The Hyper-Cube Framework for Ant Colony Optimization

Christian Blum, *Student Member, IEEE,* and Marco Dorigo, *Senior Member, IEEE*

***Abstract*—Ant colony optimization is a metaheuristic approach belonging to the class of model-based search algorithms. In this paper, we propose a new framework for implementing ant colony optimization algorithms called the hyper-cube framework for ant colony optimization. In contrast to the usual way of implementing ant colony optimization algorithms, this framework limits the pheromone values to the interval [0,1]. This is obtained by introducing changes in the pheromone value update rule. These changes can in general be applied to any pheromone value update rule used in ant colony optimization. We discuss the benefits coming with this new framework. The benefits are twofold. On the theoretical side, the new framework allows us to prove that in Ant System, the ancestor of all ant colony optimization algorithms, the average quality of the solutions produced increases in expectation over time when applied to unconstrained problems. On the practical side, the new framework automatically handles the scaling of the objective function values. We experimentally show that this leads on average to a more robust behavior of ant colony optimization algorithms.**

***Index Terms*—Ant colony optimization (ACO), metaheuristics.**

I. Introduction

ANT COLONY optimization (ACO) is a metaheuristic for hard discrete optimization problems that was first proposed in the early 1990s [1]–[3]. The inspiring source of ACO is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates quantity and quality of the food and carries some of the found food to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. The indirect communication between the ants via the pheromone trails allows them to find shortest paths between their nest and food sources. This functionality of real ant colonies is exploited in artificial ant colonies in order to solve discrete optimization problems.

From a broader perspective, ACO algorithms belong to the class of model-based search (MBS) algorithms [4], [5]. MBS algorithms are increasingly popular methods for solving discrete optimization problems. An MBS algorithm is characterized by the use of a (parametrized) probabilistic model $M \in \mathcal{M}$ (where $\mathcal{M}$ is the set of all possible probabilistic models) that is used to generate solutions to the problem under consideration. The class of MBS algorithms can be divided into two subclasses with respect to the way the probabilistic model is used. The algorithms in the first subclass use a given probabilistic model without changing the model structure during run-time, whereas the algorithms of the second subclass use and change the probabilistic model in alternating phases. ACO algorithms are examples of algorithms from the first subclass that use parametrized probabilistic models without changing them. In ACO algorithms, the probabilistic model is called the *pheromone model*. The pheromone model consists of a set of model parameters $\mathcal{T}$, that are called the *pheromone trail parameters*. The pheromone trail parameters $\mathcal{T}_i \in \mathcal{T}$ have values τ_i, called *pheromone values*. At run-time, ACO algorithms try to update the pheromone values in such a way that the probability to generate high-quality solutions increases over time. The pheromone values are updated using previously generated solutions. The update aims to concentrate the search in regions of the search space containing high-quality solutions. In particular, the reinforcement of solution components depending on the solution quality is an important ingredient of ACO algorithms. It implicitly assumes that good solutions consist of good solution components. To learn which components contribute to good solutions can help to assemble them into better solutions. In general, the ACO approach attempts to solve an optimization problem by repeating the following two steps.

1) Candidate solutions are constructed using a pheromone model; that is, a parametrized probability distribution over the solution space.
2) The candidate solutions are used to modify the pheromone values in a way that is deemed to bias future sampling toward high-quality solutions.

In this paper, we introduce a new framework for implementing ACO algorithms. We call this framework the hyper-cube framework (HCF) for ACO.[1] The framework is based on changing the pheromone update rules used in ACO algorithms so that the range of values the pheromone trail parameters can assume is limited to the interval [0,1]. The main motivation that led us to introduce the HCF is the observation that in standard ACO algorithms the pheromone values, and therefore the performance of the algorithms, strongly depend on the scale of the problem. In other words, a standard ACO

Manuscript received March 24, 2003; revised August 2, 2003, 2003. The work of M. Dorigo was supported by the Belgian FNRS. This work was supported by the "Metaheuristics Network," a Marie Curie Research Training Network funded by the Improving Human Potential program of the CEC, under Grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication. This paper was recommended by Associate Editor F. Gomide.

The authors are with IRIDIA, Université Libre de Bruxelles, Brussels, Belgium (e-mail: cblum@ulb.ac.be; mdorigo@ulb.ac.be).

Digital Object Identifier 10.1109/TSMCB.2003.821450

[1] A preliminary version of the HCF for ant colony optimization was introduced in [6].

algorithm can give different results, and more generally, have a different behavior, when applied to two isomorphic problems differing only in that one is obtained from the other by multiplying the objective function by a constant value. This is an undesirable property, which is removed by implementing ACO algorithms in the HCF. Additionally, the HCF makes some aspects of the theoretical analysis of ACO algorithms easier, and, on the practical side, makes ACO algorithms more robust. The new update rules also allow an insightful graphical interpretation. All these aspects will be discussed in the rest of the paper, whose organization is as follows.

In Section II, we outline the Ant System (AS), which was the first ACO algorithm to be proposed. In that context, we give a precise definition of the solution components and of the construction graph in ACO. In Section III, we propose the HCF for ACO algorithms and we give its graphical interpretation. In Section IV, we present some theoretical results obtained using the HCF, while we focus on the practical benefits of the HCF in Section V. Section VI presents some computational results in which we show that the proposed algorithm, although not optimized for the considered problem, has a rather good performance. Section VII offers a summary and outlines future work.

II. Ant System and Combinatorial Optimization

Informally, AS [1], [3] works by letting a set of artificial ants probabilistically and incrementally build solutions to the combinatorial optimization problem under consideration. At each step of the construction process, ants make probabilistic decisions biased by the pheromone values. The issue that we discuss in this section is how to map the considered problem to a representation that can be used by the artificial ants to build solutions.

A. Solution Components and Construction Graph

In the following, we give a formal characterization, as given for example in [7], of the representation the artificial ants use.

Let us consider the minimization[2] problem $\mathcal{P} = (\mathcal{S}, f, \Omega)$, where $\mathcal{S}$ is the *set of candidate solutions*, f is the *objective function* which assigns to each candidate solution $s \in \mathcal{S}$ an objective function (cost) value $f(s)$, and Ω is a *set of constraints*. The goal is to find a *globally optimal* feasible solution s^*; that is, a minimum cost feasible solution to the minimization problem.

The problem $(\mathcal{S}, f, \Omega)$ is mapped on a problem that can be characterized by the following list of items.

- A finite set $\mathcal{C} = \{c_1, c_2, \ldots, c_{|\mathcal{C}|}\}$ of opportunely defined *solution components*.
- The *states* of the problem, defined in terms of sequences $x = \langle c_i, c_j, \ldots, c_h, \ldots \rangle$ of finite length over the elements of $\mathcal{C}$. The set of all possible states is denoted by $\mathcal{X}$. The length of a sequence x, that is, the number of components in the sequence, is expressed by $|x|$. The maximum length of a sequence is bounded by a positive constant $n < +\infty$.
- The set of (candidate) solutions $\mathcal{S}$, with $\mathcal{S} \subseteq \mathcal{X}$.

Given this formulation, ants build solutions by performing randomized walks on the completely connected graph $G_C = (\mathcal{C}, \mathcal{L})$ whose vertexes are the components $\mathcal{C}$ and the set $\mathcal{L}$ fully connects the components $\mathcal{C}$. The graph G_C is called *construction graph* and elements of $\mathcal{L}$ are called *connections*. The problem constraints Ω are implemented in the policy followed by the ants. In most applications, ants construct feasible solutions. However, sometimes it may be necessary or beneficial to let them construct also infeasible solutions. Components $c_i \in \mathcal{C}$ and connections $l_{ij} \in \mathcal{L}$ can have associated a *pheromone trail parameter*: $\mathcal{T}_i$, if associated to components, and $\mathcal{T}_{ij}$, if associated to connections. The values of the pheromone trail parameters—called pheromone values—are denoted by τ_i, respectively, τ_{ij}. The pheromone trail parameters encode a long-term memory about the search process that is updated by the ants themselves. The pheromone values are used by the ants' heuristic rule to make probabilistic decisions on how to move on the graph.

This general description, which leaves a large amount of freedom to the algorithm designer in the definition of the solution components and of the construction graph, was motivated (in [7]) by the desire to include in the ACO metaheuristic both algorithms designed for NP-hard problems and algorithms for network routing problems. However, when only NP-hard problems are considered, one can give a more precise definition of the solution components and of the construction graph. To do so, we first give a formal definition of a combinatorial optimization problem.

Definition 1: A **Combinatorial Optimization** problem $\mathcal{P} = (\mathcal{S}, f, \Omega)$ is defined by:

- a set of discrete variables X_i with values $x_i \in D_i = \{d_1^i, \ldots, d_{|D_i|}^i\}$, $i = 1, \ldots, n$;
- a set Ω of constraints among variables;
- an *objective function* $f : D_1 \times \ldots \times D_n \rightarrow \mathbb{R}$ to be minimized.

The set of all possible feasible assignments is

$$\mathcal{S} = \{s = \{(X_1, x_1), \ldots, (X_n, x_n)\} \,|x_i \in D_i, \; s \text{ satisfies all the constraints}\}.$$

The set $\mathcal{S}$ is usually called a *search (or solution) space*, as each element of the set can be seen as a candidate solution. A solution $s^* \in \mathcal{S}$ is called a globally optimal solution, if $f(s^*) \leq f(s)\, \forall s \in S$. The set of all globally optimal solutions is denoted by $\mathcal{S}^* \subseteq \mathcal{S}$. To solve a combinatorial optimization problem one has to find a solution $s^* \in \mathcal{S}^*$.

Given the above definition, we call the combination of a decision variable X_i and one of its domain values $x_i \in D_i$ a *solution component*, denoted by c_{i,x_i}. We denote a partial solution by s^p (corresponding to a state of the problem) and the fact that a solution component c_{i,x_i} is part of the partial solution s^p by $c_{i,x_i} \in s^p$. The construction graph is the completely connected graph of all the solution components.

In contrast to the more general description of the solution components as given at the beginning of this subsection (and in [7]), pheromone trail parameters $\mathcal{T}_{i,x_i}$ can only be assigned to solution components $c_{i,x_i} \in \mathcal{C}$. The value of a pheromone trail parameter $\mathcal{T}_{i,x_i}$—called pheromone value—is denoted by τ_{i,x_i}. The set of all pheromone trail parameters is denoted by $\mathcal{T}$.

[2]Note that minimizing over an objective function f is the same as maximizing over $-f$. Therefore, every combinatorial optimization problem can be described as a minimization problem.

B. AS

The pseudocode for the AS algorithm is shown in Algorithm 1. In this algorithm, s_j, $j = 1, \ldots, n_a$, is the solution constructed by ant j, and n_a is the number of ants. After the initialization of the pheromone values, at every step of the algorithm each ant constructs a solution. These solutions are then used to update the pheromone values. The components of the AS algorithm are explained in more detail in the following.

Algorithm 1 Pseudocode for Ant System
InitializePheromoneValues($\mathcal{T}$)
while stop conditions not satisfied **do**
  for $j = 1, \ldots, n_a$ **do**
    $s_j \leftarrow$ConstructSolution($\mathcal{T}$)
  end for
  ApplyPheromoneUpdate ($\mathcal{T}$, $s_1, \ldots, s_{n_a}$)
end while

InitializePheromoneValues($\mathcal{T}$): At the beginning of the algorithm all the pheromone values are initialized to the numerical value $c > 0$. This value is small (i.e., in [0,1]) as in most ACO algorithms.[3]

ConstructSolution($\mathcal{T}$): In the construction phase an ant incrementally constructs a solution by adding solution components to the partial solution s^p constructed so far. The probabilistic choice of the next solution component to be added is done by using the *transition probabilities*

$$\mathbf{p}\left(c_{i,x_i} | s^p\right), \quad \forall c_{i,x_i} \in J(s^p) \tag{1}$$

where $J(s^p)$ denotes the set of feasible solution components that an ant is allowed to add to the current partial solution s^p (such that the resulting partial, or complete, solution is feasible). The transition probabilities are determined by a function that is called the *state transition rule*, which is a function of the pheromone values and possibly of other information. The state transition rule used by AS determines the probability of adding to the current partial solution s^p a feasible solution component $c_{i,x_i} \in J(s^p)$ as the ratio between c_{i,x_i}'s pheromone value τ_{i,x_i} and the sum of the pheromone values on all feasible solution components.

ApplyPheromoneUpdate ($\mathcal{T}$, $s_1, \ldots, s_{n_a}$): Once all ants have constructed a solution, the following rule for updating the pheromone values is applied:

$$\tau_{i,x_i} \leftarrow (1 - \rho) \cdot \tau_{i,x_i} + \sum_{\left\{s \in \mathcal{S}_{\text{upd}} | c_{i,x_i} \in s\right\}} F(s) \tag{2}$$

for $i = 1, \ldots, n$, $x_i \in \{d_1^i, \ldots, d_{|D_i|}^i\}$. $\mathcal{S}_{\text{upd}}$ is the set of solutions that were generated in the current iteration, $\rho \in (0, 1]$ is a parameter called evaporation rate, and $F : \mathcal{S} \mapsto \mathbb{R}^+$ is a function such that $f(s) < f(s') \Rightarrow F(s) \geq F(s')$, $\forall s, s' \in \mathcal{S}$, $s \neq s'$. $F(\cdot)$ is commonly called the *quality function*. The goal of this update rule is to increase the pheromone values of solution components that have been found in high-quality solutions.

[3]A notable exception is $\mathcal{MAX}-\mathcal{MIN}$ Ant System [8], in which c is usually set to a quite large number.

Although the AS algorithm provides the basic underlying mechanism of ACO algorithms to solve combinatorial optimization problems, in the form described above it is not very effective. Therefore, in the last couple of years a number of changes and extensions have been proposed that make ACO algorithms very effective, often reaching state-of-the-art performance (see [9] and [10] for overviews). The improvements over AS include ACO algorithms such as the Ant Colony System [11], and the $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [8]. Important features of these algorithms are elitist strategies, diversification and intensification mechanisms and, in the case of $\mathcal{MM}$AS, lower and upper bounds to the pheromone values. Also, the application of local search to the solutions constructed by the ants proved to be very effective. The interested reader will find a general description of ACO algorithms in [7] and [10].

C. Examples

In this subsection, we give two examples of how the introduced formalization can be used to describe ACO algorithms.

ACO for the ATSP: The asymmetric traveling salesman problem (ATSP) [12] is a NP-hard combinatorial optimization problem where the goal is to find the shortest tour (a Hamiltonian cycle) in a completely connected, directed graph $G = (V, E)$ with edge weights. The ATSP can be modeled as follows. To every node $i \in V$, we assign a decision variable X_i. The domain D_i for a decision variable X_i consists of $n - 1$ values x_{ij}, $j = 1, \ldots, n$ and $j \neq i$, where the value x_{ij} corresponds to the edge from node i to node j. A point in the domain space corresponds to a feasible solution to the ATSP if the edges corresponding to the values of the decision variables constitute a Hamiltonian cycle in G. The objective function $f(\cdot)$ gives, for each feasible solution, the sum of the edge-weights of the edges in the corresponding Hamiltonian cycle. The ATSP is a constrained problem, as not all points in the domain space correspond to feasible solutions. As outlined above, to every combination of decision variable X_i and domain value x_{ij}, we associate a model parameter $\mathcal{T}_{i,x_{ij}}$.

The construction of solutions works as follows. One of the decision variables, say X_i, is chosen uniformly at random. Then $n - 1$ construction steps are performed. In the following, s_k^p denotes a partial solution where the lower index denotes the decision variable which has to be dealt with in the current step (i.e., if in the previous construction step a decision variable X_l was assigned a value x_{lk}, then in the current construction step we have to assign a value to decision variable X_k). The transition probabilities in each of these $n - 1$ construction steps are the following:

$$\mathbf{p}\left(c_{k,x_{kj}} | s_k^p\right) = \begin{cases} \dfrac{\tau_{k,x_{kj}}}{\sum_{c_{k,x_{kl}} \in J\left(s_k^p\right)} \tau_{k,x_{kl}}}, & \text{if } c_{k,x_{kj}} \in J\left(s_k^p\right) \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

The last construction step consists in assigning the value $x_{i'i}$ to the last untreated decision variable $X_{i'}$. This description of ACO for the ATSP is equivalent to the one given in [3].

ACO for subset problems: In subset problems such as the knapsack problem (KP) [13] or the k-cardinality tree problem (KCT) [14], we are given a set of items I. From this set of

items a subset must be selected that satisfies the constraints and gives the optimal objective function value. Therefore, this kind of problem can be modeled as follows: To each item $i \in I$ we assign a binary decision variable X_i. Then, for every variable X_i, we have two solution components, $c_{i,1}$ corresponding to $X_i = 1$ and $c_{i,0}$ corresponding to $X_i = 0$. In order to construct a solution, an ant incrementally adds items to the partial solution it is constructing, using the following transition probabilities:

$$\mathbf{p}(c_{i,1}|s^p) = \begin{cases} \frac{\tau_{i,1}}{\sum_{c_{j,1} \in J(s^p)} \tau_{j,1}}, & \text{if } c_{i,1} \in J(s^p) \\ 0, & \text{otherwise.} \end{cases} \quad (4)$$

The set $J(s^p)$ consists of all components $c_{i,1}$ that correspond to items i that are not selected yet, and that can be added to the current partial solution s^p without violating feasibility. For an example of an ACO algorithm for subset problems, see [15].

III. The HCF

In the AS algorithm described in the previous section, we can regard the set of pheromone values as a $|\mathcal{C}|$-dimensional vector $\vec{\tau}$. By applying the pheromone update at the end of every iteration, this vector is changed. It moves in a $|\mathcal{C}|$-dimensional hyperspace defined by the lower and upper limits of the range of values that the pheromone trail parameters can assume. We will denote this hyperspace in the following by $\mathcal{H}_{\mathcal{T}}$. For the AS algorithm, these limits are given as follows.

Proposition 1: For any pheromone value τ_{i,x_i}, the following holds:

$$\lim_{t \to \infty} \tau_{i,x_i}(t) \leq \frac{1}{\rho} \cdot n_a \cdot F(s^*) \quad (5)$$

where $s^* \in \mathcal{S}^*$. By $\tau_{i,x_i}(t)$, we denote the pheromone values at iteration t.

Proof: The maximum possible increase of a pheromone value τ_{i,x_i} with $c_{i,x_i} \in s^*$ in any iteration is n_a times $F(s^*)$ if all the n_a ants produce the same optimal solution s^*. Therefore, due to evaporation, the pheromone value τ_{i,x_i} at iteration t is bounded by

$$\tau_{i,x_i}^{\max}(t) = (1-\rho)^t \cdot c + \sum_{j=1}^{t} (1-\rho)^{t-j} \cdot n_a \cdot F(s^*) \quad (6)$$

where c is the initial value for all the pheromone trail parameters. Asymptotically, because $0 < \rho \leq 1$, this sum converges to $(1/\rho) \cdot n_a \cdot F(s^*)$. ∎

The dependency of the upper limit for the pheromone values on the quality function $F(\cdot)$ implies that the limits of $\mathcal{H}_{\mathcal{T}}$ can be very different depending on the quality function and therefore depending on the problem instance tackled. In contrast, the pheromone update rule of the HCF as described in the following implicitly defines the hyperspace $\mathcal{H}_{\mathcal{T}}$ independently of the quality function $F(\cdot)$ and of the problem instance tackled.

The pheromone update rule of the AS algorithm in the HCF is the following:

$$\tau_{i,x_i} \leftarrow (1-\rho) \cdot \tau_{i,x_i} + \rho \cdot \sum_{\{s \in \mathcal{S}_{\text{upd}} | c_{i,x_i} \in s\}} \frac{F(s)}{\sum_{\{s' \in \mathcal{S}_{\text{upd}}\}} F(s')} \quad (7)$$

for $i = 1, \ldots, n$, $x_i \in \{d_1^i, \ldots, d_{|D_i|}^i\}$. The difference between this pheromone update rule and the one that is used in the standard AS algorithm (2) consists of the multiplication of the amount of added pheromone by ρ and the normalization of this added amount of pheromone.

In order to give a graphical interpretation of the AS update rule in the HCF, we consider a solution s to the problem from a different point of view. With respect to a solution $s \in \mathcal{S}$, we partition the set of solution components $\mathcal{C}$ into two subsets, the set $\mathcal{C}_{\text{in}}$ that contains all solution components $c_{i,x_i} \in s$, and $\mathcal{C}_{\text{out}} = \mathcal{C} \setminus \mathcal{C}_{\text{in}}$. In this way, we can associate to a solution s a binary vector $\vec{s}$ of dimension $|\mathcal{C}|$, where the position corresponding to solution component c_{i,x_i} is set to 1 if $c_{i,x_i} \in \mathcal{C}_{\text{in}}$, to 0 otherwise. This means that we can regard a solution s as a corner of the $|\mathcal{C}|$-dimensional hypercube. Therefore, the set of feasible solutions $\mathcal{S}$ can be regarded as a (sub)set of the corners of the $|\mathcal{C}|$-dimensional hypercube. In the following, we denote the convex hull of $\mathcal{S}$ by $\tilde{\mathcal{S}}$. It holds that

$$\vec{v} \in \tilde{\mathcal{S}} \Leftrightarrow \vec{v} = \sum_{s \in \mathcal{S}} \alpha_s \vec{s}, \quad \alpha_s \in [0,1], \quad \sum_{s \in \mathcal{S}} \alpha_s = 1. \quad (8)$$

For an example, see Fig. 1(a). In the following, we give a graphical interpretation of the pheromone update rule in the HCF. When written in vector form, (7) can be expressed as

$$\vec{\tau} \leftarrow (1-\rho) \cdot \vec{\tau} + \rho \cdot \vec{m} \quad (9)$$

where $\vec{m}$ is a $|\mathcal{C}|$-dimensional vector with

$$\vec{m} = \sum_{s \in \mathcal{S}_{\text{upd}}} \gamma_s \cdot \vec{s} \quad (10)$$

and

$$\gamma_s = \frac{F(s)}{\sum_{s' \in \mathcal{S}_{\text{upd}}} F(s')}. \quad (11)$$

Vector $\vec{m}$ is a vector in $\tilde{\mathcal{S}}$, the convex hull of $\mathcal{S}$, as $\sum_{s \in \mathcal{S}_{\text{upd}}} \gamma_s = 1$ and $0 \leq \gamma_s \leq 1 \; \forall s \in \mathcal{S}_{\text{upd}}$. It also holds that vector $\vec{m}$ is the weighted average of binary solution vectors. The higher the quality $F(s)$ of a solution s, the higher its influence on vector $\vec{m}$. Simple algebra allows us to express (9) as

$$\vec{\tau} \leftarrow \vec{\tau} + \rho \cdot (\vec{m} - \vec{\tau}). \quad (12)$$

This shows that the application of the AS pheromone update rule in the HCF determines a shift of the current pheromone value vector $\vec{\tau}$ toward $\vec{m}$ [see Fig. 1(b)]. The size of this shift is determined by the parameter ρ. In the extreme cases, there is either very little update (when ρ is very close to zero), or the current pheromone value vector $\vec{\tau}$ is replaced by $\vec{m}$ (when $\rho = 1$). Furthermore, if the initial pheromone value vector $\vec{\tau}$ is in $\tilde{\mathcal{S}}$, it remains in $\tilde{\mathcal{S}}$, and the pheromone values are bounded to the interval [0,1]. This means that the HCF, independently of the problem instance tackled, defines the hyperspace for the pheromone values as the $|\mathcal{C}|$-dimensional hypercube.

In order to emphasize that the HCF is not restricted to the AS algorithm, we show that the pheromone update rule in the HCF can be easily applied to two of the best-known and experimentally most successful versions of the ACO

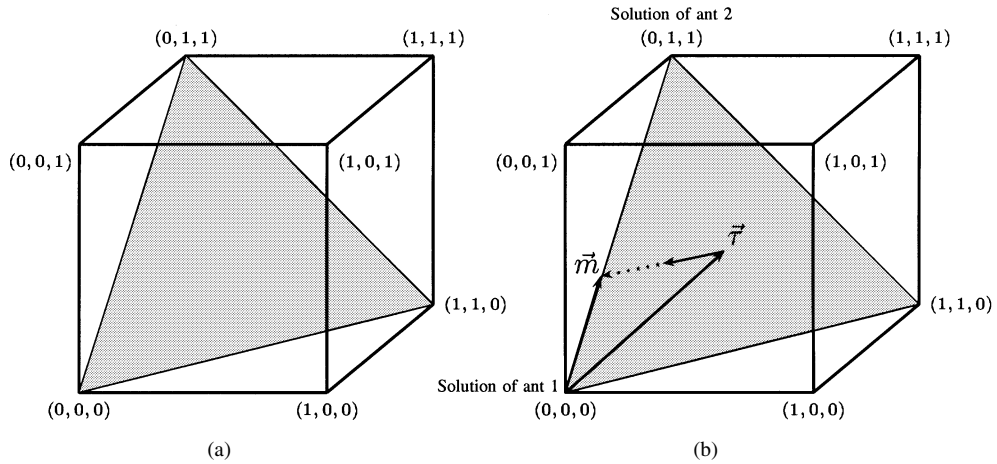

Fig. 1. Set $\mathcal{S}$ of feasible solutions consists of the three vectors (0,0,0), (1,1,0), and (0,1,1). The gray shaded area depicts the set $\bar{\mathcal{S}}$. In (b), two solutions have been generated by two ants. The vector $\vec{m}$ is the weighted average of these two solutions, (0,0,0) is of higher quality, and $\vec{\tau}$ will be shifted toward $\vec{m}$ as a result of the pheromone value update rule (7). (a) Example for the convex hull of binary solution vectors. (b) Example for the pheromone update in the HCF.

metaheuristic: Ant Colony System (ACS) and $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS). These algorithms are characterized by the use of only one solution, denoted in the following by s_{upd}, for every pheromone update. From (9) to (12), it follows that the pheromone update rule for both ACS and $\mathcal{MM}$AS in the HCF is

$$\vec{\tau} \leftarrow \vec{\tau} + \rho \cdot (\vec{s}_{\text{upd}} - \vec{\tau}). \tag{13}$$

So, the old vector $\vec{\tau}$ is shifted toward the updating solution s_{upd} in vector form.

IV. Theoretical Results

The desired behavior of a MBS algorithm for optimization can be stated as follows. The average quality of the generated solutions should grow from iteration to iteration. This is desirable, as it usually increases the probability of improving the best solution found over time. In the following, we show that the AS algorithm in the HCF shows this behavior. In order to do so, we analyze the expected behavior of the algorithm. This can be done by changing the pheromone values at each iteration according to the expected update under the assumption that the number of solutions generated per iteration is infinite. This way of examining the expected behavior of an algorithm is known from the field of evolutionary computation (see, for example, the *zeroth-order model* proposed in [16]).

We measure the expected quality $W_F(\mathcal{T})$ of the solutions generated per iteration as follows:

$$W_F(\mathcal{T}) = \sum_{s \in \mathcal{S}} F(s) \cdot \mathbf{p}(s|\mathcal{T}) \tag{14}$$

where $\mathbf{p}(s|\mathcal{T})$ is the probability that solution s is generated given the current pheromone values. We examine the AS algorithm in the HCF on unconstrained problems. By unconstrained problems, we mean that in a solution a decision variable can take any value of its domain, independently of the values of the other decision variables. For the selected type of problem, the construction of a solution is done by sampling a value for every decision variable according to the following probabilities:

$$\mathbf{p}(X_i = x_i|\mathcal{T}) = \frac{\tau_{i,x_i}}{\sum_{j=1}^{|D_i|} \tau_{i,d_j^i}}, \quad x_i \in D_i, \quad i = 1, \ldots, n. \tag{15}$$

As we assume an infinite number of solutions to be generated at each iteration, the pheromone value update in the HCF can be stated as follows:

$$\tau_{i,x_i}(t+1) \leftarrow (1-\rho) \cdot \tau_{i,x_i}(t) + \rho \cdot \sum_{s \in \mathcal{S}|_{X_i = x_i}} \frac{F(s) \cdot \mathbf{p}(s|\mathcal{T})}{W_F(\mathcal{T})} \tag{16}$$

for $i = 1, \ldots, n$, $x_i \in D_i$, where t is the iteration counter.

Theorem 1: The AS algorithm in the HCF applied to unconstrained problems has the property that pheromone values can be interpreted as probabilities throughout the run of the algorithm, provided that the initial setting is such that they are probabilities.

Proof: Assume that $\sum_{x_i \in D_i} \tau_{i,x_i}(t) = 1$, $i = 1, \ldots, n$ at iteration t. Then, it holds that

$$\mathbf{p}(X_i = x_i|t) = \tau_{i,x_i}(t), \quad \forall x_i \in D_i,\ i = 1, \ldots, n. \tag{17}$$

Then, for $i = 1, \ldots, n$ it holds that $\sum_{x_i \in D_i} \tau_{i,x_i}(t+1) =$

$$\sum_{x_i \in D_i} \left((1-\rho)\tau_{i,x_i}(t) + \rho \cdot \sum_{s \in \mathcal{S}|_{X_i = x_i}} \frac{F(s) \cdot \mathbf{p}(s|\mathcal{T})}{W_F(\mathcal{T})} \right)$$
$$= (1-\rho) \left(\sum_{x_i \in D_i} \tau_{i,x_i}(t) \right) + \rho \cdot \frac{W_F(\mathcal{T})}{W_F(\mathcal{T})} = 1$$

and, therefore, $\mathbf{p}(X_i = x_i|t+1) = \tau_{i,x_i}(t+1)$ for all $x_i \in D_i$, and $i = 1, \ldots, n$. ■

In order to proceed, we also need the following result obtained by Baum and Sell in 1968 [17].

Theorem 2: Let W be a polynomial in the variables X_{ij} where $i = 1, \ldots, n$ and $j = 1, \ldots, m_i$, $m_i \geq 1$, with nonneg-

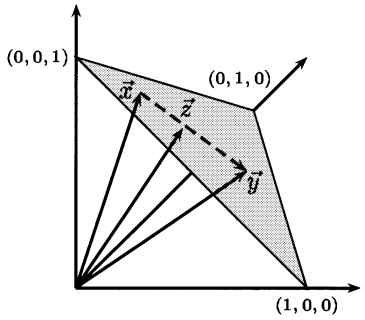

Fig. 2. Example for Theorem 2, where $n = 1$ and $m_1 = 3$. Therefore, given are the three variables X_{11}, X_{12}, and X_{13}. Furthermore, given is a polynomial $W : D \cup \partial D \mapsto \mathbb{R}$, where a point $\vec{x} \in D \cup \partial D \subset \mathbb{R}^3$ must satisfy the following conditions: 1) $x_{1j} \geq 0$, for $j = 1, 2, 3$, and 2) $x_{11} + x_{12} + x_{13} = 1$. Therefore, $D \cup \partial D$ is the convex hull of the three vectors (1,0,0), (0,1,0), and (0,0,1). Given a point $\vec{x} \in D$, Theorem 2 claims that: 1) $\vec{y} = M(\vec{x})$ is a point in $D \cup \partial D$, where the mapping M is defined by (18); 2) $W(\vec{y}) > W(\vec{x})$, unless $\vec{x} = \vec{y}$; and 3) for all $\vec{z} = (1 - \beta)\vec{x} + \beta\vec{y}$, $\beta \in (0, 1]$, it holds that $W(\vec{z}) > W(\vec{x})$, unless $\vec{z} = \vec{y}$.

ative coefficients. The continuous values of these variables are denoted by x_{ij}. Let $D \cup \partial D$ denote the manifold with boundary given by: $x_{ij} \geq 0$, $\sum_{j=1}^{m_i} x_{ij} = 1$ for all $i = 1, \ldots, n$. Define a mapping $M : D \mapsto D \cup \partial D$ such that $\vec{y} = M(\vec{x})$ is obtained by

$$y_{ij} \leftarrow \frac{x_{ij} \frac{\partial W}{\partial X_{ij}}\Big|_{\vec{x}}}{\sum_{k=1}^{m_i} x_{ik} \frac{\partial W}{\partial X_{ik}}\Big|_{\vec{x}}}. \tag{18}$$

Then, for $\vec{y}$, it holds that $y_{ij} \geq 0$, $\sum_{j=1}^{m_i} y_{ij} = 1$ for all $i = 1, \ldots, n$. Furthermore

$$W\left(M(\vec{x})\right) > W(\vec{x}), \text{ unless } M(\vec{x}) = \vec{x}. \tag{19}$$

In other words, M is a growth transformation for the polynomial W. Inequality (19) also holds for all the points lying on the segment connecting $\vec{x}$ and $M(\vec{x})$. This means that for all $\vec{z} = (1 - \beta)\vec{x} + \beta M(\vec{x})$, $\beta \in (0, 1]$, it holds that

$$W(\vec{z}) > W(\vec{x}), \text{ unless } M(\vec{x}) = \vec{x}. \tag{20}$$

See Fig. 2 for an example. Using this result, we can prove the following theorem.

Theorem 3: For the expected quality of the solutions generated at each iteration by the AS algorithm in the HCF applied to unconstrained problems, it holds that

$$W_F(\mathcal{T}|t+1) > W_F(\mathcal{T}|t) \tag{21}$$

as long as at least one pheromone value changes from iteration t to iteration $t + 1$.

Proof: Because of Theorem 1, we can replace the function W_F on the pheromone trail parameter set $\mathcal{T}$ by a function $\tilde{W}_F$ on the parameter set $\mathcal{P} = \{\mathcal{P}_{i,x_i}\}$, $i = 1, \ldots, n$, $x_i \in D_i$, where the value of a parameter $\mathcal{P}_{i,x_i}$ is $\mathbf{p}(X_i = x_i|t)$. Then it holds that

$$W_F(\mathcal{T}) = \sum_{s \in \mathcal{S}} F(s)\mathbf{p}(s|\mathcal{T}) = \sum_{s \in \mathcal{S}} F(s)\mathbf{p}(s|\mathcal{P}) = \tilde{W}_F(\mathcal{P}) \tag{22}$$

where

$$\mathbf{p}(s|\mathcal{P}) = \prod_{j=1}^{n} \delta_{prob}(s, X_j) \tag{23}$$

with $\delta_{prob}(s, X_j) =$

$$\begin{cases} \mathbf{p}\left(X_j = d_1^j|t\right), & \text{if } s \text{ is characterized by } X_j = d_1^j \\ \quad\vdots & \quad\vdots \\ \mathbf{p}\left(X_j = d_{|D_j|}^j|t\right), & \text{if } s \text{ is characterized by} X_j = d_{|D_j|}^j. \end{cases}$$

Now we identify the function $\tilde{W}_F$ with the polynomial W of Theorem 2. In the following, we show that the update of the values of the parameters of function $\tilde{W}_F$ (that is the pheromone update) is equivalent to the mapping M defined by (18). First, we consider the case $\rho = 1$. From (16), the pheromone update expressed in terms of parameter set $\mathcal{P}$ reduces to

$$\mathbf{p}(X_i = x_i|t+1) \leftarrow \frac{\sum_{s \in \mathcal{S}|_{X_i = x_i}} F(s) \cdot \mathbf{p}(s|\mathcal{P})}{\tilde{W}_F(\mathcal{P})}. \tag{24}$$

Furthermore, it holds that

$$\frac{\partial \tilde{W}_F}{\partial \mathcal{P}_{i,x_i}}\Big|_{\mathbf{p}(X_i = x_i|t)} = \sum_{s \in \mathcal{S}|_{X_i = x_i}} F(s) \cdot \mathbf{p}(s|X_i = x_i, \mathcal{P}) \tag{25}$$

where $\mathbf{p}(s|X_i = x_i, \mathcal{P})$ is the probability to generate solution s under the assumption that $X_i = x_i$. This probability can be expressed as

$$\mathbf{p}(s|X_i = x_i, \mathcal{P}) = \prod_{j \neq i, j=1}^{n} \delta_{prob}(s, X_j). \tag{26}$$

Multiplying both sides of (25) by $\mathbf{p}(X_i = x_i|t)$, we obtain that

$$\mathbf{p}(X_i = x_i|t) \cdot \frac{\partial \tilde{W}_F}{\partial \mathcal{P}_{i,x_i}}\Big|_{\mathbf{p}(X_i = x_i|t)} = \sum_{s \in \mathcal{S}|_{X_i = x_i}} F(s)\mathbf{p}(s|\mathcal{P}) \tag{27}$$

which shows the equivalence of the numerators of the fractions on the right-hand side of (18) and (24). By summing both sides of (27) over all domain values (i.e., $x_i \in D_i$), we obtain that

$$\sum_{x_i \in D_i} \mathbf{p}(X_i = x_i|t) \cdot \frac{\partial \tilde{W}_F}{\partial \mathcal{P}_{i,x_i}}\Big|_{\mathbf{p}(X_i = x_i|t)} = \tilde{W}_F(\mathcal{P}|t) \tag{28}$$

which shows the equivalence of the denominators of the fractions on the right-hand side of (18) and (24). Using Theorem

2 we obtain that $\tilde{W}_F(\mathcal{P}|t+1) > \tilde{W}_F(\mathcal{P}|t)$ and therefore that $W_F(\mathcal{T}|t+1) > W_F(\mathcal{T}|t)$ for $\rho = 1$. The general case $\rho \in (0,1]$ follows from (20). ■

This result shows that the AS algorithm in the HCF is (in expectation) able to improve the average quality of the solutions generated per iteration by only (implicitly) using first order derivatives and without adjusting or computing optimal step sizes. This is in contrast to many conventional gradient methods [18], for which an increase in quality is guaranteed only when infinitesimal steps are taken, and determining the optimal step size entails computing higher order derivatives.[4] Moreover, this result shows a relation between ant colony optimization and other statistical learning procedures that can be considered as discrete dynamical systems [20]. Examples are statistical learning procedures for relaxation labeling networks [21], for speech recognition [22], evolutionary game theory and population dynamics [23], and the convergence theory of some algorithms from the field of evolutionary computation [24]. However, in general, this result cannot be transferred to ACO algorithms applied to constrained optimization problems. The reason for this is that in constrained problems the probability of choosing a domain value for a decision variable depends on the values of the other decision variables. Therefore, (25) does not hold and Theorem 3 cannot be proven anymore.

V. Practical Benefits

In addition to the theoretical properties discussed in the previous section, the implementation of ACO algorithms in the HCF brings also some practical benefits. First, the robustness of ACO algorithms increases due to the automatic scaling of the objective function values. This is shown in Section V-A via opportunely designed experiments. Second, the handling of the pheromone values is simplified as shown on the example of the implementation of a $\mathcal{MM}$AS in the HCF in Section V-B.

A. Automatic Scaling of Objective Function Values

When developing an algorithm to tackle NP-hard combinatorial optimization problems, one of the goals of the algorithm designer is to develop a robust algorithm whose behavior is independent of the range of objective function values that the particular problem instance tackled can assume. In other words, the behavior of an optimization algorithm should be roughly independent of whether a problem instance has objective function values ranging, for example, in the interval [0,1], or in the interval [0,1000]. In the following, we show that ACO algorithms implemented in the HCF do have this property, in contrast to standard ACO implementations. We compare the AS algorithm in its standard version with AS in the HCF by applying them to unconstrained binary problems. We generated a binary problem on 15 variables ($2^{15} = 32\,768$ solutions) assigning a random objective function value from the range [0,1] to every solution (applying a uniform sampling process). We denote this problem instance by instance1. From this instance we generated another problem instance, instance2, by multiplying every objective function value by 10^3, and another one, instance3, by multiplying every objective function value of instance1 by 10^6. Our goal is minimization. Therefore, we choose as quality function $F(s) = 1/f(s)$ for all solutions s. We applied each of the two algorithm versions, AS and AS in the HCF, 100 times with six different parameter settings (number of ants $n_a \in \{10, 50, 100\}$, $\rho \in \{0.1, 0.01\}$) to the three different problem instances. The termination criterion was 500 iterations for all runs. For both algorithm versions we initialized the pheromone values to 0.5. This is the natural setting for ACO algorithms implemented in the HCF, as a setting of 0.5 gives equal chance to both update directions. Fig. 3 shows the results in summarized form. Each of the four graphs has the following format: on the x axis are the instances, on the y axis the number of ants, and on the z axis is the percentage above the optimum of the value of the best solution found by the algorithm specified on the x axis and the y axis. The values on the z axis are averaged over 100 trials.

As the three problem instances are—except for the scale of the objective function values—the same, we would expect the performance of the algorithms to change only along the y axis (i.e, we expect to observe an increase in performance when the number of ants increases). Otherwise, for a fixed number of ants we would expect an algorithm to show the same average performance for all three problem instances. However, the graphs in Fig. 3 show that this is not the case, and that for a fixed number of ants, the performance of AS is different on the three problem instances. Fig. 3(a) and (c) show an improvement of AS along the x axis (changing range of objective function values) and the y axis (increasing number of ants). In general, the behavior of AS is worse for problem instance1, because many of the quality values $F(s)$ are very high in comparison to the initial pheromone values. This leads to a quite fast convergence of the AS algorithm when applied to instance1, as illustrated in Fig. 4. In contrast, the behavior of AS in the HCF, shown in Fig. 3(b) and (d), is as expected: for a fixed number of ants, the performance is the same on the three problem instances. This indicates that the behavior of ACO algorithms in the HCF is likely to be more robust than the behavior of standard ACO implementations. As the comparison of Fig. 3(a) and (b) with Fig. 3(c) and (d) shows, this conclusion seems to be independent of the setting of ρ. Of course, it would be possible to fine tune the initial pheromone values in a standard ACO implementation depending on the problem instance tackled. However, this is not necessary for ACO implementations in the HCF, as the natural initial value of 0.5 gives an equal chance to both update directions. In this sense, an ACO algorithm implemented in the HCF has one parameter fewer.

B. Practical Benefits of Implementing $\mathcal{MM}$AS in the HCF

In this subsection, we show that the HCF confinement of the pheromone values in the interval [0,1] helps in making decisions concerning the update of the pheromone values. For this purpose, we choose the $\mathcal{MAX}$ – $\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [8], which is one of the best-performing ACO

[4]However, the computation of higher order derivates can sometimes be replaced by line-search procedures [19] that do not require any derivative computation.

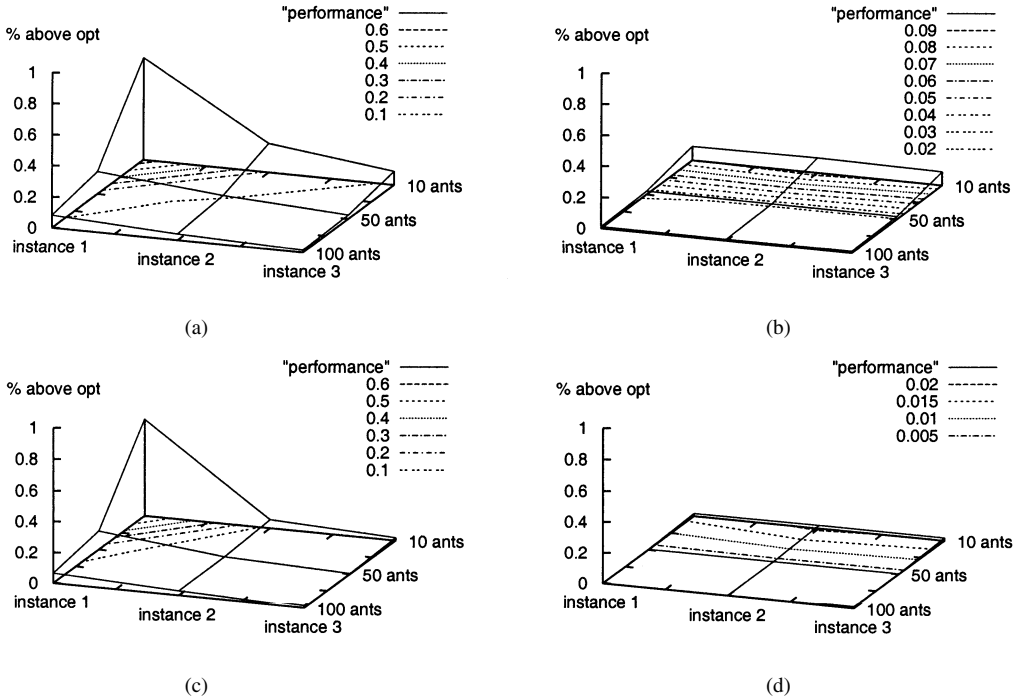

Fig. 3. The x axis (instances) and the y axis (number of ants) define different settings of the two algorithms, that are AS, (a) and (c), and AS in the HCF, (b) and (d). On the x axis are the instances, on the y axis the number of ants, and on the z axis is the percentage above optimum of the value of the best solution found by the algorithm that is specified by x axis and y axis. The values on the z axis are averaged over 100 trials. instance1 is a binary problem on 15 decision variables ($2^{15} = 32\,786$ solutions) with objective function values randomly chosen from [0,1]. instance2 is equivalent to instance1 with the objective function values multiplied by 10^3. The same holds for instance3, where the objective function values are multiplied by 10^6. (a) Results of AS for $\rho = 0.1$. (b) Results of AS in the HCF for $\rho = 0.1$. (c) and (d) Results of the two algorithms for $\rho = 0.01$. The legends of the four graphs indicate the "height" of the contour lines given on the xy-plane (i.e., they give a contour map of the performance surface). (a) AS, $\rho = 0.1$. (b) AS in the HCF, $\rho = 0.1$. (c) AS, $\rho = 0.01$. (d) AS in the HCF, $\rho = 0.01$.

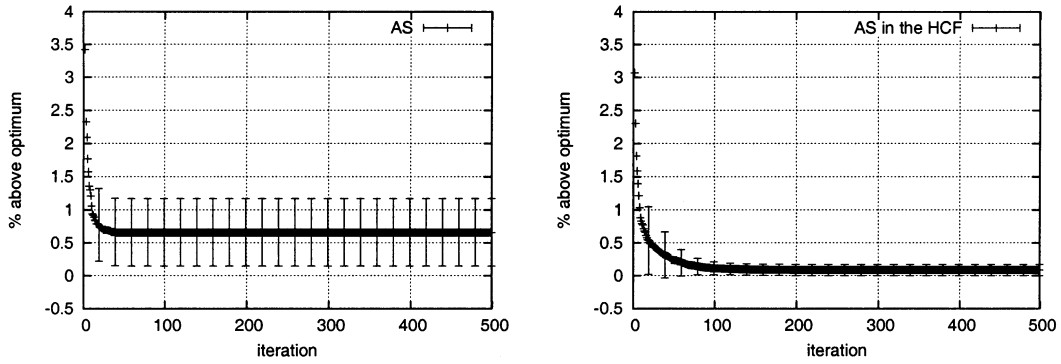

Fig. 4. Performance of (a) AS and (b) AS in the HCF applied to instance1, with settings $n_a = 10$ and $\rho = 0.01$. The curves are averaged over 100 runs and show the values of the best solutions found over time (indicated by the percentage above the optimal solution value). Every 20th iteration we additionally show the standard deviation in form of an error bar. The results show that AS converges earlier and to a worse solution than AS in the HCF. The high standard deviation of AS after convergence indicates a premature convergence. (a) AS applied to instance1. (b) AS in the HCF applied to instance1.

variants,[5] and the unconstrained binary quadratic programming (UBQP) problem.[6]

1) UBQP: The UBQP problem can be stated as follows. Given are n binary decision variables X_i, $i = 1, \ldots, n$. All points in the domain space are feasible solutions. Additionally, a symmetric quadratic matrix M of size n is given whose entries m_{ij} denote the gain of having $X_i = 1$ and $X_j = 1$ in a solution s. The objective function $f(\cdot)$ is defined as follows:

$$f(s) = \sum_{i=1}^{n} \sum_{j=1}^{n} X_i \cdot X_j \cdot m_{ij}. \tag{29}$$

The goal is to maximize f. Difficult UBQP problems, which are NP-hard, have negative as well as positive entries in M (in fact, all positive entries would make the solution with $X_i = 1\ \forall i$ the

[5]$\mathcal{MM}$AS owes its name to the fact that it applies a lower bound $\tau_{\min}$ and an upper bound $\tau_{\max}$ to the pheromone values. The reason for this is to prevent the algorithm from converging to a solution (an ACO algorithm has converged to a solution s if only s has a probability greater than ϵ to be generated, with ϵ close to zero).

[6]We choose the UBQP problem because it is easy to explain and understand, so that it should not distract the reader from our main goal in this section: the illustration of the general methodology.

optimal solution; similarly for all negative entries). According to Section II, we model the UBQP as follows. For every binary decision variable X_i, there are two solution components: $c_{i,0}$ (corresponding to $X_i = 0$) and $c_{i,1}$ (corresponding to $X_i = 1$). We denote by $c_{i,1} \in s$ the fact that $X_i = 1$ in solution s. For every solution component c_{i,x_i} ($x_i \in \{0,1\}$), we have a pheromone trail parameter $\mathcal{T}_{i,x_i}$. We choose the quality function

$$F(s) = f(s) + z \tag{30}$$

where z is a constant such that $F(s) > 0 \, \forall s \in \mathcal{S}$.

2) $\mathcal{MMAS}$ in the HCF for the UBQP: A high-level description of the $\mathcal{MM}$AS algorithm in the HCF for UBQP is given in Algorithm 2.[7] The data structures used by this algorithm, in addition to counters and to the already defined pheromone trails $\mathcal{T}$, are:

- the *iteration-best* solution $s_{\rm ib}$: the best solution generated in the current iteration by the n_a ants;
- the *best-so-far* solution $s_{\rm bs}$: the best solution generated since the start of the algorithm;
- the *restart-best* solution $s_{\rm rb}$: the best solution generated since the last restart of the algorithm;
- the *convergence factor* cf, $0 \leq \mathrm{cf} \leq 1$: a measure of how far the algorithm is from convergence;
- the Boolean variable *bs_update*: it becomes true when the algorithm reaches convergence.

The algorithm works as follows (note that we give here a high-level description of the algorithm working, and that the main procedures used by the algorithm are explained in detail in the following of the section). First, all the variables are initialized. In particular, the pheromone values are set to their initial value 0.5 by the procedure InitializePheromoneValues($\mathcal{T}$). Second, the n_a ants apply the ConstructSolution($\mathcal{T}$) procedure to construct n_a solutions. These solutions are then improved by the application of the LocalSearch(s_j) procedure. Third, the value of the variables $s_{\rm ib}$, $s_{\rm rb}$ and $s_{\rm bs}$ is updated (note that, until the first restart of the algorithm, it holds that $s_{\rm rb} \equiv s_{\rm bs}$). Fourth, pheromone trail values are updated via the ApplyPheromoneUpdate(cf, *bs_update*, $\mathcal{T}$, $s_{\rm ib}$, $s_{\rm rb}$, $s_{\rm bs}$) procedure. Fifth, a new value for the converge factor cf is computed. Depending on this value, as well as on the value of the Boolean variable *bs_update*, a decision on whether to restart the algorithm or not is taken. If the algorithm is restarted, then the procedure ResetPheromoneValues($\mathcal{T}$) is applied and all the pheromones are reset to their initial value (0.5). The algorithm is iterated until some opportunely defined termination conditions are satisfied. Once terminated, the algorithm returns the best-so-far solution $s_{\rm bs}$.

The main procedures of Algorithm 2 are now described in detail.

ConstructSolution($\mathcal{T}$): Solutions are constructed by sampling a value for every decision variable $X_i (i = 1, \ldots, n)$ according to the probabilities given by

$$\mathbf{p}(X_i = x_i | \mathcal{T}) = \tau_{i,x_i}, \quad x_i \in \{0,1\}. \tag{31}$$

[7]The commented source code of the algorithm is freely available under the GNU license and can be downloaded from the software section of the ACO home page [25].

LocalSearch(s_j): To each solution constructed by the ants a steepest descent local search based on the one-flip neighborhood [26] is applied. In the one-flip neighborhood a solution s is a neighbor of a solution s' if s and s' are different in exactly one variable.

```
Algorithm 2 MMAS in the HCF for UBQP
input: a problem instance (n, M)
s_bs ←NULL
s_rb ←NULL
cf ← 0
bs_update←FALSE
InitializePheromoneValues(T)
while stop conditions not satisfied do
  for j ← 1 to n_a do
    s_j ← ConstructSolution(T)
    s_j ← LocalSearch(s_j)
  end for
  s_ib ← arg max(F(s_1), ..., F(s_{n_a}))
  if s_rb = NULL or f(s_ib) > f(s_rb) then s_rb ← s_ib
  if s_bs = NULL or f(s_ib) > f(s_bs) then s_bs ← s_ib
  ApplyPheromoneUpdate(cf, bs_update, T, s_ib, s_rb, s_bs)
  cf ←ComputeConvergenceFactor(T)
  if cf > 0.999 then
    if bs_update=TRUE then
      ResetPheromoneValues(T)
      s_rb ←NULL
      bs_update←FALSE
    else
      bs_update←TRUE
    end if
  end if
end while
output: s_bs
```

ApplyPheromoneUpdate(cf, *bs_update*, $\mathcal{T}$, $s_{\rm ib}$, $s_{\rm rb}$, $s_{\rm bs}$): In the standard $\mathcal{MM}$AS implementation as described in [8], the following schedule for updating the pheromone values is used. At the beginning of a restart phase (i.e., when *bs_update*=FALSE and cf is close to zero), the iteration best solution $s_{\rm ib}$ is used for updating the pheromone values. Then, as the algorithm progresses (i.e., when cf increases), more and more often the best solution $s_{\rm rb}$ found in the current restart phase is used. Shortly before convergence (i.e., $\mathrm{cf} \leq 0.999$, but close to 0.999), only the restart-best solution $s_{\rm rb}$ is used, and once convergence is detected (i.e., $\mathrm{cf} > 0.999$), the control variable *bs_update* is set to TRUE, which has the effect that the best-so-far solution $s_{\rm bs}$ is used for updating the pheromone values. This is done in order to intensify the search around the best-so-far solution and can have the effect to move the search in a different zone of the search space, with the consequent decrease of the convergence factor. If this is not the case, at the next iteration of the algorithm the ResetPheromoneValues($\mathcal{T}$) procedure is applied and the algorithm restarted.

As we have seen, in the schedule described above each of the three solutions, $s_{\rm ib}$, $s_{\rm rb}$ and $s_{\rm bs}$, has a certain influence (that can be zero) on the pheromones update at different stages of the

algorithm. In general, to scale this influence is quite difficult, and may require a lot of experimentation. This is not the case for an implementation of the same algorithm in the HCF. In fact, instead of using only one solution per iteration for updating the pheromone values, we can use a weighted average of the three solutions $s_{\rm ib}$, $s_{\rm rb}$ and $s_{\rm bs}$. In the HCF, the update is therefore specified by

$$\vec{\tau} \leftarrow \vec{\tau} + \rho \cdot (\vec{m} - \vec{\tau}) \tag{32}$$

with

$$\vec{m} \leftarrow \kappa_{\rm ib} \cdot \vec{s}_{\rm ib} + \kappa_{\rm rb} \cdot \vec{s}_{\rm rb} + \kappa_{\rm bs} \cdot \vec{s}_{\rm bs} \tag{33}$$

where $\kappa_{\rm ib}$ is the weight of solution $s_{\rm ib}$, $\kappa_{\rm rb}$ is the weight of solution $s_{\rm rb}$, $\kappa_{\rm bs}$ is the weight of solution $s_{\rm bs}$, and $\kappa_{\rm ib}+\kappa_{\rm rb}+\kappa_{\rm bs} = 1$. After the pheromone update rule (32) is applied, pheromone values that exceed $\tau_{\max}$ are set back to $\tau_{\max}$ (similarly for $\tau_{\min}$).

Equation (33) allows to choose how to schedule the relative influence of the three solutions used for updating pheromones. A typical update schedule used by $\mathcal{MM}$AS in the HCF is shown in Table I. This schedule uses only the iteration best solution in the early stages of the search (i.e., when ${\rm cf} < 0.4$). Then, when the value of the convergence factor increases (i.e., $0.4 \leq {\rm cf} < 0.6$), one third of the total influence is given to the restart-best solution, which then increases to two thirds when $0.6 \leq {\rm cf} < 0.8$. Eventually, all the influence is given to the restart-best solution (i.e., when ${\rm cf} \geq 0.8$). Once the value of the convergence factor raises above 0.999, the Boolean control variable *bs_update* is set to TRUE, and all the influence is given to the best-so-far solution.

ComputeConvergenceFactor($\mathcal{T}$): The convergence factor cf, which is a function of the current pheromone values, is computed as follows:

$${\rm cf} \leftarrow 2\left(\left(\frac{\sum_{\mathcal{T}_{i,x_i} \in \mathcal{T}} \max\{\tau_{\max} - \tau_{i,x_i}, \tau_{i,x_i} - \tau_{\min}\}}{|\mathcal{T}| \cdot (\tau_{\max} - \tau_{\min})}\right) - 0.5\right).$$

This formula says that when the algorithm is initialized (or reset) so that all pheromone values are set to 0.5, then ${\rm cf} = 0$, while when the algorithm has converged, then ${\rm cf} = 1$. In all other cases, cf has a value in (0,1).

Finally, we want to point out another benefit of implementing $\mathcal{MM}$AS in the HCF. This benefit is in setting the upper bound $\tau_{\max}$, respectively, the lower bound $\tau_{\min}$, for the pheromone values. As we have seen in Theorem 1, the upper limit[8] of the pheromone values in standard ACO implementations depends on the value of an optimal solution to the problem instance tackled. Therefore, the upper limit is in general unknown, so that it is difficult to introduce an upper bound for the pheromone values. In the original paper on $\mathcal{MM}$AS [8], the upper bound $\tau_{\max}$ is set to $F(s_{\rm bs})/(1-\rho)$, which is an approximation of the upper limit. This means that the upper bound has to be reset each time a new best-so-far solution $s_{\rm bs}$ is found. As the lower bound $\tau_{\min}$ depends on the setting of $\tau_{\max}$ (see [8]), $\tau_{\min}$ has also to be updated every time a new best-so-far solution is found. However, if the $\mathcal{MM}$AS algorithm is implemented in the HCF, we can avoid this inconvenience, as the upper and lower limits of the pheromone values are known to be, respectively, 1 and 0. For our $\mathcal{MM}$AS algorithm in the HCF for UBQP, we set the upper bound to 0.999 and the lower bound to 0.001.

[8]We distinguish between "upper limit" and "upper bound", that is a value lower than the upper limit. The upper bound prevents the pheromone values from reaching the upper limit. Similarly, for "lower limit" and "lower bound".

TABLE I
SETTING OF $\kappa_{\rm ib}$, $\kappa_{\rm rb}$, AND $\kappa_{\rm bs}$ DEPENDING ON THE CONVERGENCE FACTOR cf AND THE BOOLEAN CONTROL VARIABLE *bs_update*

	bs_update = FALSE				*bs_update*
	cf < 0.4	*cf* ∈ [0.4, 0.6)	*cf* ∈ [0.6, 0.8)	*cf* ≥ 0.8	= TRUE
κ_{ib}	1	2/3	1/3	0	0
κ_{rb}	0	1/3	2/3	1	0
κ_{bs}	0	0	0	0	1

TABLE II
COMPARISON OF $\mathcal{MM}$AS IN THE HCF WITH THREE METHODS FROM THE LITERATURE ON 30 OF THE BIGGEST AVAILABLE UBQP PROBLEM INSTANCES. THE FIRST COLUMN OF EACH TABLE GIVES THE NAME OF THE PROBLEM, THE SECOND COLUMN THE BEST-KNOWN RESULT, AND THE OTHER COLUMNS THE RESULTS OBTAINED BY THE FOUR ALGORITHMS COMPARED. FOR EACH OF THESE COLUMNS, VALUES THAT ARE AS GOOD AS THE BEST-KNOWN RESULT ARE SHOWN IN BOLDFACE. (a) COMPARISON ON THE 10 BIGGEST PROBLEM INSTANCES PROVIDED BY GLOVER *et al.* IN [30]. (b) COMPARISON ON 20 OF THE BIGGEST PROBLEM INSTANCES PROVIDED BY BEASLEY IN [28]

Instance	Best known	$\mathcal{MM}$AS in the HCF	GLS [26]	STS [28]	SSA [28]
200-1e	16464	**16464**	**16464**	**16464**	16458
200-2e	23395	**23395**	**23395**	**23395**	**23395**
200-3e	25243	**25243**	**25243**	**25243**	**25243**
200-4e	35594	**35594**	**35594**	**35594**	**35594**
200-5e	35154	**35154**	**35154**	**35154**	**35154**
500-1f	61194	**61194**	**61194**	**61194**	61183
500-2f	100161	**100161**	**100161**	100158	100158
500-3f	138035	**138035**	**138035**	138021	**138035**
500-4f	172771	172734	not given	**172771**	172150
500-5f	190507	**190507**	**190507**	190502	190498

(a)

Instance	Best known	$\mathcal{MM}$AS in the HCF	GLS [26]	STS [28]	SSA [28]
500-1bea	116586	**116586**	**116586**	**116586**	**116586**
500-2bea	128339	**128339**	**128339**	128223	128204
500-3bea	130812	**130812**	**130812**	**130812**	**130812**
500-4bea	130097	**130097**	**130097**	**130097**	130077
500-5bea	125487	**125487**	**125487**	**125487**	125315
500-6bea	121772	**121772**	**121772**	121719	121719
500-7bea	122201	**122201**	**122201**	**122201**	**122201**
500-8bea	123559	**123559**	**123559**	**123559**	123469
500-9bea	120798	**120798**	**120798**	120797	**120798**
500-10bea	130619	**130619**	**130619**	**130619**	**130619**
1000-1bea	371438	371336	**371438**	**371438**	371134
1000-2bea	354932	**354932**	**354932**	**354932**	354637
1000-3bea	371236	**371236**	**371236**	371073	371226
1000-4bea	370675	**370675**	**370675**	370560	370265
1000-5bea	352760	352730	352730	352736	352297
1000-6bea	359629	**359629**	**359629**	359452	359313
1000-7bea	371193	**371193**	**371193**	370999	370815
1000-8bea	351994	351886	**351994**	351836	351001
1000-9bea	349337	349254	349254	348732	348309
1000-10bea	351415	**351415**	351408	351408	**351415**

(b)

VI. SOME COMPUTATIONAL RESULTS

As already mentioned, the main contribution of this paper is in the methodology that it puts forward, rather than in the experimental results obtained. However, the simple (and not tuned)

algorithm that we described above (Algorithm 2) achieves results that compete in quality with other methods for the UBQP published in the scientific literature. We show this by applying $\mathcal{MM}$AS in the HCF to 30 of the biggest benchmark instances available at the OR-LIB [27], and by comparing the results (in terms of the best solution values found) to three other methods, namely genetic local search (GLS) by Merz and Freisleben [26], simple tabu search (STS) and simple simulated annealing (SSA) by Beasley [28]. The GLS algorithm uses exactly the same local search procedure as $\mathcal{MM}$AS in the HCF, while STS and SSA are both based on the one-flip neighborhood, which is the neighborhood that is used by the local search procedure used by GLS and $\mathcal{MM}$AS in the HCF. We applied $\mathcal{MM}$AS in the HCF twenty times to each problem instance with the parameter setting $\rho = 0.05$, $n_a = 10$ (where n_a is the number of ants), and κ_{ib}, κ_{rb}, and κ_{bs} as specified in Table I. As the time limit for each trial, we set 20 s for the 200-variable problems, 30 s for the 500-variable problems, and 200 s for the 1000-variable problems. The results for $\mathcal{MM}$AS in the HCF were obtained on a PC with Athlon 1100 MHz processor under Linux. Because the computers used to run experiments are very different and average results are available only for GLS, we limit the comparison of the four algorithms to the best solution values they found.[9] The results are presented in Table II. They show that $\mathcal{MM}$AS in the HCF and GLS have a comparable performance (i.e., $\mathcal{MM}$AS in the HCF finds the best known solution for 25 of the 30 problem instances, whereas GLS does so for 26 of the 30 problem instances.). Both algorithms clearly outperform STS and SSA. In this context, it is interesting to note that GLS, and therefore also $\mathcal{MM}$AS in the HCF, are not much worse than more sophisticated state-of-the-art algorithms such as the one proposed in [29].

VII. Conclusion

In this paper, we have proposed the HCF for ant colony optimization (HCF-ACO). This framework brings two main benefits to ACO researchers. First, from the point of view of the theoretician, we prove that AS in the HCF, when applied to unconstrained problems, generates solutions whose expected value monotonically does not decrease with the number of algorithm iterations. Second, from the point of view of the experimental researcher, we show through two examples that the implementation of ACO algorithms in the HCF increases their robustness and makes it easier the handling of the pheromones.

It is also interesting to note that the HCF helps in clarifying the relations between ant colony optimization and other MBS algorithms like population-based incremental learning [31] and the univariate marginal distribution algorithm [32]. We refer the interested reader to [5] for more information on this subject.

References

[1] M. Dorigo, "Optimization, Learning and Natural Algorithms (in Italian)," Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1992.

[2] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive Feedback as a Search Strategy," Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, Tech. Rep. 91-016, 1991.

[3] ——, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, pp. 29–41, Feb. 1996.

[4] M. Zlochin and M. Dorigo, "Model-based search for combinatorial optimization: a comparative study," in *Proc. PPSN-VII, 7th Int. Conf. Parallel Problem Solving From Nature*, vol. 2439, Lecture Notes in Computer Science, J. J. Merelo, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacanas, and H.-P. Schwefel, Eds. Berlin, Germany, 2002, pp. 651–661.

[5] M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo, "Model-based search for combinatorial optimization: A critical survey," Ann. Oper. Res., 2004, to be published.

[6] C. Blum, A. Roli, and M. Dorigo, "HC-ACO: the hyper-cube framework for ant colony optimization," in *Proc. MIC'2001—Metaheuristics Int. Conf.*, vol. 2, 2001, pp. 399–403.

[7] M. Dorigo and G. Di Caro, "The ant colony optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, UK: McGraw-Hill, 1999, pp. 11–32.

[8] T. Stützle and H. H. Hoos, "$\mathcal{MAX} - \mathcal{MIN}$ ant system," *Future Gen. Comput. Syst.*, vol. 16, no. 8, pp. 889–914, 2000.

[9] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: algorithms, applications and advances," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger, Eds. Norwell, MA: Kluwer, 2002, vol. 57, International Series in Operations Research & Management Science, pp. 251–285.

[10] ——, *Ant Colony Optimization*. Cambridge, MA: MIT Press, 2004.

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

[12] E. Lawler, A. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, *The Traveling Salesman Problem*. New York: Wiley, 1985.

[13] S. Martello and P. Toth, *Knapsack Problems*. New York: Wiley, 1990.

[14] M. Fischetti, H. Hamacher, K. Jørnsten, and F. Maffioli, "Weighted k-cardinality trees: complexity and polyhedral structure," *Networks*, vol. 24, pp. 11–21, 1994.

[15] C. Blum, "Ant colony optimization for the edge-weighted k-cardinality tree problem," in *GECCO 2002: Proc. Genetic and Evolutionary Computation Conf.*, W. B. Langdon *et al.*, Ed. San Francisco, CA: Morgan Kaufmann, 2002, pp. 27–34.

[16] A. Prügel-Bennett and A. Rogers, "Modeling genetic algorithm dynamics," in *Theoretical Aspects of Evolutionary Computation*, L. Kallel *et al.*, Ed. Berlin, Germany: Springer-Verlag, 2001, Natural Computing Series, pp. 59–85.

[17] L. Baum and G. Sell, "Growth transformations for functions on manifolds," *Pacific J. Math.*, vol. 27, no. 2, pp. 211–227, 1968.

[18] D. Bertsekas, *Nonlinear Programming*. Belmont, MA: Athena, 1995.

[19] D. Luenberger, *Linear and Nonlinear Programming*. Reading, MA: Addison-Wesley, 1984.

[20] R. A. Holmgren, *A First Course in Discrete Dynamical Systems*. Berlin, Germany: Springer-Verlag, 1996.

[21] M. Pelillo, "Relaxation labeling networks for the maximum clique problem," *J. Artif. Neural Net.*, vol. 2, no. 4, pp. 313–328, 1995.

[22] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition," *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1035–1074, 1983.

[23] J. Hofbauer and K. Sigmund, *Evolutionary Games and Population Dynamics*. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[24] H. Mühlenbein and T. Mahnig, "Evolutionary computation and Wright's equation," *Theor. Comput. Sci.*, vol. 287, no. 1, pp. 145–165, 2002.

[25] ACO. (2004, Jan.) The Ant Colony Optimization Home Page. [Online] Available: http://www.aco-metaheuristic.org

[26] P. Merz and B. Freisleben, "Genetic algorithms for binary quadratic progamming," in *GECCO 1999: Proc. Genetic and Evolutionary Computation Conf.*, W. Banzhaf *et al.*, Ed. San Mateo, CA: Morgan Kaufmann, 1999, vol. 1, pp. 417–424.

[27] J. E. Beasley, "OR-Library: distributing test problems by electronic mail," *J. Oper. Res. Soc.*, vol. 41, no. 11, pp. 1069–1072, 1990.

[28] J. E. Beasley, "Heuristic algorithms for the unconstrained binary quadratic programming problem," Tech. Rep., Management School, Imperial College, London, U.K., 1998.

[29] A. Lodi, K. Allemand, and T. M. Liebling, "An evolutionary heuristic for quadratic 0-1 programming," *Eur. J. Oper. Res.*, vol. 119, pp. 662–670, 2000.

[9] However, if we normalize the computation times with respect to the different computers speed, then the computation times of $\mathcal{MM}$AS in the HCF and GLS are comparable, and much lower than those of STS and SSA.

[30] F. Glover, G. Kochenberger, and B. Alidaee, "Adaptive memory tabu search for binary quadratic programs," *Manag. Sci.*, vol. 44, pp. 336–345, 1998.
[31] S. Baluja and R. Caruana, "Removing the genetics from the standard genetic algorithm," in *Proc. 12th Int. Conf. Machine Learning (ML-95)*, A. Prieditis and S. Russel, Eds. San Mateo, CA: Morgan Kaufmann, 1995, pp. 38–46.
[32] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions," in *Proc. 4th Conf. Parallel Problem Solving From Nature—PPSN IV*, H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, Eds. Berlin, Germany: Springer-Verlag, 1996, vol. 1411, Lecture Notes in Computer Science, pp. 178–187.

Christian Blum (S'02) received the M.S. degree in mathematics from the University of Kaiserslautern, Kaiserslautern, Germany, in 1998. He is currently working toward the Ph.D. degree at IRIDIA, Université Libre de Bruxelles, Belgium,

From 1999 to 2000, he was with the Advanced Computation Laboratory (ACL), Imperial Cancer Research Fund (ICRF), London, U.K. In 2000, he joined IRIDIA, where he participates in the activities of the Metaheuristics Network project, which is funded by the European Commission. His research interests include metaheuristics in combinatorial optimization with a focus on theoretical and practical aspects of ant colony optimization.

Marco Dorigo (S'92–M'93–SM'96) received the M.Tech. degree in industrial technologies engineering in 1986 and the Ph.D. degree in information and systems electronic engineering in 1992, both from the Politecnico di Milano, Milan, Italy, and the title of Agrégé de l'Enseignement Supérieur, from the Université Libre de Bruxelles, Belgium, in 1995.

From 1992 to 1993, he was a Research Fellow at the International Computer Science Institute, Berkeley, CA. In 1993, he was a NATO-CNR Fellow, and from 1994 to 1996, a Marie Curie Fellow. Since 1996, he has been a tenured Researcher of the FNRS, the Belgian National Fund for Scientific Research, and a Research Director of IRIDIA, the Artificial Intelligence Laboratory of the Université Libre de Bruxelles. He is the inventor of the ant colony optimization metaheuristic. His current research interests include metaheuristics for discrete optimization, swarm intelligence, and swarm robotics.

Dr. Dorigo is an Associate Editor for the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and for the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS. He is a member of the Editorial Board of numerous international journals, including *Adaptive Behavior, AI Communications, Artificial Life, Evolutionary Computation, Journal of Heuristics, Cognitive Systems Research,* and the *Journal of Genetic Programming and Evolvable Machines*. In 1996, he was awarded the Italian Prize for Artificial Intelligence and in 2003 the Marie Curie Excellence Award.