Delft University of Technology

Delft Center for Systems and Control

Technical report 10-019

Generalized pheromone update for ant colony learning in continuous state spaces*

J. van Ast, R. Babuška, and B. De Schutter

If you want to cite this report, please use the following reference instead:
J. van Ast, R. Babuška, and B. De Schutter, "Generalized pheromone update for ant colony learning in continuous state spaces," *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC 2010)*, Barcelona, Spain, pp. 2617–2624, July 2010.

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.51.19 (secretary)
fax: +31-15-278.66.79
URL: `http://www.dcsc.tudelft.nl`

*This report can also be downloaded via `http://pub.deschutter.info/abs/10_019.html`

Generalized Pheromone Update for Ant Colony Learning in Continuous State Spaces

Jelmer van Ast, Robert Babuška, and Bart De Schutter

***Abstract*—In this paper, we discuss the Ant Colony Learning (ACL) paradigm for non-linear systems with continuous state spaces. ACL is a novel control policy learning methodology, based on Ant Colony Optimization. In ACL, a collection of agents, called ants, jointly interact with the system at hand in order to find the optimal mapping between states and actions. Through the stigmergic interaction by pheromones, the ants are guided by each others experience towards better control policies. In order to deal with continuous state spaces, we generalize the concept of pheromones and the local and global pheromone update rules. As a result of this generalization, we can integrate both crisp and fuzzy partitioning of the state space into the ACL framework. We compare the performance of ACL with these two partitioning methods by applying it to the control problem of swinging-up and stabilizing an under-actuated pendulum.**

I. Introduction

Ant Colony Optimization (ACO) is a metaheuristic for solving combinatorial optimization problems and its key ingredients are the pheromones [1]. These pheromones act as a reinforcing mechanism, stimulating metaphorical ants to search for better solutions in regions of the solution space that are likely to contain the optimal solution. ACO resembles the way real ants in colonies cooperate to find routes from sources of food to the nest.

The most important ACO algorithms are the Ant System (AS) , the Ant Colony System (ACS) , and the MAX-MIN Ant System, which have successfully been applied to various optimization problems [1]. The AS is important as it is the first ACO algorithm, while the ACS and the MAX-MIN Ant System have introduced the concepts of elitism and the local pheromone update rule. The ACS and the MAX-MIN Ant System are amongst the best performing ACO algorithms for combinatorial optimization problems [2], [3]. A survey of industrial applications of ACO is presented in [4]. One of the first real applications of the ACO framework to optimization problems in continuous search spaces is described in [5]. Other work includes the Aggregation Pheromones System in [6] and the Differential Ant-Stigmergy Algorithm in [7].

Ant Colony Learning (ACL) is a paradigm for control policy learning for non-linear systems, based on the principles of ACO . It has been introduced in [8] with crisp state space partitioning and extended in [9] with fuzzy state space partitioning. In this paper, we unify both methods by generalizing the concept of pheromones. Rather than associating a pheromone variable to a state-action pair, generalized pheromones are associated to a state-action pair only to a certain degree. The main advantage of this is that a continuous-valued state can be represented in the ACL framework without introducing non-determinism through discretization. In this paper, we relate crisp and fuzzy state space partitioning in the ACL framework by means of the generalized pheromone concept. We also present a comparison of both methods when applied to the non-linear control problem of swinging up and stabilizing an inverted pendulum. Note that ACL cannot be compared to other ACO techniques, since the application domains of control policy learning and combinatorial optimization are very different.

This paper is structured as follows. In Section II, the ACL paradigm is presented by describing the optimal control policy learning problem, outlining the algorithm, and discussing its most important aspects in more detail. Section III discusses the generalization of the pheromone update rules and the action selection rule to deal with continuous state spaces. It also describes how crisp and fuzzy state space partitioning can be unified in this way. In Section IV the inverted pendulum swing-up and stabilization problem is used to compare both versions of ACL and the effect of the way of partitioning the state space.

Jelmer van Ast, Robert Babuška, and Bart De Schutter are with the Delft Center for Systems and Control of the Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands (email: j.m.vanast@tudelft.nl, r.babuska@tudelft.nl, b.deschutter@tudelft.nl). Bart De Schutter is also with the Marine and Transport Technology Department of the Delft University of Technology.

II. Ant Colony Learning for Optimal Control

A. The Optimal Policy Learning Problem

Assume that we have a nonlinear dynamic system, characterized by a continuous-valued state vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^{\mathrm{T}} \in \mathcal{X} \subseteq \mathbb{R}^n$, with $\mathcal{X}$ the state space and n the order of the system. Also assume that the state can be controlled by an input $\mathbf{u} \in \mathcal{U}$ that can only take a finite number of values and that the state can be measured at discrete time steps, with a sample time T_s with t the discrete time index. The sampled system is denoted as:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)).$$

The optimal control problem we consider is to control the state of the system from any given initial state $\mathbf{x}(0) = \mathbf{x}_0$ to a desired goal state $\mathbf{x}(T) = \mathbf{x}_\mathrm{g}$ in at most T steps and in an optimal way, where optimality is defined by minimizing a certain cost function. As an example, take the following

quadratic cost function:

$$J(s) = J(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) = \sum_{t=0}^{T-1} \mathbf{e}^{\mathrm{T}}(t+1)\mathbf{Q}\mathbf{e}(t+1) + \mathbf{u}^{\mathrm{T}}(t)\mathbf{R}\mathbf{u}(t), \quad (1)$$

with s the solution found by a given ant, $\tilde{\mathbf{x}} = \mathbf{x}(1), \ldots, \mathbf{x}(T)$ and $\tilde{\mathbf{u}} = \mathbf{u}(0), \ldots, \mathbf{u}(T-1)$ respectively the sequences of states and actions in that solution, $\mathbf{e}(t+1) = \mathbf{x}(t+1) - \mathbf{x}_{\mathrm{g}}$ the error at time $t+1$, and $\mathbf{Q}$ and $\mathbf{R}$ positive definite matrices of appropriate dimensions. The problem is to find a nonlinear mapping $\mathbf{u}(t) = \mathbf{h}(\mathbf{x}(t))$ from the state to the input that, when applied to the system in $\mathbf{x}_0$ results in a sequence of state-action pairs $(\mathbf{u}(0), \mathbf{x}(1))$, $(\mathbf{u}(1), \mathbf{x}(2))$, …, $(\mathbf{u}(T-1), \mathbf{x}(T))$ that minimizes this cost function. The function $\mathbf{h}$ is called the control policy and acts as a state feedback controller. At time $t = T$ the interaction with the system is stopped and each ant is either in the goal state ($\mathbf{x}(T) = \mathbf{x}_{\mathrm{g}}$), or has timed out. We make the assumption that $\mathbf{Q}$ and $\mathbf{R}$ are selected in such a way that $\mathbf{x}_{\mathrm{g}}$ can be reached in at most T time steps. The matrices $\mathbf{Q}$ and $\mathbf{R}$ balance the importance of speed versus the aggressiveness of the controller. This kind of cost function is frequently used in optimal control of linear systems, as the optimal controller minimizing the quadratic cost can be derived as a closed expression after solving the corresponding Riccati equation using the $\mathbf{Q}$ and $\mathbf{R}$ matrices and the matrices of the linear state space description [10]. In our case, we aim at finding control policies for non-linear systems, which in general cannot be derived analytically from the system description and the $\mathbf{Q}$ and $\mathbf{R}$ matrices. Note that our method is not limited to the use of quadratic cost functions.

In order to apply ACL , the continuous-valued state must be represented by a finite number of parameters, which is called *discretization*. We define a discrete state $\mathbf{q} \in \mathcal{Q}$. The discretization of a continuous-valued state is denoted by $\mathbf{q} \leftarrow \mathrm{discretize}(\mathbf{x}, \mathcal{Q})$.

B. Outline of the ACL Algorithm

In ACL , the general outline of the algorithm is as follows. The set $\mathcal{C}$ contains the ants that have not yet reached the goal state. Initially, all M ants in this set are distributed randomly over the state space of the system and the pheromone levels $\tau_{\mathbf{qu}}$ associated with each state-action pair $(\mathbf{q}, \mathbf{u})$ are set to an initial value $\tau_{\mathbf{qu}}(0) = \tau_0$. In what is called a *trial*, all ants make interaction steps with the system. First, they decide based on the pheromone levels which action to perform, after which they apply this action to their own copy of the system. They store the state-action pair to their personal record, called their partial solution s_{p} and the pheromone level at that state-action pair $\tau_{\mathbf{qu}}$ is annealed through the local pheromone update. Each copy of the system responds to the input by changing its state $\mathbf{x}$, which is observed by the ants as a change of the discrete state $\mathbf{q}$, after which they repeat the process by choosing a new action until they reach the goal state $\mathbf{q}_{\mathrm{g}}$, which terminates the trial. After all ants have terminated their trial, or have timed out, all partial solutions are added to the multiset $\mathcal{S}_{\mathrm{trial}}$. This set is used in the global pheromone update step, where all solutions in the set are evaluated over the cost function, and the state-action pairs contained in the solutions receive a pheromone update accordingly. Figure 1 illustrates the layout of the algorithm, with the ants interacting in parallel with their copies of the system, while reading from and writing to a common memory of pheromone levels.

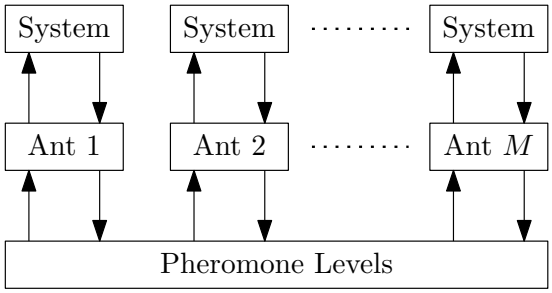


Fig. 1. Schematic overview of the layout of the algorithm. The ants interact in parallel with their copies of the system, while reading from and writing to a common memory of pheromone levels.

The following sections describe the main steps of the algorithm, i.e., the action selection, the local pheromone update, and the global pheromone update. The complete algorithm is given in Algorithm 1. There, the assignment $\mathbf{x}_c \leftarrow \mathrm{random}(\mathcal{X})$ in Step 6 selects for ant c a random state $\mathbf{x}_c$ from the state space $\mathcal{X}$ with a uniform probability distribution. Although the domain $\mathcal{X}$ is listed as an input, in fact the input is any discrete representation of this domain compatible with the $\mathrm{random}(\mathcal{X})$ function.

From the output of the algorithm, which are the pheromone values, the control policy can be derived according to (4). In the following, we explicitly distinguish between the steps in the inner loop and the steps in the outer loop of the algorithm. In the inner loop, the iterations are indexed by t, while in the outer loop, the iterations are indexed by k. In order to understand the timing of the pheromone updates unambiguously, the pheromone variables in the inner loop receive the superscript "local": $\tau^{\mathrm{local}}_{\mathbf{qu}}$. Before starting the inner loop, the current pheromone levels are copied to the local pheromone levels: $\tau^{\mathrm{local}}_{\mathbf{qu}}(0) = \tau_{\mathbf{qu}}(k)$ for all state-action pairs. The first step in the inner loop is the selection of the action.

C. Action Selection

In the action selection step, each ant c determines which action to apply to the system in a given state $\mathbf{q}_c$. Action selection in ACL is done in a similar way as in the AS [1], but without the heuristic variables and within the state-action framework:

$$\mathbf{u}_c \sim \mathbf{p}_c\{\mathbf{u}|\mathbf{q}_c\} = \frac{\left(\tau^{\mathrm{local}}_{\mathbf{q}_c\mathbf{u}}\right)^{\alpha}}{\sum_{\boldsymbol{\ell} \in \mathcal{U}_{\mathbf{q}_c}} \left(\tau^{\mathrm{local}}_{\mathbf{q}_c\boldsymbol{\ell}}\right)^{\alpha}}, \quad \mathbf{u} \in \mathcal{U}_{\mathbf{q}_c} \quad (2)$$

where $\mathbf{p}_c\{\mathbf{u}|\mathbf{q}_c\}$ is the probability for an ant c to choose the action $\mathbf{u}$ in the state $\mathbf{q}_c$ and $\mathcal{U}_{\mathbf{q}_c}$ is the action set available to ant c in state $\mathbf{q}_c$. This action selection rule is called the

Algorithm 1 The Ant Colony Learning algorithm.

Input: $\mathcal{Q}, \mathcal{U}, \mathcal{X}, \mathbf{f}, M, \tau_0, \rho, \gamma, \alpha, T, K$

$k \leftarrow 0$; $\tau_{\mathbf{qu}} \leftarrow \tau_0, \quad \forall(\mathbf{q}, \mathbf{u}) \in \mathcal{Q} \times \mathcal{U}$
repeat
  $t \leftarrow 0$; $\mathcal{S}_{\text{trial}} \leftarrow \emptyset$; $\mathcal{C} \leftarrow \{1, 2, \ldots, M\}$
  for all ants $c \in \mathcal{C}$ in parallel **do**
    $s_{\text{p},c} \leftarrow \emptyset$
    $\mathbf{x}_c(0) \leftarrow \text{random}(\mathcal{X})$
    repeat
      $\mathbf{q}_c(t) \leftarrow \text{discretize}(\mathbf{x}_c(t), \mathcal{Q})$
      Action selection:
      $\mathbf{u}_c(t) \sim \mathbf{p}_c\{\mathbf{u}|\mathbf{q}_c(t)\} = \frac{\tau^{\alpha}_{\mathbf{q}_c\mathbf{u}}}{\sum_{\boldsymbol{\ell} \in \mathcal{U}_{\mathbf{q}_c}} \tau^{\alpha}_{\mathbf{q}_c\boldsymbol{\ell}}}, \mathbf{u} \in \mathcal{U}_{\mathbf{q}_c}$
      $s_{\text{p},c} \leftarrow s_{\text{p},c} \cup \{(\mathbf{q}_c(t), \mathbf{u}_c(t))\}$
      $\mathbf{x}_c(t+1) \leftarrow \mathbf{f}(\mathbf{x}_c(t), \mathbf{u}_c(t))$
      Local pheromone update:
      $\tau_{\mathbf{q}_c\mathbf{u}_c} \leftarrow (1-\gamma)\tau_{\mathbf{q}_c\mathbf{u}_c} + \gamma\tau_0$
      if $\text{discretize}(\mathbf{x}_c(t+1), \mathcal{Q}) = \mathbf{q}_{\text{g}}$ **then**
        $\mathcal{C} \leftarrow \mathcal{C} \setminus \{c\}$
      end if
      $t \leftarrow t + 1$
    until $t = T$ **or** $\mathcal{C} = \emptyset$
    $\mathcal{S}_{\text{trial}} \leftarrow \mathcal{S}_{\text{trial}} \cup \{s_{\text{p},c}\}, \quad \forall c \in \{1, 2, \ldots, M\}$
  end for
  Global pheromone update,
  $\tau_{\mathbf{qu}} \leftarrow (1-\rho)\tau_{\mathbf{qu}} + \rho \sum_{\substack{s \in \mathcal{S}_{\text{trial}}: \\ (\mathbf{q},\mathbf{u}) \in s}} J^{-1}(s),$
  $\forall(\mathbf{q}, \mathbf{u}) : \exists s \in \mathcal{S}_{\text{trial}} : (\mathbf{q}, \mathbf{u}) \in s$
  $k \leftarrow k + 1$
until $k = K$

Output: $\tau_{\mathbf{qu}}, \quad \forall(\mathbf{q}, \mathbf{u}) \in \mathcal{Q} \times \mathcal{U}$

random proportional, or *Boltzmann* action selection rule and the amount of exploration is implicit in the choice of α and the pheromone levels.

D. Local Pheromone Update

The pheromones are initialized equally for all vertices and set to a small positive value τ_0. During every trial, all ants construct their solutions in parallel by interacting with the system until they either have reached the goal state, or the trial exceeds a certain pre-specified number of steps T. After every step, each ant c performs a local pheromone update for the $(\mathbf{q}_c, \mathbf{u}_c)$-pair just visited, similar to the ACS [1]:

$$\tau^{\text{local}}_{\mathbf{q}_c\mathbf{u}_c}(t+1) = (1-\gamma)\tau^{\text{local}}_{\mathbf{q}_c\mathbf{u}_c}(t) + \gamma\tau_0, \tag{3}$$

with $\gamma \in [0, 1)$ the local pheromone decay rate. The purpose of the local pheromone update is to stimulate exploration of the state-action space, by making it less attractive for an ant to choose the same action in a certain state as its predecessor. After the local pheromone update, all ants that have reached the goal are removed from the set $\mathcal{C}$. When this set is empty, or when the inner loop has timed-out (i.e., when $t = T$), the algorithm continues with the global pheromone step in the outer loop.

E. Global Pheromone Update

After completion of the trial (which, let us assume, happens when $t = T$), the pheromone levels are updated according to the following global pheromone update step:

$$\tau_{\mathbf{qu}}(k+1) = (1-\rho)\tau^{\text{local}}_{\mathbf{qu}}(T) + \rho \sum_{\substack{s \in \mathcal{S}_{\text{trial}}(k): \\ (\mathbf{q},\mathbf{u}) \in s}} J^{-1}(s),$$
$$\forall(\mathbf{q}, \mathbf{u}) : \exists s \in \mathcal{S}_{\text{trial}}(k) : (\mathbf{q}, \mathbf{u}) \in s,$$

with $\mathcal{S}_{\text{trial}}$ the multiset of all candidate solutions found in the trial and $\rho \in (0, 1]$ the global pheromone decay rate. This type of update rule is comparable to the AS update rule [1], with the important difference that only the pheromone levels are evaporated that are associated with the elements in the update set of solutions. Note that elitism in the global pheromone update is not possible, since the best solution would then always be the solution starting just prior to the goal state with the action taking the system to the goal state immediately. Since we aim at learning optimal control policies from any initial state, we must also include every solution found in the update. The pheromone deposit is equal to $J^{-1}(s) = J^{-1}(\tilde{\mathbf{q}}, \tilde{\mathbf{u}})$, the inverse of the cost function over the sequence of discretized state-action pairs in s, e.g., according to (1). Note that minimizing the cost corresponds to maximizing the pheromone levels corresponding to the optimal solution. After the global pheromone update, the algorithm continues for an incremented k at the start of the outer loop until the maximal number of trials have taken place (i.e., when $k = K$).

F. Control Policy

The control policy can be extracted from the pheromone levels as follows:

$$\mathbf{u} = \mathbf{h}(\mathbf{q}) = \arg\max_{\boldsymbol{\ell} \in \mathcal{U}_{\mathbf{q}}} (\tau_{\mathbf{q}\boldsymbol{\ell}}), \tag{4}$$

in which ties are broken randomly. This equation means that the control policy assigns the action to a given state that maximizes the associated pheromone levels.

III. Generalization of Pheromone Update Rules

The basic ACL algorithm, as described before, has the disadvantage that it requires the state to be discretized. In this section, we will describe the negative effects of discretization on the performance of the algorithm. We will introduce a generalization of the concept of pheromones, such that these effects can be eliminated.

A. State Space Partitioning

One possibility of applying ACL is to discretize the state $\mathbf{x}$ into a finite number of bins to get the discrete state $\mathbf{q}$. Depending on the sizes and the number of these bins, portions of the state space will be represented with the same discrete state. One can imagine that applying an input to the system that is in a particular discrete state, results in the system to move to a next discrete state with some probability. Fig. 2

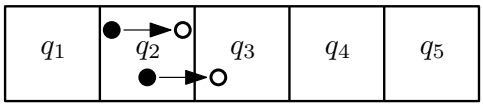


Fig. 2. A simple scenario to illustrate that non-determinism is introduced to the discrete state transitions if the underlying system is continuous. In both cases, the initial state is discretized to the same bin, the same action is applied, but the resulting discrete state is different.

illustrates this non-determinism induced by the discretization of a continuous state space system.

With fuzzy approximation, the domain of each state variable is partitioned using membership functions. We define the membership functions for the state variables to be triangular-shaped, such that the membership degrees for any value of the state on the domain always sum up to one. Only the centers of the membership functions have to be stored. An example of such a fuzzy partitioning is given in Fig. 3.

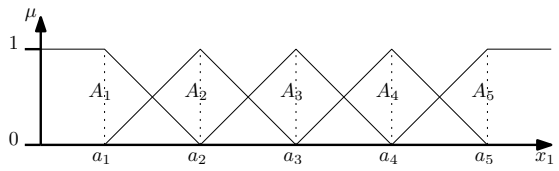


Fig. 3. Membership functions $A_1, \ldots A_5$, with centers $a_1, \ldots, a_5$ on an infinite domain.

Let A_i denote the membership functions for the state variable x_1, with a_i their centers for $i = 1, \ldots, N_A$, with N_A the number of membership functions for x_1. Similarly for x_2, denote the membership functions by B_i, with b_i their centers for $i = 1, \ldots, N_B$, with N_B the number of membership functions for x_2. Likewise, the membership functions can be defined for the other state variables in $\mathbf{x}$, but for the sake of notation, the discussion in this chapter limits the number to two, without loss of generality.

At a discrete time step t, the membership degrees of a specific value of the state to A_i and B_i are denoted by $\mu_{A_i}(x_1(t))$ and $\mu_{B_i}(x_2(t))$ respectively. The membership degree of x_1 to A_i can be computed as follows:

$$\mu_{A_i}(x_1) = \begin{cases} \max\left(0, \min\left(1, \frac{a_2 - x_1}{a_2 - a_1}\right)\right) & \text{if } i = 1 \\ \max\left(0, \min\left(\frac{x_1 - a_{N_A-1}}{a_{N_A} - a_{N_A-1}}, 1\right)\right) & \text{if } i = N_A \\ \max\left(0, \min\left(\frac{x_1 - a_{i-1}}{a_i - a_{i-1}}, \frac{a_{i+1} - x_1}{a_{i+1} - a_i}\right)\right) & \text{otherwise} \end{cases}$$

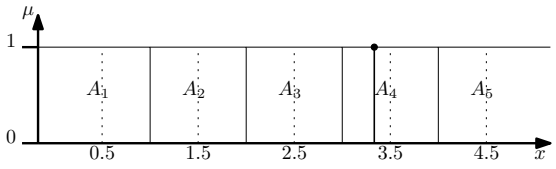
The degree of fulfillment is computed by multiplying the two membership degrees:

$$\beta_{ij}(\mathbf{x}(t)) = \mu_{A_i}(x_1(t)) \cdot \mu_{B_j}(x_2(t)).$$

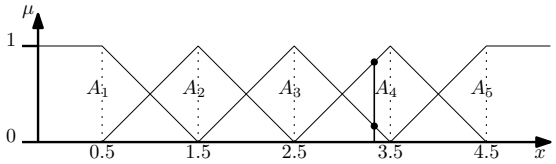
Let the vector of all degrees of fulfillment for a certain state at time t be denoted by:

$$\boldsymbol{\beta}(t) = [\beta_{11}(\mathbf{x}(t)) \;\; \beta_{12}(\mathbf{x}(t)) \;\; \ldots \;\; \beta_{N_A N_B}(\mathbf{x}(t))]^{\mathrm{T}}, \qquad (5)$$

which is a vector containing $\beta_{ij} \in [0, 1]$ for all combinations of i and j, and which elements sum up to one. In order to illustrate what this vector looks like, consider the example from Figure 4. In this example, a one-dimensional state $x = 3.2$ is partitioned with crisp bins and fuzzy membership functions of which the centers are the same.



(a) The crisp representation of $x = 3.2$ is: $\boldsymbol{\beta}(x) = [0, 0, 0, 1, 0]^{\mathrm{T}}$.



(b) The fuzzy representation of $x = 3.2$ is: $\boldsymbol{\beta}(x) = [0, 0, 0.3, 0.7, 0]^{\mathrm{T}}$.

Fig. 4. A simple example of a one-dimensional state x that is partitioned with crisp bins and fuzzy membership functions of which the centers are the same. This illustrates the meaning of $\boldsymbol{\beta}(\mathbf{x})$.

In ACL , it is useful to view the control problem in the context of a graph, in which the states and pheromones are associated with the vertices and the arcs respectively. The control problem can then be regarded as finding the optimal path through the graph. Figure 5 illustrates this.

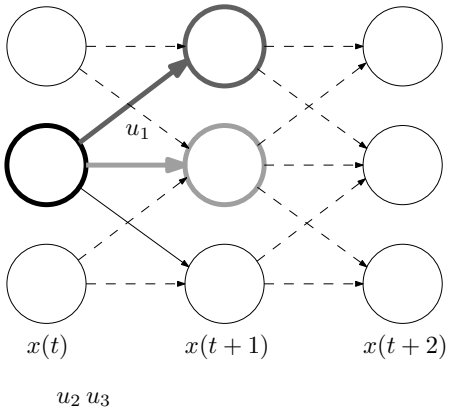


Fig. 5. Graphical representation of a fuzzy state transition. In this figure, at time t, the state is $x(t) = 2$ and action u_1 is chosen, which brings the system to state $x(t+1) = 2.6$. In the graph, the new state is represented by a fuzzy combination of the states 2 and 3.

The operator to partition a state $\mathbf{x}$ using the fuzzy membership functions defined by $\mathcal{Q}$ will be denoted as:

$$\boldsymbol{\beta}(\mathbf{x}) \leftarrow \text{fuzzify}(\mathbf{x}, \mathcal{Q}). \qquad (6)$$

Note that the discretization operator from Algorithm 1 is in fact a special case of (6), as also illustrated in Figure 4. In fuzzy ACL , each element of $\boldsymbol{\beta}$ will be associated to a vertex in the graph. With fuzzy interpolation, there is no artificially

introduced non-determinism in the decision problem, but the transition from vertex to vertex now does not directly correspond to a state transition. The pheromones are associated to the arcs as usual, but the updating needs to take into account the degree of fulfillment of the associated membership functions. An ant is not assigned to a certain vertex at a certain time, but to all vertices at the same time according to some degree of fulfillment. For this reason, a pheromone τ_{ij} is now denoted as $\tau_{i\mathbf{u}}$ with i the index of the vertex (i.e. the corresponding element of $\boldsymbol{\beta}$) and $\mathbf{u}$ the action. Similar to the definition of the vector of all degrees of fulfillment in (5), the vector of all pheromones for a certain action $\mathbf{u}$ at time t is denoted as:

$$\boldsymbol{\tau}_{\mathbf{u}}(t) = [\tau_{1\mathbf{u}}(t) \;\; \tau_{2\mathbf{u}}(t) \;\; \dots \;\; \tau_{N_{AB}\mathbf{u}}(t)]^{\mathrm{T}},$$

where $N_{AB} = N_A \cdot N_B$.

With respect to memory requirements, the crisp representation using $\boldsymbol{\beta}(\mathbf{x})$ always has exactly one non-zero element, which is then by definition equal to one. With pair-wise overlapping normalized membership functions, like the ones shown in Figure 3, the fuzzy representation using $\boldsymbol{\beta}(\mathbf{x})$ has at most 2^d non-zero elements, with d the dimension of the state space. When $\boldsymbol{\beta}(\mathbf{x})$ is stored as a sparse data structure, this means that the memory requirements when using fuzzy partitioning scales exponentially with the number of dimensions. It also means that the memory requirements are independent of the number of membership functions used to represent the state space.

Using $\boldsymbol{\beta}$ and $\boldsymbol{\tau}_{\mathbf{u}}$, we can reformulate the action selection and the local and global pheromone update rules from Section II.

B. Action Selection

The action is chosen randomly according to the following probability distribution:

$$\mathbf{u}_c \sim \mathbf{p}_c\{\mathbf{u}|\boldsymbol{\beta}_c\} = \sum_{i=1}^{N_{AB}} \beta_{c,i} \frac{\left(\tau_{i,\mathbf{u}}^{\mathrm{local}}(t)\right)^{\alpha}}{\sum_{\boldsymbol{\ell}\in\mathcal{U}} \left(\tau_{i,\boldsymbol{\ell}}^{\mathrm{local}}(t)\right)^{\alpha}}. \qquad (7)$$

which weights the pheromones according to the degree of fulfillment of the current state of the ant. Note that when $\boldsymbol{\beta}_c$ contains exactly one 1 and for the rest only zeros, this would correspond to the crisp case, reducing (7) to (2).

C. Local Pheromone Update

The local pheromone update from (3) can be modified to the fuzzy case as follows:

$$\boldsymbol{\tau}_{\mathbf{u}_c}^{\mathrm{local}}(t+1) = (\mathbf{1} - \gamma\boldsymbol{\beta}_c)\boldsymbol{\tau}_{\mathbf{u}_c}^{\mathrm{local}}(t) + (\gamma\boldsymbol{\beta}_c)\tau_0, \qquad (8)$$

where all operations are performed element-wise and $\mathbf{1} = \begin{bmatrix}1 & 1 & \dots & 1\end{bmatrix}^{\mathrm{T}}$ is a vector of ones of proper size. Note that when $\boldsymbol{\beta}_c$ contains exactly one 1 and for the rest only zeros, corresponding to the crisp case, (8) reduces to (3).

It can be derived that after N ants have performed a local pheromone for a given state-action pair, the pheromone vector has been updated according to:

$$\boldsymbol{\tau}_{\mathbf{u}}^{\mathrm{local}} \leftarrow (\boldsymbol{\tau}_{\mathbf{u}}^{\mathrm{local}} - \tau_0\mathbf{1}) \prod_{c=1}^{N} (\mathbf{1} - \gamma\boldsymbol{\beta}_c) + \tau_0\mathbf{1},$$

where again all operations are performed element-wise. This result shows that the final value of the pheromone level is independent of the order in which the local pheromone updates are performed. This is important when ACL is implemented in a serial manner, while the ants are actually presumed to operate in parallel.

D. Global Pheromone Update

In order to derive a fuzzy representation of the global pheromone update step, we introduce the following indicator vectors. The elements of indicator vectors can take a real value from the domain $[0, 1]$. The most basic indicator vector that we need is $\mathcal{I}_{\mathbf{u},s_i^{(j)}}$, which, in the case of crisp state space partitioning has only one element equal to 1, namely the one for $(\mathbf{q}, \mathbf{u}) = s_i^{(j)}$. Since a solution $s_i = \{s_i^{(1)}, s_i^{(2)}, \dots, s_i^{(N_{s_i})}\}$ is an ordered set of solution components $s_i^{(j)} = (\mathbf{q}_j, \mathbf{u}_j)$, another indicator vector, $\mathcal{I}_{\mathbf{u},s_i}$, can be created by taking the union of all $\mathcal{I}_{\mathbf{u},s_i^{(j)}}$:

$$\mathcal{I}_{\mathbf{u},s_i} = \bigcup_{j=1}^{N_{s_i}} \mathcal{I}_{\mathbf{u},s_i^{(j)}}.$$

Let us denote the multiset of solutions as the ordered set of solutions $\mathcal{S}_{\mathrm{trial}} = \{s_1, s_2, \dots, s_{N_{\mathcal{S}_{\mathrm{trial}}}}\}$, we can then create $\mathcal{I}_{\mathbf{u},\mathcal{S}_{\mathrm{trial}}}$ by taking the union of all $\mathcal{I}_{\mathbf{u},s}$:

$$\mathcal{I}_{\mathbf{u},\mathcal{S}_{\mathrm{trial}}} = \bigcup_{i=1}^{N_{\mathcal{S}_{\mathrm{trial}}}} \mathcal{I}_{\mathbf{u},s_i}.$$

In fact, $\mathcal{I}_{\mathbf{u},s_i}$ can be regarded as a representation of the state-action pair $(\mathbf{q}_i, \mathbf{u}_i)$. In order to generalize the global pheromone update step to the fuzzy case, realize that $\boldsymbol{\beta}(\mathbf{x}(t))$ from (5) can be seen as an indicator vector $\mathcal{I}_{\mathbf{u},s_i^{(j)}}$, if combined with an action $\mathbf{u}$. For the union operator, we can take a fuzzy union set operator, such as:

$$(A \cup B)(x) = \max[\mu_A(x), \mu_B(x)],$$

which for a vector operates on its elements. In this operator, A and B are membership functions, x a variable and $\mu_A(x)$ the degree to which x belongs to A. Note that when A maps x to a crisp domain $\{0, 1\}$, the union operator is still valid. Using these notations, we can write the generalized global pheromone update rule as:

$$\begin{aligned}\boldsymbol{\tau}_{\mathbf{u}}(k+1) = &\left\{(1-\rho)\boldsymbol{\tau}_{\mathbf{u}}^{\mathrm{local}}(T) + \rho \sum_{s\in\mathcal{S}_{\mathrm{trial}}} J^{-1}(s)\mathcal{I}_{\mathbf{u},s}\right\} \\ &\cdot \mathcal{I}_{\mathbf{u},\mathcal{S}_{\mathrm{trial}}} + (\mathbf{1} - \mathcal{I}_{\mathbf{u},\mathcal{S}_{\mathrm{trial}}})\boldsymbol{\tau}_{\mathbf{u}}^{\mathrm{local}}(T) \\ =&(\mathbf{1} - \rho\mathcal{I}_{\mathbf{u},\mathcal{S}_{\mathrm{trial}}})\boldsymbol{\tau}_{\mathbf{u}}^{\mathrm{local}}(T) \\ &+ \rho \sum_{s\in\mathcal{S}_{\mathrm{trial}}} J^{-1}(s)\mathcal{I}_{\mathbf{u},s},\end{aligned}$$

where all multiplications are performed element-wise. It can easily be seen that, since the operations involving the solutions from $\mathcal{S}_{\mathrm{trial}}$ are all either unions or sums, the global pheromone update for multiple ants is invariant with respect to the order of the pheromone deposits by the individual ants.

E. Control Policy

Regarding the terminal condition for the ants, with the fuzzy implementation, none of the vertices can be identified as being the terminal vertex. We define a set of membership functions that is used to express the linguistic fuzzy term of the state being *close to the goal*. Specifically, this is satisfied when the membership degree of the state to the membership function with its core equal to the goal state is larger than 0.5. If this has been satisfied, the ant is considered to have terminated its trial.

In order to obtain the control policy for a given state, this state must first be represented by a $\boldsymbol{\beta}$ vector. For each action, the pheromone level is a linear combination of all elements in the pheromone vector and $\boldsymbol{\beta}$. The policy then assigns the action that has the highest pheromone level as follows:

$$\mathbf{u} = \mathbf{h}(\boldsymbol{\beta}) = \arg\max_{\boldsymbol{\ell} \in \mathcal{U}} \left(\sum_{i=1}^{N_{AB}} \beta_i \tau_{i\boldsymbol{\ell}} \right).$$

The following section will apply ACL with both crisp and fuzzy state space partitioning to the non-linear control problem of swinging up and stabilizing an inverted pendulum.

IV. Pendulum Swing-Up and Stabilization

A. Problem Description

The pendulum is modeled as a pole, attached to a pivot point at which a motor exerts a torque. The objective is to get the pendulum from a certain initial position to its unstable upright position, and to keep it stabilized within a certain band around that unstable position. The torque is, however, limited such that it is not possible to move the pendulum to its upright position in one movement. The pendulum problem is a nice abstraction of more complex robot control problems, like the stabilization of a walking humanoid robot. The behavior can be easily analyzed, while the learning problem is challenging. The non-linear state equations of the pendulum are given by:

$$\begin{aligned} \dot{\theta}(t) &= \omega(t) \\ J\dot{\omega}(t) &= K_{\mathrm{m}} u(t) - mgL\sin(\theta(t)) - D\omega(t), \end{aligned}$$

with $\theta(t) = x_1(t)$ and $\omega(t) = x_2(t)$ the state variables, representing the angle and angular velocity of the pole in continuous time respectively. Furthermore, $u(t)$ is the applied torque and the other parameters with their values as used in the simulations are listed in Table I.

TABLE I
The parameters of the pendulum model and their values.

Symbol	Value	Unit	Meaning
J	0.005	$\mathrm{kg}\cdot\mathrm{m}^2$	pendulum inertia
K_{m}	0.1	$-$	motor gain
D	0.01	$\mathrm{kg}\cdot\mathrm{s}^{-1}$	damping
m	0.1	kg	mass
L	0.1	m	pendulum length
g	9.81	$\mathrm{m}\cdot\mathrm{s}^{-2}$	gravitational acceleration

B. Set-up of the Experiment

The system is sampled with a sampling time $T_{\mathrm{s}} = 0.1\mathrm{s}$ and the states are discretized using bins or fuzzy membership functions, the centers of which define the discrete state space:

$$\begin{aligned} \mathcal{Q}_\theta &= \left\{ 0, \frac{2\pi}{N_\theta}, \ldots, \frac{2\pi(N_\theta - 1)}{N_\theta} \right\} \\ \mathcal{Q}_\omega &= \left\{ -\omega_{\max}, -\omega_{\max} + \frac{2\omega_{\max}}{N_\omega - 1}, \ldots, \omega_{\max}, \right\}, \end{aligned}$$

where $N_\theta = 40$ and $N_\omega = 41$ are the number of discretization bins for θ and ω respectively and $\omega_{\max}$ is the maximum (absolute) angular velocity expected to occur. The angle will be observed as $\theta(\mathrm{mod}2\pi)$.

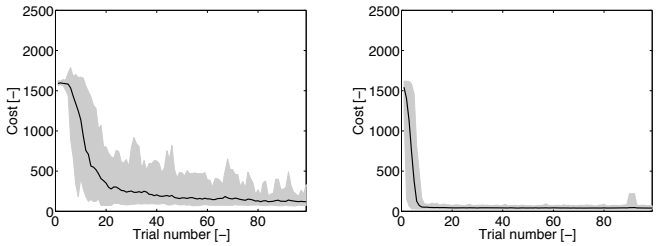
We will compare the performance of crisp and fuzzy ACL . The number of ants is $M = 250$. The global and local pheromone decay rates are $\rho = \gamma = 0.1$. Regarding the action selection, we choose $\alpha = 3$. The initial pheromone level is $\tau_0 = 0.0001$. A trial is stopped after $T = 300$ steps, and the algorithm is stopped after $K = 100$ trials. An ant reaches the goal if its state is $|\pi - \theta| \leq 0.1$ and $|\omega| \leq 0.1$. The quadratic cost function from (1) is used to measure the performance:

$$\begin{aligned} J(s) = J(\tilde{\mathbf{x}}, \tilde{\mathbf{u}}) &= \sum_{t=0}^{T-1} \mathbf{e}^{\mathrm{T}}(t+1)\mathbf{Q}\mathbf{e}(t+1) + \mathbf{R}u^2(t), \\ \mathbf{Q} = \begin{bmatrix} 1 & 0 \\ 0 & 0.1 \end{bmatrix}, &\quad \mathbf{R} = 0.05. \end{aligned}$$

The goal state is $\mathbf{x}_{\mathrm{g}} = \begin{bmatrix} \pi & 0 \end{bmatrix}^{\mathrm{T}}$. The action set contains 3 actions, namely $\mathcal{U} = \{-0.8, 0, 0.8\}[\mathrm{Nm}]$, which will be referred to as full negative torque, zero torque, and full positive torque respectively. The experiments are carried out 30 times. The plots will show the average performance and the min-max area of worst-case and best-case performance.

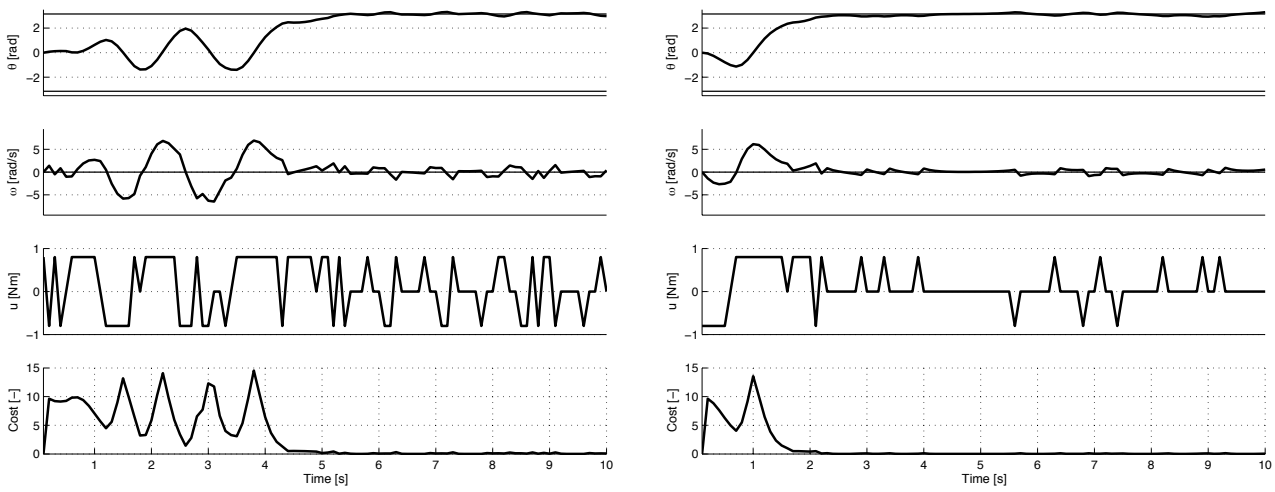
C. Results

After each trial, the system is simulated using the current control policy and a set of initial states. The resulting state trajectories are evaluated over the cost function and the average of these costs is recorded. This is what we call the cost of the policy. Fig. 6 shows the evolution of the cost of the policy over the trials.



(a) Crisp partitioning. (b) Fuzzy partitioning.

Fig. 6. The cost of the control policy as a function of the number of trials passed since the start of the experiment. The black line in each plot is the average cost over 30 experiments and the gray area represents the range of costs for these experiments.



(a) Crisp partitioning.

(b) Fuzzy partitioning.

Fig. 7. The inverted pendulum controlled from an initial state $\mathbf{x}_0 = [0 \quad 0]^{\mathrm{T}}$ to the goal state $\mathbf{x}_{\mathrm{g}} = [\pi \quad 0]^{\mathrm{T}}$ by the resulting policy derived in these experiments with crisp and fuzzy state space partitioning respectively. The top two graphs show the trajectories of the two states, while the third graph shows the input, and the bottom graph shows the cost.

There is a clear difference between the results of ACL with crisp partitioning and ACL with fuzzy partitioning. With fuzzy partition, the cost of the policy decreases rapidly and remains at approximately the same value for about the remainder of the trials. There is a little bump near the end of the algorithm, indicating that it remains possible for the policy to change, which is due to exploration. It is clearly observed that the results for the 30 experiments are very similar, indicating a very good repeatability of the experiment and a predictable behavior of the algorithm.

With crisp partitioning, the cost of the policy decreases slower, with much more variation over the 30 experiments. The average of the cost at the end of the algorithm is also a little higher than that of ACL with fuzzy partitioning. This means that ACL with crisp partitioning requires more trials, is less predictable, and results in a less optimal solution compared to ACL with fuzzy partitioning.

In Fig. 7, the best policy found at the end of the experiments is used to control the pendulum from the downright position to the upright position. It can be seen that the policy resulting from ACL with fuzzy partitioning controls the pendulum more than two times faster than the policy resulting from ACL with crisp partitioning. There is also less chattering of the pendulum near the goal. This is because the control policy with fuzzy partitioning is a continuous mapping from the state space to the action space, whereas crisp partitioning results in a discrete mapping.

V. Conclusions and Recommendations

In this paper, we have discussed the application of ACL in continuous state spaces. The purpose of ACL is to learn control policies for systems with non-linear dynamics by interacting with the system. Based on the same principles as ACO , the ants decide on their actions using pheromone levels. We have presented a generalization of the concept of pheromones, enabling it to be applied in continuous domains.

We have discussed two ways of partitioning the continuous state space. With crisp partitioning, each partitioning bin represents a range of continuous states, while with fuzzy partitioning, the continuity of the state space is preserved by using fuzzy membership functions. The implication of this difference is that crisp partitioning introduces discretization noise in the state transitions observed by the ants, while with fuzzy partitioning this is not the case.

Both ACL algorithms have been applied to the control problem of swinging up and stabilizing an inverted pendulum. With fuzzy partitioning, during the algorithm, the cost of the policy decreased faster, in a more predictable way, and to a lower value compared to ACL with crisp partitioning. The reason for this difference is that the discretization noise makes it much less predictable to what state trajectories a particular policy leads. Fuzzy partitioning does not have this drawback and enables ACL to learn a near optimal control policy in under 10 trials, which is really fast. In practice, such fast learning is achieved when the ants are implemented in parallel. We recommend future research to focus on the parallel implementation of ACL .

Acknowledgment

This research is financially supported by Senter, Ministry of Economic Affairs of The Netherlands within the BSIK-ICIS project “Self-Organizing Moving Agents” (grant no. BSIK03024)

References

[1] M. Dorigo and T. Stützle, *Ant Colony Optimization.* Cambridge, MA, USA: The MIT Press, 2004.

[2] M. Dorigo and L. Gambardella, “Ant Colony System: a cooperative learning approach to the traveling salesman problem,” *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.

[3] T. Stützle and U. Hoos, "MAX-MIN Ant Systems," *Journal of Future Generation Computer Systems*, vol. 16, pp. 889–914, 2000.
[4] B. Fox, W. Xiang, and H. P. Lee, "Industrial applications of the ant colony optimization algorithm," *International Journal of Advanced Manufacturing Technology*, vol. 31, no. 7-8, pp. 805–814, 2007.
[5] K. Socha and C. Blum, "An ant colony optimization algorithm for continuous optimization: application to feed-forward neural network training," *Neural Computing & Applications*, vol. 16, no. 3, pp. 235–247, May 2007.
[6] S. Tsutsui, M. Pelikan, and A. Ghosh, "Performance of aggregation pheromone system on unimodal and multimodal problems," in *Proceedings of the 2005 Congress on Evolutionary Computation (CEC 2005)*, vol. 1, Edinburgh, Scotland, September 2005, pp. 880–887.
[7] P. Korosec, J. Silc, K. Oblak, and F. Kosel, "The differential ant-stigmergy algorithm: an experimental evaluation and a real-world application," in *Proceedings of the 2007 Congress on Evolutionary Computation (CEC 2007)*, Singapore, September 2007, pp. 157–164.
[8] J. M. van Ast, R. Babuška, and B. De Schutter, "Novel ant colony optimization approach to optimal control," *International Journal of Intelligent Computing and Cybernetics*, vol. 2, no. 3, pp. 414–434, 2009.
[9] ——, "Fuzzy ant colony optimization for optimal control," in *Proceedings of the American Control Conference (ACC 2009)*, Saint Louis, MO, USA, June 2009, pp. 1003–1008.
[10] K. J. Åström and B. Wittenmark, *Computer Controlled Systems – Theory and Design*. Englewood Cliffs, NJ: Prentice-Hall, January 1990.