# Updating ACO Pheromones Using Stochastic Gradient Ascent and Cross-Entropy Methods

Marco Dorigo[1], Mark Zlochin[2,⋆], Nicolas Meuleau[3], and Mauro Birattari[4]

[1] IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
mdorigo@ulb.ac.be
[2] Dept. of Computer Science, Technion – Israel Institute of Technology, Haifa, Israel
zmark@cs.technion.ac.il
[3] IRIDIA, Université Libre de Bruxelles, Brussels, Belgium
nmeuleau@iridia.ulb.ac.be
[4] Intellektik, Darmstadt University of Technology, Darmstadt, Germany
mbiro@intellektik.informatik.tu-darmstadt.de

**Abstract.** In this paper we introduce two systematic approaches, based on the stochastic gradient ascent algorithm and the cross-entropy method, for deriving the pheromone update rules in the Ant colony optimization metaheuristic. We discuss the relationships between the two methods as well as connections to the update rules previously proposed in the literature.

## 1 Introduction

The necessity to solve $\mathcal{NP}$-hard optimization problems, for which the existence of efficient exact algorithms is highly unlikely, has led to a wide range of stochastic approximation algorithms. Many of these algorithms, that are often referred to as *metaheuristics*, are not specific to a particular combinatorial problem, but rather present a general approach for organizing the search in the solution space. Examples of well-known metaheuristics are evolutionary algorithms, simulated annealing and tabu search.

At a very abstract level, many metaheuristics can be seen as methods that look for good solutions (possibly optimal ones) by repeating the following two steps:

1. Candidate solutions are constructed using some parameterized probabilistic model, that is, a parameterized probability distributions over the solution space.
2. The candidate solutions are used to update the model's parameters in a way that is deemed to bias future sampling toward low cost solutions.

Recently, a metaheuristic inspired by the foraging behavior of ant has been defined. This metaheuristic, called *ant colony optimization* (ACO), has been successfully applied to the solution of numerous NP-hard problems [1] as well as to dynamic optimization problems [2]. The main innovation of the ACO metaheuristic consists in proposing a novel way to implement the two steps described above:

⋆ This work was carried out while the author was at IRIDIA, Université Libre de Bruxelles, Belgium.

S. Cagnoni et al. (Eds.): EvoWorkshops 2002, LNCS 2279, pp. 21–30, 2002.

1. A structure called *construction graph* is coupled with a set of stochastic agents called *artificial ants*, which build new solutions using local information, called *pheromone*,[1] stored in the construction graph.
2. Once ants have built solutions, they use information collected during the construction phase to update the pheromone values.

The exact form of the pheromone update (i.e., the second component of the general approach outlined above) is not strictly defined and a good amount of freedom is left to the algorithm designer concerning how it should be implemented. Various model update rules have been proposed within the ACO framework, but they are all of a somewhat heuristic nature and are lacking a theoretical justification.

On the other hand, the *stochastic gradient ascent* (SGA) [3] and the *cross-entropy* (CE) [4] methods suggest principled ways of updating the parameters associated to the construction graph. In the following we show how the SGA and the CE methods can be cast into the ACO framework. While these two methods have different motivations, we find that in some cases the CE method leads to the same update rule as does SGA. Moreover, quite unexpectedly, some existing ACO updates are re-derived as a particular implementation of the CE method.

The paper is organized as follows. In Section 2 we present the solution construction mechanism used by the ACO metaheuristic. Section 3 presents several typical pheromone update rules among those most used by practitioners and researchers. In Section 4 and 5 we introduce the stochastic gradient ascent and the cross entropy methods respectively, and we show how, once cast into the ACO framework, they can be used to define novel pheromone updating rules. We conclude in Section 6 with a brief summary of the contributions of this short paper.

## 2 Solution Construction in ACO Metaheuristic

Let us consider a minimization problem[2] $(\mathcal{S}, f)$, where $\mathcal{S}$ is the *set of feasible solutions* and $f$ is the *objective function*, which assigns to each solution $s \in \mathcal{S}$ a cost value $f(s)$. The goal of the minimization problem is to find an optimal solution $s^*$, that is, a feasible solution of minimum cost. The set of all optimal solutions is denoted by $\mathcal{S}^*$. We assume that the combinatorial optimization problem $(\mathcal{S}, f)$ is mapped on a problem that can be characterized by the following list of items[3]:

- A finite set $\mathcal{C} = \{c_1, c_2, \ldots, c_{N_C}\}$ of *components*.
- A finite set $\mathcal{X}$ of *states* of the problem, defined in terms of all the possible sequences $x = \langle c_i, c_j, \ldots, c_k, \ldots \rangle$ over the elements of $\mathcal{C}$. 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 $n < +\infty$.

[1] This is the terminology used in the ACO literature for historical reasons. We will keep with this tradition in this paper.

[2] The obvious changes must be done if a maximization problem is considered.

[3] How this mapping can be done in practice has been described in a number of earlier papers on the ACO metaheuristic; see, for example, [1].

- The set of (candidate) solutions $\mathcal{S}$ is a subset of $\mathcal{X}$ (i.e., $\mathcal{S} \subseteq \mathcal{X}$).
- A set of feasible states $\tilde{\mathcal{X}}$, with $\tilde{\mathcal{X}} \subseteq \mathcal{X}$, defined via a set of *constraints* $\Omega$.
- A non-empty set $\mathcal{S}^*$ of optimal solutions, with $\mathcal{S}^* \subseteq \tilde{\mathcal{X}}$ and $\mathcal{S}^* \subseteq \mathcal{S}$.

Given the above formulation, artificial ants build candidate solutions by performing randomized walks on the completely connected, weighted graph $\mathcal{G} = (\mathcal{C}, \mathcal{L}, \mathcal{T})$, where the vertices are the components $\mathcal{C}$, the set $\mathcal{L}$ fully connects the components $\mathcal{C}$, and $\mathcal{T}$ is a vector gathering so-called *pheromone trails* $\tau$.[4] The graph $\mathcal{G}$ is called *construction graph*.

Each artificial ant is put on a randomly chosen vertex of the graph and then it performs a randomized walk by moving at each step from vertex to vertex in the graph in such a way that the next vertex is chosen stochastically according to the strength of the pheromone currently on the arcs.[5] While moving from one node to another of the graph $\mathcal{G}$, constraints $\Omega$ may be used to prevent ants from building infeasible solutions. Formally, the solution construction behavior of a generic ant can be described as follows:

ANT_SOLUTION_CONSTRUCTION
for each ant:
- select a start node $c_1$ according to some problem dependent criterion,
- set $k = 1$ and $x_k = \langle c_1 \rangle$.
- While $(x_k = \langle c_1, c_2, \ldots, c_k \rangle \in \tilde{\mathcal{X}}$ and $x_k \notin \mathcal{S}$ and $J_{x_k} \neq \emptyset)$ do:
  at each step $k$, after building the sequence $x_k$, select the next node (component) $c_{k+1}$ randomly following

$$P_{\mathcal{T}}(c_{k+1} = c | x_k) = \begin{cases} \dfrac{F_{(c_k,c)}\big(\tau(c_k,c)\big)}{\sum\limits_{(c_k,y)\in J_{x_k}} F_{(c_k,y)}\big(\tau(c_k,y)\big)} & \text{if } (c_k,c) \in J_{x_k}, \\ 0 & \text{otherwise;} \end{cases} \tag{1}$$

  where a connection $(c_k, y)$ belongs to $J_{x_k}$ iff the sequence $x_{k+1} = \langle c_1, c_2, \ldots, c_k, y \rangle$ satisfies the constraints $\Omega$ (i.e., $x_{k+1} \in \tilde{\mathcal{X}}$) and $F_{(i,j)}(z)$ is some monotonic function (most commonly, $z^\alpha \eta(i,j)^\beta$, where $\alpha, \beta > 0$ and $\eta$ are heuristic "visibility" values [6]). If at some stage $x_k \notin \mathcal{S}$ and $J_{x_k} = \emptyset$, the construction process has reached a dead-end and is therefore abondoned.[6]

After the solution construction has been completed, the artificial ants update the pheromone values. Next we describe several typical updates that were suggested in the past within the ACO framework.

[4] Pheromone trails can be associated to components, connections, or both. In the following, unless stated otherwise, we assume that the pheromone trails are associated to connections, so that $\tau(i,j)$ is the pheromone associated to the connection between components $i$ and $j$. It is straightforward to extend algorithms to the other cases.

[5] It should be noted that the same type of model was later (but independently) used in the CE framework under the name "associated stochastic network" [4,5].

[6] This situation may be prevented by allowing artificial ants to build infeasible solutions as well. In such a case an infeasibility penalty term is usually added to the cost function. It should be noted, however, that in most settings ACO was applied to, the dead-end situation does not occur.

## 3 Typical ACO Pheromone Updates

Many different schemes for pheromone update have been proposed within the ACO framework. Most of them can be described, however, using the following generic scheme:

GENERIC_ACO_UPDATE
- $\forall s \in \hat{S}_t, \forall (i,j) \in s: \ \tau(i,j) \leftarrow \tau(i,j) + Q_f(s|S_1, \ldots, S_t)$
- $\forall (i,j): \ \tau(i,j) \leftarrow (1-\rho) \cdot \tau(i,j)$
  where $S_i$ is the sample in the $i$-th iteration, $\rho$, $0 \leq \rho < 1$, is the evaporation rate and $Q_f(s|S_1, \ldots, S_t)$ is some "quality function", which is typically required to be non-increasing with respect to $f$ and is defined over the "reference set" $\hat{S}_t$.

Different ACO algorithms may use different quality functions and reference sets. For example, in the very first ACO algorithm — Ant System [7] — the quality function was simply $1/f(s)$ and the reference set $\hat{S}_t = S_t$. In a more recently proposed scheme, called *iteration best update* [6], the reference set was a singleton containing the best solution within $S_t$ (if there were several iteration-best solutions, one of them was chosen randomly). For the *global-best update* [8,6], the reference set contained the best among all the iteration-best solutions (and if there were more than one global-best solution, the earliest one was chosen). In [7] an *elitist* strategy was introduced, in which the update was a combination of the previous two.

A somewhat different pheromone update was used in *ant colony system* (ACS) [6]. There the pheromones are evaporated by the ants online during the solution construction, hence only the pheromones involved in the construction evaporate.

Another modification of the generic update described above was recently proposed under the name Hyper-Cube (HC) ACO [9]. The HC-ACO, applied to combinatorial problems with binary coded solutions,[7] normalizes the quality function, hence obtaining an automatic scaling of the pheromone values:

$$\tau_i \leftarrow (1-\rho)\tau_i + \rho \, \frac{\sum_{\substack{s \in \hat{S}_t \\ s_i = 1}} Q_f(s)}{\sum_{s \in \hat{S}_t} Q_f(s)}. \tag{2}$$

While all the updates described above are of somewhat heuristic nature, the SGA and the CE methods allow to derive pheromone update rules in a more systematic manner, as we show next.

## 4 Stochastic Gradient Ascent

The construction process described above implicitly defines a probability distribution over the solution space. Let us denote this distribution by $P_{\mathcal{T}}$, where $\mathcal{T}$ is the vector of

[7] Using the notation of Section 2, in such a case the components are bit assignments to the locations, and the pheromone values are associated with the components, rather than with the connections.

pheromone values. Now, the original optimization problem may be replaced with the following equivalent continuous *maximization problem*:

$$\mathcal{T}^* = \operatorname*{argmax}_{\mathcal{T}} \mathcal{E}(\mathcal{T}), \tag{3}$$

where $\mathcal{E}(\mathcal{T}) = E_{\mathcal{T}} Q_f(s)$, $E_{\mathcal{T}}$ denotes expectation with respect to $P_{\mathcal{T}}$, and $Q_f(s)$ is a fixed *quality function*, which is strictly decreasing with respect to $f$. It may be easily verified that $P_{\mathcal{T}^*}$ is greater than zero only over $\mathcal{S}^*$, hence solving problem (3) is equivalent to solving the original combinatorial optimization problem.

One may then search for an optimum (possibly a local one) of problem (3) using a gradient ascent method (in other words, gradient ascent may be used as a heuristic to change $\mathcal{T}$ with the goal of solving (3)):

- Start from some initial guess $\mathcal{T}^0$.
- At stage $t$, calculate the gradient $\nabla\mathcal{E}(\mathcal{T}^t)$ and update $\mathcal{T}^{t+1}$ to be $\mathcal{T}^t + \alpha_t \nabla\mathcal{E}(\mathcal{T}^t)$.

The gradient can be calculated (theoretically) as follows:

$$\begin{aligned}\nabla\mathcal{E} = \nabla E_{\mathcal{T}} Q_f(s) &= \nabla \sum_s Q_f(s) P_{\mathcal{T}}(s) = \sum_s Q_f(s) \nabla P_{\mathcal{T}}(s) \\ &= \sum_s P_{\mathcal{T}}(s) Q_f(s) \nabla \ln P_{\mathcal{T}}(s) = E_{\mathcal{T}} Q_f(s) \nabla \ln P_{\mathcal{T}}(s). \end{aligned} \tag{4}$$

However, the gradient ascent algorithm cannot be implemented in practice, as for its evaluation a summation over the whole search space is needed. A more practical alternative would be to use *stochastic gradient ascent* [3], which replaces the expectation in Equation 4 by an empirical mean of a sample generated from $P_{\mathcal{T}}$.

The update rule for the stochastic gradient is:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s), \tag{5}$$

where $S_t$ is the sample at iteration $t$. It remains to be shown how the gradient $\nabla \ln P_{\mathcal{T}^t}(s)$ can be evaluated. The following calculation is a generalization of the one in [10].

From the definition of ANT_SOLUTION_CONSTRUCTION, it follows that, for $s = \langle c_1, c_2, \ldots \rangle$,

$$P_{\mathcal{T}}(s) = \prod_{k=1}^{|s|-1} P_{\mathcal{T}}\big(c_{k+1} |\, \mathrm{pref}_k(s)\big), \tag{6}$$

where $\mathrm{pref}_k(s)$ is the $k$-prefix of $s$, and consequently

$$\nabla \ln P_{\mathcal{T}}(s) = \sum_{k=1}^{|s|-1} \nabla \ln P_{\mathcal{T}}\big(c_{k+1} |\, \mathrm{pref}_k(s)\big). \tag{7}$$

Finally, given a pair of components $(i,j) \in \mathcal{C}^2$, using Equation (1), it is easy to verify that:

– if $i = c_k$ and $j = c_{k+1}$ then

$$\frac{\partial}{\partial\tau(i,j)}\Big(\ln P_{\mathcal{T}}\big(c_{k+1}|\ \text{pref}_k(s)\big)\Big) =$$

$$\frac{\partial}{\partial\tau(i,j)}\Big(\ln F\big(\tau(i,j)\big) - \ln \sum_{(i,y)\in J_{x_k}} F\big(\tau(i,y)\big)\Big) =$$

$$\left(1 - F\big(\tau(i,j)\big) \Big/ \sum_{(i,y)\in J_{x_k}} F\big(\tau(i,y)\big)\right) \frac{F'\big(\tau(i,j)\big)}{F\big(\tau(i,j)\big)} =$$

$$\Big(1 - P_{\mathcal{T}}\big(j|\ \text{pref}_k(s)\big)\Big) G\big(\tau(i,j)\big),$$

where $G(\cdot) = F'(\cdot)/F(\cdot)$ and the subscript of $F$ was omitted for the clarity of presentation.

– if $i = c_k$ and $j \neq c_{k+1}$ then

$$\frac{\partial \ln\Big(P_{\mathcal{T}}\big(c_{k+1}|\ \text{pref}_k(s)\big)\Big)}{\partial\tau(i,j)} = -P_{\mathcal{T}}\big(j|\ \text{pref}_k(s)\big) G\big(\tau(i,j)\big).$$

By combining these results, the following pheromone update rule is derived:

SGA_UPDATE

$\forall s = \langle c_1, \ldots, c_k, \ldots\rangle \in S_t, 1 \leq k < |s|$ :

– $\tau(c_k, c_{k+1}) \leftarrow \tau(c_k, c_{k+1}) + \alpha_t Q_f(s) G\big(\tau(c_k, c_{k+1})\big)$

– $\forall y : \tau(c_k, y) \leftarrow \tau(c_k, y) - \alpha_t Q_f(s) P_{\mathcal{T}}\big(y|\ \text{pref}_k(s)\big) G\big(\tau(c_k, y)\big)$

Hence any connection $(i,j)$ used in the construction of a solution is reinforced by an amount $\alpha_t Q_f(s) G\big(\tau(i,j)\big)$, and any connection *considered* during the construction, has its pheromone values evaporated by an amount $\alpha_t Q_f(s) P_{\mathcal{T}}\big(j|\ \text{pref}_k(s)\big) G\big(\tau(i,j)\big)$. Note, that if the solutions are allowed to contain loops, a connection may be updated more than once for the same solution.

In order to guarantee the stability of the resulting algorithm, it is desirable for the estimate of the gradient $\nabla \ln P_{\mathcal{T}}(s)$ to be bounded. This means that a function $F$, for which $G = F'/F$ is bounded, should be used. Meuleau and Dorigo [10] suggest using $F(\cdot) = \exp(\cdot)$, which leads to $G \equiv 1$. It should be further noted that if, in addition, $Q_f = 1/f$ and $\alpha_t = 1$, the reinforcement part becomes $1/f$, as in Ant System.

## 5 Cross-Entropy Method

The cross-entropy (CE) method was initially proposed in the stochastic simulation field as a tool for rare events estimation and later adapted as a tool for combinatorial optimization

[4]. In this overview we present a more straightforward derivation of the cross-entropy method (as a combinatorial optimization tool), without reference to the rare events estimation.

Starting from some initial distribution $P_0 \in \mathcal{M}$, the CE method inductively builds a series of distributions $P_t \in \mathcal{M}$, in an attempt to increase the probability of generating low-cost solutions after each iteration. A tentative way to achieve this goal is to set $P_{t+1}$ equal to

$$\hat{P} \propto P_t Q_f, \tag{8}$$

where $Q_f$ is, again, some quality function dependent on the cost value.

If this were possible, after $n$ iteration we would obtain $P_n \propto P_0 Q_f^n$, and as $n \to \infty$, $P_n$ would converge to a probability distribution restricted to $\mathcal{S}^*$. Unfortunately, even if the distribution $P_t$ belongs to the family $\mathcal{M}$, the distribution $\hat{P}$ as defined by (8) does not necessarily remain in $\mathcal{M}$, hence some sort of projection is needed. A natural candidate for $P_{t+1}$, is the distribution $P \in \mathcal{M}$ that minimizes the *Kullback-Leibler divergence* [11], which is a commonly used measure of misfit between two distributions:

$$D(\hat{P} \| P) = \sum_s \hat{P}(s) \ln \frac{\hat{P}(s)}{P(s)}, \tag{9}$$

or, equivalently, the *cross-entropy*: $-\sum_s \hat{P}(s) \ln P(s)$.

Since $\hat{P} \propto P_t Q_f$, cross-entropy minimization is equivalent to the following maximization problem

$$P_{t+1} = \operatorname*{argmax}_{P \in \mathcal{M}} \sum_s P_t(s) Q_f(s) \ln P(s). \tag{10}$$

It should be noted that, unlike SGA, in the cross-entropy method the quality function is only required to be non-increasing with respect to the cost and may also depend on the iteration index, either deterministically or stochastically, for example, depending on the points sampled so far. One common choice is, for example, $Q_f^t(s) = I(f(s) < f_t)$, where $I(\cdot)$ is an indicator function, and $f_t$ is, for example, some quantile (e.g., lower 10%) of the cost distribution during the last iteration.

Similarly to the gradient ascent algorithm, the maximization problem (10) cannot be solved in practice, as the evaluation of the function $\sum_s P_t(s) Q_f(s) \ln P(s)$ requires summation over the whole solution space, and once again a finite sample approximation is used instead:

$$P_{t+1} = \operatorname*{argmax}_{P \in \mathcal{M}} \sum_{s \in S_t} Q_f(s) \ln P(s), \tag{11}$$

where $S_t$ is a sample from $P_t$.

Let us now consider problem (11) in more details. At the maximum the gradient must be zero:

$$\sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}}(s) = 0. \tag{12}$$

In some relatively simple cases, for example, when the solution $s$ is represented by an unconstrained string of bits of length $n$, $(s_1, \ldots, s_n)$, and there is a single parameter $\tau_i$ for

the $i$-th position in the string,[8] such that $P_{\mathcal{T}}(s) = \prod_i p_{\tau_i}(s_i)$, the equations system (12) reduces to a set of independent equations:

$$\frac{d \ln p_{\tau_i}}{d\tau_i} \sum_{\substack{s \in S_t \\ s_i = 1}} Q_f(s) = -\frac{d \ln(1 - p_{\tau_i})}{d\tau_i} \sum_{\substack{s \in S_t \\ s_i = 0}} Q_f(s), \tag{13}$$

which may often be solved analytically. For example, for $p_{\tau_i} = \tau_i$ it can be easily shown that the solution of Equation (13) is simply

$$\tau_i = \frac{\sum_{\substack{s \in S_t \\ s_i = 1}} Q_f(s)}{\sum_{s \in S_t} Q_f(s)}. \tag{14}$$

Now, since the pheromone trails $\tau_i$ in (14) are random variables, whose values depend on the particular sample, we may wish to make our algorithm more robust by introducing some conservatism into the update. For example, rather than discarding the old pheromone values, the new values may be taken to be a convex combination of the old values and the solution (14):

$$\tau_i \leftarrow (1 - \rho)\tau_i + \rho \frac{\sum_{\substack{s \in S_t \\ s_i = 1}} Q_f(s)}{\sum_{s \in S_t} Q_f(s)}. \tag{15}$$

The resulting update is identical to the one used in the Hyper-Cube ACO [9].

In general, however, Equations (12) are coupled and an analytical solution is unavailable. Nevertheless, in the actual implementations of the CE method the update was of the form (14) (with some brief remarks about using (15)) [5], which may be considered as an approximation to the exact solution of the cross-entropy minimization problem (11).

Still, even if the exact solution is not known, some iterative methods for solving this optimization problem may be used. A natural candidate for the iterative solution of the maximization problem (11) is gradient ascent:

- Start with $\mathcal{T}' = \mathcal{T}^t$. (Other starting points are possible, but this is the most natural one, since we may expect $\mathcal{T}^{t+1}$ to be close to $\mathcal{T}^t$.)
- Repeat:
  - $\mathcal{T}' \leftarrow \mathcal{T}' + \alpha \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}'}(s)$
  - until some stopping criteria is satisfied.
- Set $\mathcal{T}^{t+1} = \mathcal{T}'$.

It should be noted that, since the new vector $\mathcal{T}^{t+1}$ is a random variable, depending on a sample, there is no use in running the gradient ascent process till full convergence. Instead, in order to obtain some robustness against sampling noise, we may use a fixed number of gradient ascent updates. One particular choice, which is of special interest, is the use of a single gradient ascent update, leading to the updating rule:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s) \tag{16}$$

[8] This is a particular subtype of models, used in HC-ACO [9], without any non-trivial constraints.

which is identical to the SGA update (5). However, as it was already mentioned earlier, the CE method imposes less restrictions on the quality function (e.g., allowing it to change over time), hence the resulting algorithm may be seen as a generalization of SGA.

To conclude, we have shown that if we use (14) as a (possibly approximate) solution of Equation (11), the Hyper-Cube ACO algorithm is derived. If otherwise we use a single-step gradient ascent for solving (11), we obtain a generalization of the SGA update, in which the quality function is allowed to change over time.

## 6 Conclusions

The ant colony optimization metaheuristics attempts to solve a combinatorial problem by repeatedly constructing solutions using locally available pheromones, and updating them, so as to increase the probability of generating good solutions in the future. While the construction process, employed in ACO, is well-defined (and, in fact, is one of the distinctive features of ACO metaheuristic), a considerable amount of freedom remains regarding the exact form of the pheromone update rule.

We have described two general approaches, the SGA and the CE methods, for updating pheromone values for the ACO metaheuristic. These two methods provide systematic, theoretically founded ways for deriving the update rules. Moreover, we have also shown that in many cases the updates used by the two methods are quite similar (or even identical in some cases), and sometimes they coincide with existing ACO updates.

While the newly derived pheromone updates do have a solid theoretical motivation, little can be said *a priori* about their performance as compared to the existing update rules. The empirical evaluation of these methods is the subject of ongoing research.

**Acknowledgments**

Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate. Mark Zlochin is supported through a Training Site fellowship funded by the Improving Human Potential (IHP) programme of the Commission of the European Community (CEC), grant HPRN-CT-2000-00032. Nicolas Meuleau is supported by a Marie Curie fellowship funded by the CEC, grant HPMF-CT-2000-00230. Mauro Birattari is supported by a fellowship from the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. More generally, this work was partially supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided in this paper 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.

## References


1. M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw Hill, London, UK, 1999.
2. G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. *Journal of Artificial Intelligence Research*, 9:317–365, 1998.
3. D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.
4. R. Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.
5. R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In S. Uryasev and P. M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2001.
6. M. Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
7. M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.
8. T. Stützle and H. H. Hoos. The MAX-MIN ant system and local search for the traveling salesman problem. In *Proceedings of ICEC'97 - 1997 IEEE 4th International Conference on Evolutionary Computation*, pages 308–313. IEEE Press, Piscataway, NJ, 1997.
9. C. Blum, A. Roli, and M. Dorigo. HC–ACO: The hyper-cube framework for Ant Colony Optimization. In *Proceedings of MIC'2001 – Meta–heuristics International Conference*, volume 2, pages 399–403, Porto, Portugal, 2001. Also available as technical report TR/IRIDIA/2001-16, IRIDIA, Université Libre de Bruxelles, Brussels, Belgium.
10. N. Meuleau and M. Dorigo. Ant colony optimization and stochastic gradient descent. *Artificial Life*, 2002, in press.
11. S. Kullback. *Information Theory and Statistics*. John Wiley & Sons, New York, NY, 1959.