AC^2 is that, as the shop can process job of different types, an ant associated to that job will have different pheromone type from an ant associated to a different type of job. According to this, AC^2 manages many pheromone matrices such as the number of different types of jobs the shop floor can process. From the experimental study (applied to different shop floors) the authors claim that, for complex problems, AC^2 evolves local decision making policies that lead to near-optimal solutions with respect to its global performance.

A recent work by Fernandes et al. [19] proposed an extension of the Univariate Marginal Distribution Algorithm (UMDA) (see Mühlenbein and Paas [38]) called Reinforcement-Evaporation UMDA (RE UMDA). This new algorithm includes a different update strategy for the probability model based on the equation of the transition probability equations found in ACO algorithms. To do that, RE UMDA uses two real vectors τ^0 and τ^1 that, respectively, represent the pheromone values associated with the desirability of having a 0 or 1 at a particular position in the solution (a binary search space it is assumed). These two vectors are then used to calculate the probability of assigning 1 (respectively 0) to a particular solution component, i.e., to generate new solutions. As mentioned before, the two pheromone trail vectors are updated (reinforcement stage) according to the solutions found in the current iteration before obtaining the probability to generate new solutions. The evaporation stage, which uses an evaporation parameter as in the traditional ACO algorithms, takes place when the new population has been completely generated. This new approach delays (or avoids) the complete convergence of the population which increases the chances to adaptation to a new environment when a change occurs. RE UMDA was tested on dynamic versions of Onemax and Royal Road

(R1) functions. In addition, several variations of specific parameters of UMDA were studied under the dynamic functions considered.

Xi et al. [45] propose an Ant Colony System (ACS) based methodology to deal with the curing of polymeric coating process, a complex and dynamic optimization problem of great interest in the automotive industry. This is a large-scale multi-stage dynamic optimization problem that involves a time variant objective function (energy consumption for the coating process) and also time dependent linear and nonlinear constraints. More precisely, the ACS-based methodology is called *a dynamic model-embedded ACS-based optimization methodology.* The solution search space is represented by N trees (N is the problem dimension) where each of them is traversed by a set of M ants in charge of cooperatively finding a complete solution by building a tour on each of the N trees, i.e., when ant j traverses tree i, a value for dimension i will be found by ant j. Pheromone and heuristic values are associated to each branch tree to bias the probability values that will guide the ants when traveling the respective trees from the root to the leaves. Each tree node is assumed to have L possible branches (i.e., L possible values for each problem dimension). As in a standard ACS, global and local pheromone updates are applied, as well as similar transition rule as the originally defined in ACS. The authors claim that by building the solutions in this way the algorithm can easily adapt the solutions to the current state of the problem environment to reach the minimum energy consumption. The proposed ACS-based methodology was successfully compared with a genetic algorithm (GA) as the first one was capable of decreasing in about 9% the energy consumption with respect a Genetic Algorithm. Although the reported results are encouraging, it is still necessary to study in more detail the changes produced in the values associated to the edges in the trees when a change occurs in the environment.

In a recent work, an ant-stigmergy based algorithm to solve dynamic optimization problems (DASA) was proposed by Korošec and Šilc [31]. Interestingly, DASA was first proposed to solve (static) problems in continuous domains (see [32]). More precisely, it was applied on a benchmark suite from the Special Session on Real Parameter Optimization of the International Congress on Evolutionary Computation (CEC) 2005. Interestingly, the same algorithm DASA was applied without any modification on the set of benchmark problems provided for CEC' 2009 Special Session on Evolutionary Computation in Dynamic and Uncertain Environments with encouraging results. The main idea behind the DASA design is the construction of a special graph called *differential graph* used by ants to build a solution (vector of real numbers) starting from a given solution called temporary best solution which is initially chosen at random. Each edge in the graph represents either an increment or decrement (Δ value) that have to be applied to a particular dimension as the ant walks through the graph. The decision to choose the next vertex to visit is based on the pheromone values which are initialized according to the Cauchy distribution. Although no exhaustive discussion is provided in this work concerning the features of DASA, the results showed a natural capacity of the algorithm when dealing with the kind of tested dynamic problems. In [5, 6] the authors propose to distribute agents or entities-based applications on a grid or network of computers using several distinct colonies of ants. Each colony represents a distinct computing resource identified by a color. The colored ants compete to detect and colonize evolving communities, or organizations in a dynamic graph representing the set of entities (nodes) and their interactions one with another (edges). The importance of the interaction can be used to weight the edges. Communities are commonly defined as areas of a graph where nodes are more connected one with another than with the other parts of the graph. Organizations are evolving communities, as the underlying graph changes during time. Indeed, often, although individual nodes and edges of a community appear, change, or disappear, the community remains stable for a longer period of time. The proposed algorithm (called AntCO^2) shows two important differences with standard ACB: it does not use an explicit objective function, and it uses several colonies of ants in competition one with another. The ants detect organizations of the evolving graph by laying down "colored" pheromone corresponding to their colony. Pheromone of the same color as an ant probabilistically attracts it, whereas other colors repulse it. Furthermore, the larger the weight on an edge, the more this edge attracts ants, and strongly connected areas capture ants. These mechanisms act as positive feedback to create "colonized" areas on organizations.

Both the evolution of the network (disappearance of edges and node) and the evaporation of pheromone act as negative feedback to remove old solutions (old colonized areas) that are no more valid when the environment changes, therefore providing adaption to dynamics of the graph. Indeed, there is no need to evaluate any objective function to use this algorithm, which allows it to be easily distributed, since it uses only local information. Colonies can be of distinct sizes (number of ants) to accommodate the difference in power of the corresponding computing resources, and therefore colonize larger areas to distribute more entities or agents on more powerful computers. Furthermore, colonies can be added or removed as computing resources appear or disappear, therefore providing another level of dynamism.

9.4.4 Summary of ACB Applications on DOPs

To finish the main section on ACB for DOPs, we show in Table 9.1 a summary of the main applications commented in this chapter. Table 9.1 displays in the first column the type of application considered (Application), the name and reference of the ant colony based algorithm used (ACB), and finally some remarks about the algorithm are given in the third column (Remarks). In column ACB the proposal is called as: Standard ACO (Section 9.4.1), P-ACO (Section 9.4.2), and the respective names found in the literature for other ant colony based algorithms (Section 9.4.3).

Table 9.1 Summary of the reviewed literature indicating the application, type of ACB or algorithms' names, and some additional remarks about the respective proposal.

Application	ACB	Remarks
Dynamic TSP	Standard ACO [3, 4, 18, 24] P-ACO [25, 26]	Mostly aimed to investigate different strategies for pheromone updating on dynamic problems where components are added/eliminated at certain times.
Dynamic QAP	P-ACO [25, 26]	Like the previous row.
Dynamic VRP	Standard ACO [36]	Idem above with new orders arriving when vehicles have already started their tours.
Load balancing in telecommunication newtorks	Ant-Based Control [42]	Use of simple mobile agents (ants) with abilities to laid pheromone trails. Pheromone tables are used to balance the load generated by calls between nodes.
Ad hoc networks	P-ACO, PAdapt, and GreedyAnts [29]	Variations of P-ACO are studied to manage in different ways the modification of the pheromone matrix. The algorithms use knowledge of the problem structure (dynamic component) to carry on the pheromone updating process.
Dynamic load balancing in individual-based simulations	AntCO^2 [5, 6]	Use of several ant colonies in competition to colonize communities in an evolving network of interacting entities.
Continuous functions	DASA [31]	An ant-stigmergy based algorithm originally designed for static continuous functions is successfully applied on a benchmark of dynamic continuous functions.
Shop floor scheduling problem	AC^2 [13]	Ants use only the stigmergy principle to make all shop routing decisions by altering and reacting to their dynamically changing common environment through the use of simulated pheromone trails.
Oscillatory Royal Road & dynamic Schaffer's function	Standard ACO [20]	Optimization of dynamic binary landscapes by stressing the role of negative feedback when modifying pheromone values.
Routing tables in communication networks	AntNet [14]	Tiny packages (ants) are used to collect and distribute information from the network to modify the routing tables.
Communications paths in wireless mobile ad hoc networks	Ant-based algorithm [39]	An ant colony constructs and maintains communication paths trying to minimize both, the length of the constructed paths and the number of link reconnections.
Curing of Polymeric Coating	ACS-based algorithm [45]	Utilization of tree structures to find quality values for each problem dimension. The traversing from the root to the leaves in the respective trees is governed by the deposited pheromone values. The pheromone updating is made by rules resembling those used in Ant Colony Systems.
OneMax & Royal Road	RE UMDA [19]	Use of principles of pheromone trail to keep diversity.

9.5 Conclusions

Many real-world problems are dynamic by definition and the use of metaheuristic techniques to solve them seems to be a good alternative, as those kind of algorithms are robust and flexible. Ant colony based algorithms share these particular features as they can be easily adapted to deal with dynamic problems.

In this chapter we presented a general perspective of the more relevant works regarding the application of ant colony based algorithms for dynamic optimization problems. The main mechanisms used in different implementations found in the literature were described. Interestingly, the metaphor of ant colony behavior could potentially be used in many different ways, which make ACB algorithms good candidates to solve known and unseen DOPs.

Promising research areas seem to be related with applications in which the metaphor of ants behavior (basically stigmergy by pheromone trail) can be used as a source of information to rapidly react to the changes. In that regard, it could be interesting to define and thoroughly study general strategies to adapt pheromone values on classes of dynamics problems as well as comparisons with other, more studied and applied, metaheuristics for solving DOPs, e.g., evolutionary algorithms.

Acknowledgments. The first author acknowledges funding from Universidad Nacional de San Luis (UNSL), Argentina and the National Agency for Promotion of Science and Technology, Argentina (ANPCYT). The second author acknowledges funds from the Junta de Andalucía (CICE), under contract P07-TIC-03044 (DIRICOM project) and Spanish Ministry of Sciences and Innovation (MICINN) and FEDER under contracts TIN2008-06491-C04-01 (M* project) and TIN2011-28194 (roadME project).

References

[1] Abdunnaser, Y.: Adapting Evolutionary Approaches for Optimization in Dynamic Environments. PhD thesis, University of Waterloo, Waterloo, Ontario, Canada (2006)

[2] Angus, D.: The current state of ant colony optimisation applied to dynamic problems. Technical Report TR009, Centre for Intelligent Systems & Complex Processes, Faculty of Information & Communication Technologies Swinburne University of Technology, Melbourne, Australia (2006)

[3] Angus, D., Hendtlass, T.: Ant Colony Optimisation Applied to a Dynamically Changing Problem. In: Hendtlass, T., Ali, M. (eds.) IEA/AIE 2002. LNCS (LNAI), vol. 2358, pp. 618–627. Springer, Heidelberg (2002)

[4] Angus, D., Hendtlass, T.: Dynamic ant colony optimisation. Applied Intelligence 23(1), 33–38 (2005)

[5] Bertelle, C., Dutot, A., Guinand, F., Olivier, D.: Organization Detection Using Emergent Computing. International Transactions on Systems Science and Applications 2(1), 61–70 (2006)

[6] Bertelle, C., Dutot, A., Guinand, F., Olivier, D.: Organization Detection for Dynamic Load Balancing in Individual-Based Simulations. Multi-Agent and Grid Systems 3(1), 42 (2007)

[7] Bianchi, L.: Notes on dynamic vehicle routing - the state of the art. Technical Report IDSIA 05-01, Istituto Dalle Molle di Studi sull' Intelligenza Artificiale (IDSIA), Manno-Lugano, Switzerland (2000)

[8] Blackwell, T.: Particle Swarm Optimization in Dynamic Environments. In: Yang, S., Ong, Y., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 29–49. Springer, Heidelberg (2007)

[9] Blum, C.: Ant colony optimization: Introduction and recent trends. Physics of Life Reviews 2(4), 353–373 (2005)

[10] Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers, Norwell (2002)

[11] Branke, J., Schmeck, H.: Designing evolutionary algorithms for dynamic optimization problems. In: Advances in Evolutionary Computing, pp. 239–262. Springer-Verlag New York, Inc. (2003)

[12] Bullnheimer, B., Hartl, R.F., Strauss, C.: An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research 89, 319–328 (1999)

[13] Cicirello, V.A., Smith, S.F.: Ant Colony Control for Autonomous Decentralized Shop Floor Routing. In: International Symposium on Autonomous Decentralized Systems, pp. 383–390. IEEE Computer Society, Dallas (2001)

[14] Di Caro, G., Dorigo, M.: AntNet: distributed stigmergetic control for communications networks. J. Artif. Int. Res. 9, 317–365 (1998)

[15] Dorigo, M., Gambardella, L.M.: Ant Colony System: a Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)

[16] Dorigo, M., Maniezzo, V., Colorni, A.: Ant system: Optimization by a colony of cooperating agents. IEEE Trans. on Systems, Man, and Cybernetics–Part B 26(1), 29–41 (1996)

[17] Dorigo, M., Stützle, T.: Ant Colony Optimization. MIT Press (2004)

[18] Eyckelhof, C.J., Snoek, M.: Ant Systems for a Dynamic TSP. In: Proceedings of the Third International Workshop on Ant Algorithms, ANTS 2002, pp. 88–99. Springer, London (2002)

[19] Fernandes, C.M., Lima, C., Rosa, A.C.: UMDAs for dynamic optimization problems. In: Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO 2008, pp. 399–406. ACM, New York (2008)

[20] Fernandes, C.M., Rosa, A.C., Ramos, V.: Binary ant algorithm. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 41–48. ACM, New York (2007)

[21] Gambardella, L.M., Taillard, E.D., Dorigo, M.: Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society 50(2), 167–176 (1999)

[22] Goss, S., Aron, S., Deneubourg, J.L., Pasteels, J.M.: Self-organized shortcuts in the argentine ant. Naturwissenschaften 76, 579–581 (1989)

[23] Grassé, P.P.: La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. la theéorie de la stigmergie: Essai dínterprétation du comportement des termites constructeurs. Insectes Sociaux 6(1), 41–48 (1959)

[24] Guntsch, M., Middendorf, M.: Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)

[25] Guntsch, M., Middendorf, M.: Applying population based aco to dynamic optimization problems. In: Proceedings of the Third International Workshop on Ant Algorithms, ANTS 2002, pp. 111–122. Springer (2002)

[26] Guntsch, M., Middendorf, M.: A Population Based Approach for ACO. In: Cagnoni, S., Gottlieb, J., Hart, E., Middendorf, M., Raidl, G.R. (eds.) EvoIASP 2002, EvoWorkshops 2002, EvoSTIM 2002, EvoCOP 2002, and EvoPlan 2002. LNCS, vol. 2279, pp. 72–81. Springer, Heidelberg (2002)

[27] Guntsch, M., Middendorf, M., Schmeck, H.: An Ant Colony Optimization Approach to Dynamic TSP. In: Spector, L., Goodman, E.D., Wu, A., Langdon, W.B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M.H., Burke, E. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2001, pp. 860–867. Morgan Kaufmann, San Francisco (2001)

[28] Hendtlass, T., Moser, I., Randall, M.: Dynamic Problems and Nature Inspired Meta-Heuristics. In: Proceedings of the Second IEEE International Conference on e-Science and Grid Computing, E-SCIENCE 2006, pp. 111–116. IEEE Computer Society, Washington, DC (2006)

[29] Ho, C.K., Ewe, H.T.: Ant Colony Optimization Approaches for the Dynamic Load-Balanced Clustering Problem in Ad Hoc Networks. In: Swarm Intelligence Symposium, SIS 2007, pp. 76–83. IEEE (April 2007)

[30] Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - A survey. IEEE Transactions on Evolutionary Computation 9(3), 303–318 (2005)

[31] Korošec, P., Šilc, J.: The differential ant-stigmergy algorithm applied to dynamic optimization problems. In: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation, CEC 2009, pp. 407–414. IEEE Press, Piscataway (2009)

[32] Korošec, P., Šilc, J., Oblak, K., Kosel, F.: The differential ant-stigmergy algorithm: an experimental evaluation and a real-world application. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 157–164 (September 2007)

[33] Leguizamón, G., Ordóñez, G., Molina, S., Alba, E.: Canonical Metaheuristics for Dynamic Optimization Problems. In: Alba, E., Blum, C., Isasi, P., León, C., Gómez, J.A. (eds.) Optimization Techniques for Solving Complex Problems, pp. 83–100. John Wiley & Sons, Inc. (2008)

[34] Maniezzo, V.: Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem. Informs Journal on Computing 11(4), 358–369 (1999)

[35] Monmarché, N., Guinand, F., Siarry, P.: Artificial Ants. Wiley-ISTE (2010)

[36] Montemanni, R., Gambardella, L.M., Rizzoli, A.E., Donati, A.V.: A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System. In: Second International Workshop on Freight Transportation and Logistics, pp. 27–30 (2003)

[37] Morrison, R.W.: Designing Evolutionary Algorithms for Dynamic Environments. Natural Computing Series. Springer (2004)

[38] Mühlenbein, H., Paass, G.: From Recombination of Genes to the Estimation of Distributions I. Binary Parameters. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996)

[39] Pigné, Y., Guinand, F.: Short and Robust Communication Paths in Dynamic Wireless Networks. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 520–527. Springer, Heidelberg (2010)

[40] Psaraftis, H.N.: Dynamic vehicle routing: Status and prospect. Annals Operations Research 61, 143–164 (1995)

[41] Reimann, M., Doerner, K., Hartl, R.F.: D-ants: Savings based ants divide and conquer the vehicle routing problem. Computers & Operations Research 31(4), 563–591 (2004)

[42] Schoonderwoerd, R., Holland, O.E., Bruten, J.L., Rothkrantz, L.J.M.: Ant-based load balancing in telecommunications networks. Adaptive Behavior 2, 169–207 (1996)

[43] Socha, K., Dorigo, M.: Ant colony optimization for continuous domains. European Journal of Operational Research 185(3), 1155–1173 (2008)

[44] Stützle, T., Dorigo, M.: ACO algorithms for the quadratic assignment problem. In: Corne, D., Dorigo, M., Glover, F. (eds.) New Ideas in Optimization, pp. 33–50. McGraw Hill, London (1999)

[45] Xiao, J., Li, J., Xu, Q., Huang, W., Lou, H.: ACS-based Dynamic Optimization for Curing of Polymeric Coating. The American Institute of Chemical Engineers (AIChE) Journal 52(2), 1410–1422 (2005)