Technical report 09-009

Novel ant colony optimization approach to optimal control*

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

Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft
The Netherlands
phone: +31-15-278.24.73 (secretary)
URL: `https://www.dcsc.tudelft.nl`

[*] This report can also be downloaded via `https://pub.deschutter.info/abs/09_009.html`

NOVEL ANT COLONY OPTIMIZATION APPROACH TO OPTIMAL CONTROL

Jelmer Marinus van Ast
Delft Center for Systems and Control, Delft University of Technology
Mekelweg 2, 2628 CD Delft, The Netherlands
j.m.vanast@tudelft.nl
http://www.dcsc.tudelft.nl/˜jvanast

Robert Babuška
Delft Center for Systems and Control, Delft University of Technology
Mekelweg 2, 2628 CD Delft, The Netherlands
r.babuska@tudelft.nl
http://www.dcsc.tudelft.nl/˜rbabuska

Bart De Schutter
Delft Center for Systems and Control & Marine and Transport Technology
Delft University of Technology, Mekelweg 2, 2628 CD Delft, The Netherlands
b@deschutter.info
http://www.dcsc.tudelft.nl/˜bdeschutter

Purpose - In this paper, a novel Ant Colony Optimization (ACO) approach to optimal control is proposed. The standard ACO algorithms have proven to be very powerful optimization metaheuristic for combinatorial optimization problems. They have been demonstrated to work well when applied to various NP-complete problems, such as the traveling salesman problem. In this paper, ACO is reformulated as a model-free learning algorithm and its properties are discussed.
Design/methodology/approach - First, it is described how quantizing the state space of a dynamic system introduces stochasticity in the state transitions and transforms the optimal control problem into a stochastic combinatorial optimization problem, motivating the ACO approach. The algorithm is presented and is applied to the time-optimal swing-up and stabilization of an underactuated pendulum. In particular, the effect of different numbers of ants on the performance of the algorithm is studied.
Findings - The simulations show that the algorithm finds good control policies reasonably fast. An increasing number of ants results in increasingly better policies. The simulations also show that although the policy converges, the ants keep on exploring the state space thereby capable of adapting to variations in the system dynamics.
Research limitations/implications - This research introduces a novel ACO approach to optimal control and as such marks the starting point for more research of its properties. In particular, quantization issues must be studied in relation to the performance of the algorithm.
Originality/value - The work presented is original as it presents the first application of ACO to optimal control problems.

Keywords: ant colony optimization; optimal control; combinatorial optimization; underactuated pendulum

Papertype: **Research Paper**

1. Introduction

Ant Colony Optimization (ACO) is a metaheuristic for solving combinatorial optimization problems [Dorigo and Blum (2005)]. Inspired by ants and their behavior in finding shortest paths from their nest to sources of food, the virtual ants in ACO aim at jointly finding optimal paths in a given search space. The key ingredients in ACO are the pheromones. With real ants, these are chemicals deposited by the ants and their concentration encodes a map of trajectories, where stronger concentrations represent the trajectories that are more likely to be optimal. In ACO, the ants read and write values to a common pheromone matrix. Each ant autonomously decides on its actions biased by these pheromone values. This indirect form of communication is called stigmergy. Over time, the pheromone matrix converges to encode the optimal solution of the combinatorial optimization problem, but the ants typically do not all converge to this solution, thereby allowing for constant exploration and the ability to adapt the pheromone matrix to changes in the problem structure. These characteristics have resulted in a strong increase of interest in ACO over the last decade since its introduction in the early nineties [Colorni *et al.* (1992)].

The Ant System (AS), which is the basic ACO algorithm, and its variants, have successfully been applied to various optimization problems, such as the traveling salesman problem [Dorigo and Stützle (2004)], load balancing [Sim and Sun (2003)], job shop scheduling [Huang and Yang (2008); Alaykran *et al.* (2007)], optimal path planning for mobile robots [Fan *et al.* (2003)], and telecommunication routing [Wang *et al.* (2009)]. An implementation of the ACO concept of pheromone trails for real robotic systems is described in [Purnamadjaja and Russell (2005)]. A survey of industrial applications of ACO is presented in [Fox *et al.* (2007)]. A survey of ACO and other metaheuristics to stochastic combinatorial optimization problems can be found in [Bianchi *et al.* (2006)].

This paper introduces a novel method for applying an ACO algorithm to the design of optimal control policies for continuous-time, continuous-state dynamic systems and extends the original contribution by the authors of this paper in [van Ast *et al.* (2008)]. In that paper, the continuous model of the system to be controlled was transformed to a stochastic discrete automaton after which a variation of the Ant System was applied to derive the control policy. In the current paper, the algorithm is directly applied to the equations of the system dynamics and the transformation to a stochastic automaton is merely used as a way of analyzing and discussing the performance of the algorithm. [Birattari *et al.* (2002)] is the first work linking ACO to optimal control. Although presenting a formal framework, called *ant programming*, they neither apply it, nor do they study its performance. Our algorithm shares some similarities with a particular reinforcement learning algorithm, Q-learning. Earlier work by Gambardella and Dorigo [1995] introduced the Ant-Q algorithm, which is the most notable other work relating ACO with Q-learning. However, Ant-Q has been developed for combinatorial optimization problems and not to optimal control problems. Because of this difference, our ACO approach for an optimal control algorithm is novel in all major structural aspects of the Ant-Q algorithm, namely the choice of the action selection method, the absence of the heuristic variable, and the choice of the update set of ants. Furthermore, our algorithm is very straightforward to apply, as there are fewer parameters that need to be set a priori, compared to Ant-Q and the well-known benchmark ACO algorithms. The effectiveness of our method is demonstrated by applying it to the control task of swinging up and stabilizing an underactuated pendulum. The pendulum problem is a nice abstraction of more complex robot control problems, like the stabilization of a walking humanoid robot. Finding the optimal control policy by interacting with the system is a challenging task. The results show that a near optimal controller is found quickly with the proposed ACO method. We study the influence of the number of ants on the convergence of the policy for this particular optimal control problem.

The remainder of this paper is structured as follows. Section 2 reviews the basic ACO heuristic and two of its variants, namely the Ant System and the Ant Colony System. Section 3 formalizes the type of control problems we aim to solve, discusses the stochasticity introduced by quantizing the continuous dynamics of the system to be controlled, and presents our ACO algorithm for optimal control problems. In Section 4, the underactuated pendulum swing-up and stabilization problem is formalized including a discussion of the parameters used in our algorithm and the results of applying our algorithm to this control problem with a special focus on the influence of the number of ants. Section 5 concludes this paper.

2. Ant Colony Optimization

2.1. *Framework for ACO Algorithms*

ACO algorithms have been developed to solve hard combinatorial optimization problems [Dorigo and Blum (2005)]. A combinatorial optimization problem can be represented as a tuple $P = \langle \mathcal{S}, F \rangle$, where $\mathcal{S}$ is the solution space with $s \in \mathcal{S}$ a specific candidate solution and where $F : \mathcal{S} \rightarrow \mathbb{R}_+$ is a fitness function assigning strictly positive values to candidate solutions, where higher values correspond to better solutions. The purpose of the algorithm is to find a solution $s^* \in \mathcal{S}$, or set of solutions $\mathcal{S}^* \subseteq \mathcal{S}$ that maximize the fitness function. The solution s^* is then called an optimal solution and $\mathcal{S}^*$ is called the set of optimal solutions.

In ACO, the combinatorial optimization problem is represented by a graph consisting of a set of vertices and a set of arcs connecting the vertices. A particular solution s is a concatenation of solution components (i, j), which are pairs of a vertex i and an arc that connects this vertex to another vertex j. The concatenation of solution components forms a path from the initial vertex to the terminal vertex. How the terminal vertices are defined depends on the problem considered.

For instance, in the traveling salesman problem[a], there are multiple terminal vertices, namely for each ant the terminal vertex is equal to its initial vertex, after visiting all other vertices exactly once. For the application to control problems, as considered in this paper, the terminal vertex corresponds to the desired state of the system. Two values are associated with the arcs: a pheromone trail variable τ_{ij} and a heuristic variable η_{ij}. The pheromone trail represents the acquired knowledge about the optimal solutions over time and the heuristic variable provides a priori information about the quality of the solution component, i.e., the quality of moving from a vertex i to a vertex j. In the case of the traveling salesman problem, the heuristic variables typically represent the inverse of the distance between the respective pair of cities. In general, a heuristic variable represents a short-term quality measure of the solution component, while the task is to acquire a concatenation of solution components that overall form an optimal solution. A pheromone variable, on the other hand, encodes the measure of the long-term quality of concatenating the respective solution component.

2.2. *Ant Systems and Ant Colony Systems*

The most basic ACO algorithm is called the Ant System (AS) and works as follows. A set of M ants is randomly distributed over the vertices. The heuristic variables η_{ij} are set to encode the prior knowledge by favoring the choice of some vertices over others. For each ant c, the partial solution $s_{\mathrm{p},c}$ is initially empty and all pheromone variables are set to some initial value τ_0. In each iteration, each ant decides based on some probability distribution, which solution component (i,j) to add to its partial solution $s_{\mathrm{p},c}$. The probability $p_c\{j|i\}$ for an ant c on a vertex i to move to a vertex j within its feasible neighborhood $\mathcal{N}_i$ is defined as:

$$p_c\{j|i\} = \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{l\in\mathcal{N}_i}\tau_{il}^{\alpha}\eta_{il}^{\beta}}, \forall j \in \mathcal{N}_i, \tag{1}$$

with $\alpha \geq 1$ and $\beta \geq 1$ determining the relative importance of η_{ij} and τ_{ij} respectively. The feasible neighborhood $\mathcal{N}_i$ is the set of not yet visited vertices that are connected to the vertex i. By moving from vertex i to vertex j, ant c adds the associated solution component (i,j) to its partial solution $s_{\mathrm{p},c}$ until it reaches its terminal vertex and completes its candidate solution.

The candidate solutions of all ants are evaluated using the fitness function $F(s)$ and the resulting value is used to update the pheromone levels by:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \sum_{s\in\mathcal{S}_{\mathrm{upd}}}\Delta\tau_{ij}(s), \tag{2}$$

[a] In the traveling salesman problem, there is a set of cities connected by roads of different lengths and the problem is to find the sequence of cities that takes the traveling salesman to all cities, visiting each city exactly once and bringing him back to its initial city with a minimum length of the tour.

with $\rho \in (0, 1)$ the evaporation rate and $\mathcal{S}_{\text{upd}}$ the set of solutions that are eligible to be used for the pheromone update, which will be explained further on in this section. This update step is called the *global* pheromone update step. The pheromone deposit $\Delta\tau_{ij}(s)$ is computed as:

$$\Delta\tau_{ij}(s) = \begin{cases} F(s) & \text{, if } (i,j) \in s \\ 0 & \text{, otherwise.} \end{cases}$$

The pheromone levels are a measure of how desirable it is to add the associated solution component to the partial solution. In order to incorporate forgetting, the pheromone levels decrease by some factor in each iteration. This is called pheromone evaporation in correspondence to the physical evaporation of the chemical pheromones with real ant colonies. By evaporation, it can be avoided that the algorithm prematurely converges to suboptimal solutions. Note that in (2) the pheromone level on all vertices is evaporated and only those vertices that are associated with the solutions in the update set receive a pheromone deposit.

In the following iteration, each ant repeats the previous steps, but now the pheromone levels have been updated and can be used to make better decisions about which vertex to move to. After some stopping criterion has been reached, the values of τ and η on the graph encode the solution, in which the optimal (i, j) pairs are defined in the following way:

$$(i,j): \quad j = \arg\max_{l}(\tau_{il}), \quad \forall i.$$

Note that all ants are still likely to follow suboptimal trajectories through the graph, thereby exploring constantly the solution space and keeping the ability to adapt the pheromone levels to changes in the problem structure.

There exist various rules to construct $\mathcal{S}_{\text{upd}}$, of which the most standard one is to use all the candidate solutions found in the trial $\mathcal{S}_{\text{trial}}$[b]. This update rule is typical for the Ant System (AS) [Dorigo *et al.* (1996)]. However, other update rules have shown to outperform the AS update rule in various combinatorial optimization problems. Rather than using the complete set of candidate solutions from the last trial, either the best solution from the last trial, or the best solution since the initialization of the algorithm can be used. The former update rule is called *Iteration Best* in the literature (which should be called *Trial Best* in our terminology), and the latter is called *Best-So-far*, or *Global Best* in the literature [Dorigo and Blum (2005)]. These methods result in a strong bias of the pheromone trail reinforcement towards solutions that have been proven to perform well and additionally reduce the

[b] In ACO literature, the term trial is seldom used. It is rather a term from the reinforcement learning (RL) community [Sutton and Barto (1998)]. In our opinion it is also a more appropriate term for ACO, especially in the setting of optimal control, and we will use it to denote the part of the algorithm from the initialization of the ants over the state space until the global pheromone update step. The corresponding term for a trial in the ACO literature is iteration and the set of all candidate solutions found in each iteration is denoted as $\mathcal{S}_{\text{iter}}$. In this paper, equivalently to RL, we prefer to use the word iteration to indicate one interaction step with the system.

computational complexity of the algorithm. As the risk exists that the algorithm prematurely converges to suboptimal solutions, these methods are only superior to AS if extra measures are taken to prevent this. Two of the most successful ACO variants in practice that implement the update rules mentioned above, are the Ant Colony System (ACS) [Dorigo and Gambardella (1997)] and the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System [Stützle and Hoos (2000)].

An important element from the ACS algorithm, which acts as a measure to avoid premature convergence to suboptimal solutions is the local pheromone update step, which occurs after each iteration with the trials and is defined as follows:

$$\tau_{ij} \leftarrow (1-\gamma)\tau_{ij} + \gamma\tau_0, \tag{3}$$

where $\gamma \in (0,1)$ is a parameter similar to ρ, ij is the solution component (i,j) just added, and τ_0 is the initial value of the pheromone trail. The effect of (3) is that during the trial visited solution components are made less attractive for other ants to take, in that way promoting the exploration of other, less frequently visited, solution components. In this paper, the ACO for optimal control algorithm is based on the AS combined with the local pheromone update step of the ACS algorithm.

3. ACO for Optimal Control

3.1. *The Optimal Control 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 k the discrete time index. The sampled system is denoted as:

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)). \tag{4}$$

The optimal control problem is to control the state of the system from any given initial state $\mathbf{x}(0) = \mathbf{x}_0$ to a desired goal state $\mathbf{x}(K) = \mathbf{x}_{\mathrm{g}}$ in an optimal way, where optimality is defined by minimizing the following quadratic cost function:

$$J = \sum_{k=0}^{K-1} \mathbf{e}^{\mathrm{T}}(k+1)\mathbf{Q}\mathbf{e}(k+1) + \mathbf{u}^{\mathrm{T}}(k)\mathbf{R}\mathbf{u}(k), \tag{5}$$

with $\mathbf{e}(k+1) = \mathbf{x}(k+1) - \mathbf{x}_{\mathrm{g}}$ the error at time $k+1$ and $\mathbf{Q}$ and $\mathbf{R}$ positive definite matrices of appropriate dimensions. The problem is to find a nonlinear mapping from the state to the input $\mathbf{u}(k) = \mathbf{g}(\mathbf{x}(k))$ that, when applied to the system in $\mathbf{x}_0$ results in a sequence of state-action pairs $(\mathbf{u}(0), \mathbf{x}(1))$, $(\mathbf{u}(1), \mathbf{x}(2))$, …, $(\mathbf{u}(K-1), \mathbf{x}_{\mathrm{g}})$ that minimizes this cost function. The function $\mathbf{g}$ is called the control policy. We make the assumption that $\mathbf{x}_{\mathrm{g}}$ is reachable in at most K time steps. If this is not the case, a minimal cost J will not correspond to a sequence of state-action pairs that ends in $\mathbf{x}_{\mathrm{g}}$. The quadratic cost function in (5) is minimized if the goal is both reachable and reached in minimum time respecting the dynamics of the system in (4) and the restrictions on the size of the input. The matrices **Q** and **R** balance the importance of speed versus the aggressiveness of the controller. This kind of cost function is frequently used in the 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 **Q** and **R** matrices and the matrices of the linear state space description [Åström and Wittenmark (1990)]. 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 **Q** and **R** matrices. Note that our method is not limited to the use of quadratic cost functions. Figure 1 presents a schematic layout of the ACO algorithm for control problems. Note that each ant interacts with a copy of the system and contributes to the learning of one joint control policy. The details of the algorithm are described in Section 3.4.

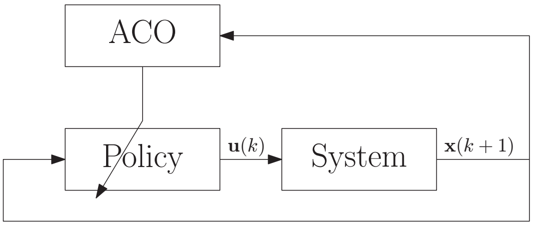


Fig. 1. Layout of the ACO control scheme.

3.2. *Markov Decision Processes*

The control policy we aim to find with our ACO algorithm will be a reactive policy, meaning that it will define a mapping from states to actions without the need of storing the states (and actions) of previous time steps. This poses the requirement on the system that it can be described by a state-transition mapping for a quantized state **q** and an action (or input) **u** in discrete time as follows:

$$\mathbf{q}(k+1) \sim p(\mathbf{q}(k), \mathbf{u}(k)), \tag{6}$$

with p a probability distribution function over the state-action space. In this case, the system is said to be a Markov Decision Process (MDP) and the probability distribution function p is said to be the Markov model of the system. Our ACO approach for optimal control problems requires an MDP. This means that the state vector **q** should be large enough to represent an accurate Markov model of the system. Most reinforcement learning algorithms are also constructed to deal with MDPs [Sutton and Barto (1998)].

Note that finding an optimal control policy for an MDP is equivalent to finding the optimal sequence of state-action pairs from any given initial state to a certain goal state, which is a combinatorial optimization problem. When the state transitions are stochastic, like in (6), it is a stochastic combinatorial optimization problem. As ACO algorithms are especially applicable to (stochastic) combinatorial optimization problems, the application to deriving control policies is evident.

3.3. *Stochasticity Through Quantization*

Assume a continuous-time, continuous-state system and sampled with a sample time T_s. In order to apply ACO, the state $\mathbf{x}$ must be quantized into a finite number of bins to get the quantized state $\mathbf{q}$. Depending on the sizes and the number of these bins, portions of the state space will be represented with the same quantized state. One can imagine that applying an input to the system that is in a particular quantized state results in the system to move to a next quantized state with some probability. In order to illustrate these aspects of the quantization, we consider the continuous model to be cast as a discrete stochastic *automaton*.

Definition 3.1. *An automaton is defined by the triple* $\Sigma = (\mathcal{Q}, \mathcal{U}, \phi)$*, with*

- $\mathcal{Q}$*: a finite or countable set of discrete states,*
- $\mathcal{U}$*: a finite or countable set of discrete inputs, and*
- $\phi : \mathcal{Q} \times \mathcal{U} \times \mathcal{Q} \to [0, 1]$*: a state transition function.*

Given a discrete state vector $\mathbf{q} \in \mathcal{Q}$, a discrete input symbol $\mathbf{u} \in \mathcal{U}$, and a discrete next state vector $\mathbf{q}' \in \mathcal{Q}$, the (Markovian) state transition function ϕ defines the probability of this state transition, $\phi(\mathbf{q}, \mathbf{u}, \mathbf{q}')$, making the automaton stochastic. The probabilities over all states $\mathbf{q}'$ must each sum up to one for each state-action pair $(\mathbf{q}, \mathbf{u})$. An example of a stochastic automaton is given in Figure 2. In this figure, it is clear that, e.g., applying an action $u = 1$ to the system in $q = 1$ can move the system to a next state that is either $q = 1$ with a probability of 0.2, or $q = 2$ with a probability of 0.8.

The probability distribution function determining the transition probabilities reflects the system dynamics and the set of possible control actions is reflected in the structure of the automaton. The probability distribution function can be estimated from simulations of the system over a fine grid of pairs of initial states and inputs, but for the application of ACO, this is not necessary. The ACO algorithm can directly interact with the continuous-state dynamics of the system, as will be described in the following section.

3.4. *ACO Algorithm for Optimal Control*

At the start of every trial, each ant is initialized with a random continuous-valued state of the system. This state is quantized onto $\mathcal{Q}$. Each ant has to determine

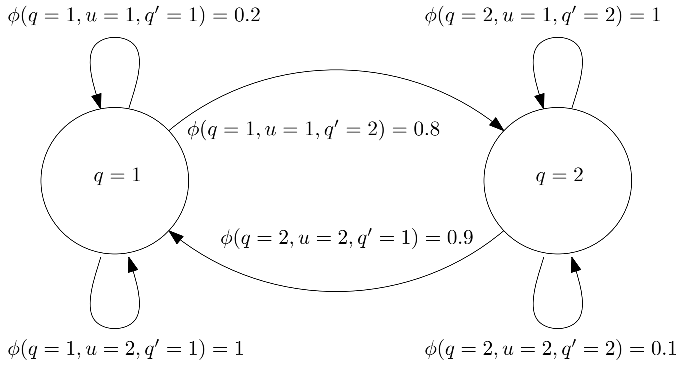


Fig. 2. An example of a stochastic automaton.

which action to choose from. It is not known to the ant to which state this action will take him, as in general there is a set of next states to which the ant can move, according to the system dynamics. Each ant adds the state-action pair to its partial solutions.

No heuristic values are associated with the vertices, as we assume in this paper that no a priori information is available about the quality of solution components. This is implemented by setting all heuristic values to one. It can be seen that η_{ij} disappears from (1) in this case. Only the value of α remains as a tuning parameter, now in fact only determining the amount of exploration as higher values of α make the probability higher for choosing the action associated with the largest pheromone level. In the Ant Colony System, there is an explicit exploration-exploration step when the next node j associated with the highest value of $\tau_{ij}^{\alpha}\eta_{ij}^{\beta}$ is chosen with some probability ϵ (exploitation) and a random action is chosen with the probability $(1-\epsilon)$ according to (1) (exploration). In our algorithm, as we do not have the heuristic factor η_{ij}, the amount of exploration over exploitation can be tuned by setting the value of α as mentioned before. For this reason, we do not include an explicit exploration-exploitation step based on ϵ. Without this and without the heuristic factor, the algorithm needs less tuning and is easier to apply.

The probability of an ant c to choose an action $\mathbf{u}$ when in a state $\mathbf{q}_c$ is:

$$p_c\{\mathbf{u}|\mathbf{q}_c\} = \frac{\tau_{\mathbf{q}_c\mathbf{u}}^{\alpha}}{\sum_{\mathbf{l}\in\mathcal{U}_{\mathbf{q}_c}} \tau_{\mathbf{q}_c\mathbf{l}}^{\alpha}}. \tag{7}$$

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 iterations $K_{\max}$. After every iteration, the ants perform a local pheromone update, equal to (3), but in the setting of our ACO for optimal control framework:

$$\tau_{\mathbf{q}_c\mathbf{u}_c}(k+1) \leftarrow (1-\gamma)\tau_{\mathbf{q}_c\mathbf{u}_c}(k) + \gamma\tau_0.$$

After completion of the trial, the pheromone levels are updated according to the following global pheromone update step:

$$\tau_{\mathbf{qu}}(\kappa+1) \leftarrow (1-\rho)\tau_{\mathbf{qu}}(\kappa) + \rho \sum_{\substack{s \in \mathcal{S}_{\text{trial}};\\ (\mathbf{q},\mathbf{u}) \in s}} J^{-1}(s), \quad \forall (\mathbf{q},\mathbf{u}); \exists s \in \mathcal{S}_{\text{trial}}; (\mathbf{q},\mathbf{u}) \in s,$$

with $\mathcal{S}_{\text{trial}}$ the set of all candidate solutions found in the trial and κ the trial counter. This type of update rule is comparable to the AS update rule, with the important difference that only the pheromone levels are evaporated that are associated with the elements in the update set of solutions. The pheromone deposit is equal to $J^{-1}(s)$, the inverse of the cost function over the sequence of state-action pairs in s according to (5).

The complete algorithm is given in Algorithm 1. In this algorithm the function $\mathbf{x}_c \leftarrow \text{random}(\mathcal{X})$ selects for an ant c a random state $\mathbf{x}_c$ from the state space $\mathcal{X}$ with a uniform probability distribution. The function $\mathbf{q}_c \leftarrow \text{quantize}(\mathbf{x}_c)$ quantizes for an ant c the state $\mathbf{x}_c$ into a quantized state $\mathbf{q}_c$ using the prespecified quantization bins from $\mathcal{Q}$. A flow diagram of the algorithm is presented in Figure 3.

Algorithm 1 The ACO algorithm for optimal control problems.

```
$\kappa \leftarrow 0$
$\tau_{\mathbf{qu}} \leftarrow \tau_0, \quad \forall (\mathbf{q},\mathbf{u}) \in \mathcal{Q} \times \mathcal{U}$
repeat
  $k \leftarrow 0$; $\mathcal{S}_{\text{trial}} \leftarrow \emptyset$
  for all ants $c$ in parallel do
    $s_{\text{p},c} \leftarrow \emptyset$
    $\mathbf{x}_c \leftarrow \text{random}(\mathcal{X})$
    repeat
      $\mathbf{q}_c \leftarrow \text{quantize}(\mathbf{x}_c)$
      $\mathbf{u}_c \sim p_c\{\mathbf{u}|\mathbf{q}_c\} = \frac{\tau^{\alpha}_{\mathbf{q}_c\mathbf{u}}}{\sum_{\mathbf{l} \in \mathcal{U}_{\mathbf{q}_c}} \tau^{\alpha}_{\mathbf{q}_c\mathbf{l}}}, \quad \forall \mathbf{u} \in \mathcal{U}_{\mathbf{q}_c}$
      $s_{\text{p},c} \leftarrow s_{\text{p},c} \bigcup \{(\mathbf{q}_c,\mathbf{u}_c)\}$
      $\mathbf{x}_c(k+1) \leftarrow f(\mathbf{x}_c(k),\mathbf{u}_c)$
      Local pheromone update:
      $\tau_{\mathbf{q}_c\mathbf{u}_c}(k+1) \leftarrow (1-\gamma)\tau_{\mathbf{q}_c\mathbf{u}_c}(k) + \gamma\tau_0$
      $k \leftarrow k+1$
    until $k = K_{\max}$
    $\mathcal{S}_{\text{trial}} \leftarrow \mathcal{S}_{\text{trial}} \bigcup \{s_{\text{p},c}\}$
  end for
  Global pheromone update:
  $\tau_{\mathbf{qu}}(\kappa+1) \leftarrow (1-\rho)\tau_{\mathbf{qu}}(\kappa) + \rho \sum_{s \in \mathcal{S}_{\text{trial}}; (\mathbf{q},\mathbf{u}) \in s} J^{-1}(s), \quad \forall (\mathbf{q},\mathbf{u}); \exists s \in \mathcal{S}_{\text{trial}}; (\mathbf{q},\mathbf{u}) \in s$
  $\kappa \leftarrow \kappa+1$
until $\kappa = \kappa_{\max}$
```

We categorize our algorithm as model-free, as the ants do not have direct access to the model of the system and must learn the control policy just by interacting with it. Note that as the ants interact with the system in parallel, the implementation of this algorithm to a real system requires multiple copies of this system, one for each ant. This hampers the practical applicability of the algorithm to real systems. However, when a model of the real system is available, the parallelism can take place in software. In that case, the learning happens off-line and after convergence, the resulting control policy can be applied to the real system.

3.5. *Selection of the Parameters*

In order to run the algorithm, some parameters need to be selected. The matrices $\mathbf{Q}$ and $\mathbf{R}$ from the cost function (5) need to be selected based on the performance criteria of the optimal control policy with a trade-off between the responsiveness (control speed) and aggressiveness (control power). Based on the system dynamics, the sampling time T_{s} and the quantization space for the state $\mathcal{Q}$ and the action $\mathcal{U}$ must be chosen properly. Faster sampling and finer quantization of the state space make the discretized representation less stochastic, but the number of iterations needed to move from an initial state to a given goal state larger. As such, the task for the ants to find the optimal paths will be more difficult and a longer convergence time can be expected. Furthermore, a larger $\mathcal{Q}$ requires more memory to store the pheromone matrix.

For the algorithm itself, the maximum number of time steps per trial $K_{\max}$ and the maximum number of trials per run of the algorithm $\kappa_{\max}$ need to be chosen. Each of them must be respectively set to a sufficiently large number for the ants to be able to reach the goal state from any given initial state and for the pheromone matrix to converge. The initial pheromone level τ_0 needs to be set to a small positive value. The local and global pheromone levels, γ and ρ respectively, need to be set to a specific value in the range $(0, 1)$. Values of $\gamma = \rho = \{0.01, 0.1\}$ are typically used in literature (see, e.g., [Dorigo *et al.* (1996)] and [Dorigo and Gambardella (1997)]). Because we want the local pheromone update (causing exploration) to have a somewhat lower influence on the pheromone update than the global pheromone update (assigning the credit of the obtained solutions), a good choice is to take $\gamma = 0.01$ and $\rho = 0.1$.

The value of α from the action selection step in (7) trades-off the amount of exploration versus exploitation of the state space by the ants. In our algorithm, in contrast to the Ant System and Ant Colony System algorithms, there is no explicit exploitation rule. A value of $\alpha = 2$ is most common in the literature, but in order to make the action selection step to focus a bit more on the current optimal action, we have observed that a slightly larger value of $\alpha = 3$ works better.

Another important parameter to consider is the number of ants M. Larger values of M are expected to result in a better sampling of the state-action space and as such to improve the convergence to the optimal policy. In the following section,

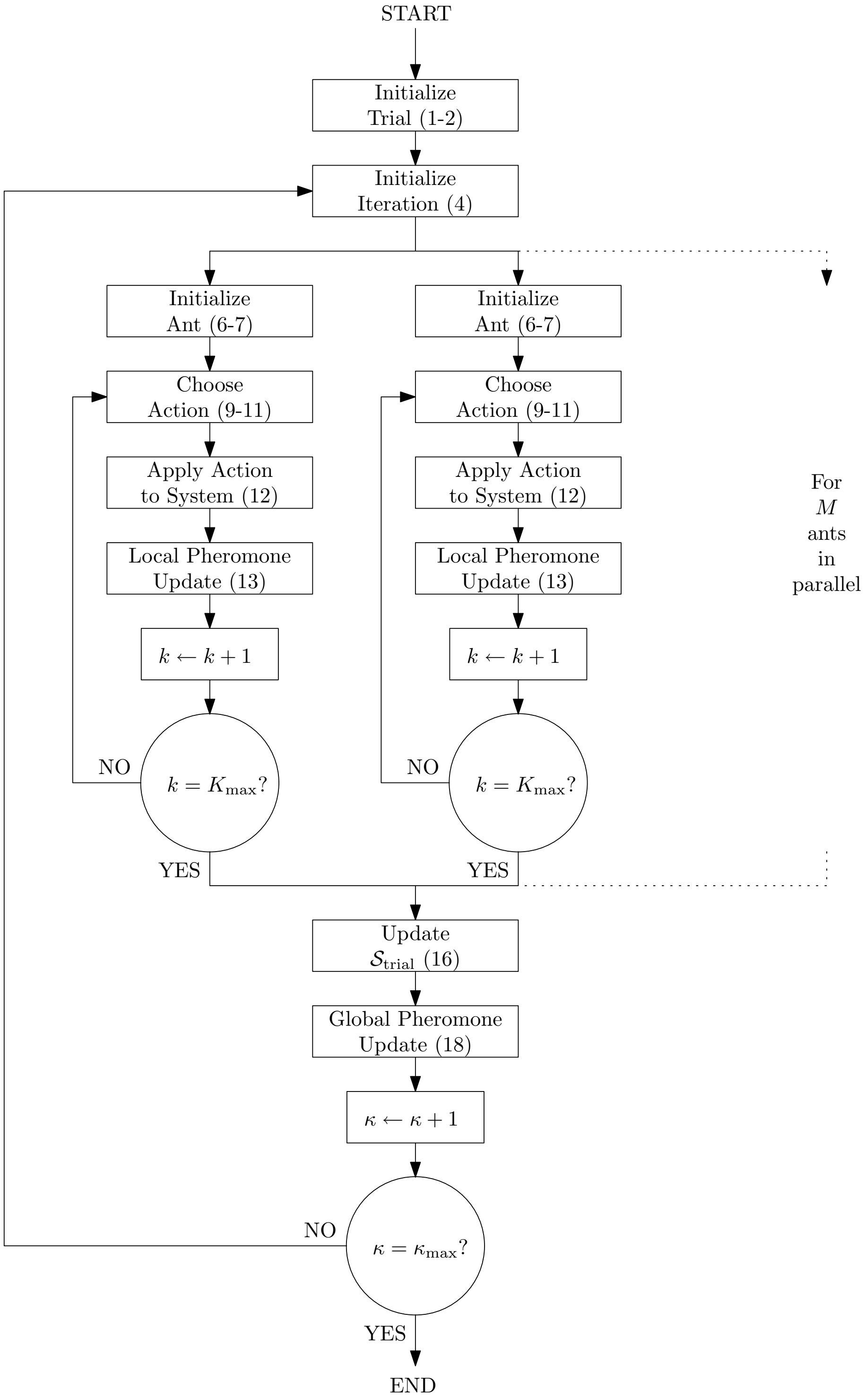


Fig. 3. Flow diagram of the ACO algorithm for optimal control problems. The numbers between parentheses refer to the respective lines of Algorithm 1.

our algorithm is applied to the optimal control of an underactuated pendulum. In particular we study the influence of the number of ants on the performance of the algorithm.

4. ACO Optimal Control of an Underactuated Pendulum

In this section, we will apply our ACO algorithm to the optimal swing-up and stabilization of an underactuated pendulum. We will focus on the performance of the algorithm with respect to the number of ants.

4.1. *Underactuated Pendulum Swing-Up and Stabilization*

The underactuated pendulum swing-up and stabilization problem is a challenging control problem due to the narrow solution space. 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:

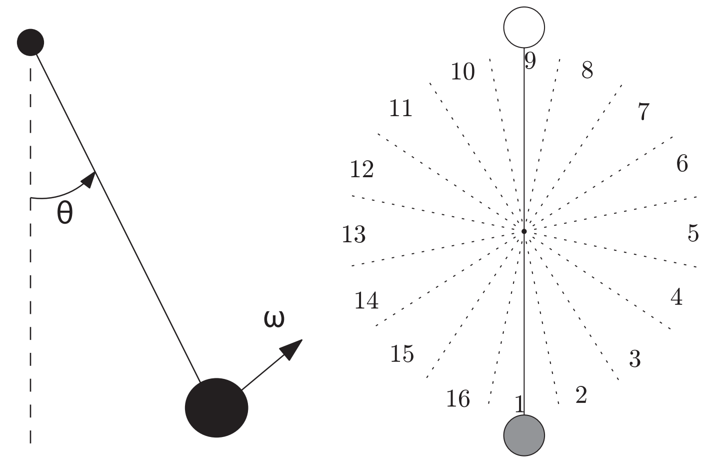
$$J\dot{\omega}(t) = Ku(t) - mgR\sin(\theta(t)) - D\omega(t), \tag{8}$$

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. The signs of the state variables are indicated in Figure 4(a). Furthermore, u is the applied torque and the other parameters with their values as used in the simulations are listed in Table 1.

Table 1. The parameters of the pendulum model and their values used in the experiment.

Parameter	Value	Explanation
J	$0.005\ \mathrm{kg}\cdot\mathrm{m}^2$	pole inertia
K	0.1	motor gain
D	$0.01\ \ \mathrm{kg}\cdot\mathrm{s}^{-1}$	damping
m	$0.1\ \ \mathrm{kg}$	mass
g	$9.81\ \ \mathrm{m}\cdot\mathrm{s}^{-2}$	gravitational acceleration
R	$0.1\ \ \mathrm{m}$	pole half-length

The system is sampled in time and the states θ and ω are quantized according



(a) Convention of θ and ω for the pendulum.

(b) A particular (equidistant) quantization of θ with 16 bins.

Fig. 4. Schematic of the pendulum and quantization of its angle.

to:

$$\begin{aligned} \theta_q &= i, \quad \text{if} \quad \theta \pmod{2\pi} \in (b_{\theta,i}, b_{\theta,i+1}], \\ \omega_q &= i, \quad \text{if} \quad \omega \in (b_{\omega,i}, b_{\omega,i+1}], \end{aligned}$$

with $b_{\theta,i}$ and $b_{\omega,i}$ respectively the ith element of the ordered sets

$$\begin{aligned} \mathcal{B}_\theta &= \left\{ \frac{\pi}{N_\theta}, \frac{3\pi}{N_\theta}, \ldots, \frac{(2N_\theta - 1)\pi}{N_\theta} \right\} \\ \mathcal{B}_\omega &= \left\{ -\infty, -\omega_{\max}, -\omega_{\max} + \frac{2\omega_{\max}}{N_\omega - 2}, \ldots, \omega_{\max}, +\infty \right\}, \end{aligned}$$

where N_θ and $N_\omega \geq 3$ are the number of quantization bins for θ and ω respectively and $\omega_{\max}$ is the maximum (absolute) angular velocity expected to occur. Note that N_θ must be even to make sure that both equilibria fall within a bin and not at the boundary of two neighboring bins, which would result in chattering of the quantized state when the pendulum is near one of the equilibria. For similar reasons, N_ω must be odd.

4.2. *Simulation Parameters*

The ants sample the non-linear state dynamics from (8) with a sample time of $T_\mathrm{s} = 0.1$ s. The number of bins for quantizing the states with a homogeneous distribution are $N_\theta = 40$ and $N_\omega = 41$ for the angle and angular velocity respectively. The maximum angular velocity expected to occur is set to $\omega_{\max} = 9 \text{ rad} \cdot \text{s}^{-1}$. The action set consists of only three actions, namely plus and minus the maximum torque of 0.8 Nm and the zero torque. The total number of quantized states is

$40 \times 41 = 1640$. All the ants are initialized with a random state at the start of each trial. In each trial, the ants get $K_{\max} = 300$ iterations to find the quantized goal state containing the goal state $\mathbf{x}_{\mathrm{g}}$, which is sufficiently long for the ants to swing up and stabilize the pendulum. The simulation runs for a maximum of $\kappa_{\max} = 100$ trials. The global and local evaporation rates of the pheromone levels are fixed to $\rho = 0.1$ and $\gamma = 0.01$ respectively. The value of α from the action-selection step in (7) is chosen to be equal to 3. We repeat the experiment for different number of ants $M \in \{100, 250, 500, 1000, 1250, 1500, 1750, 2000\}$.

4.3. *Simulation Results*

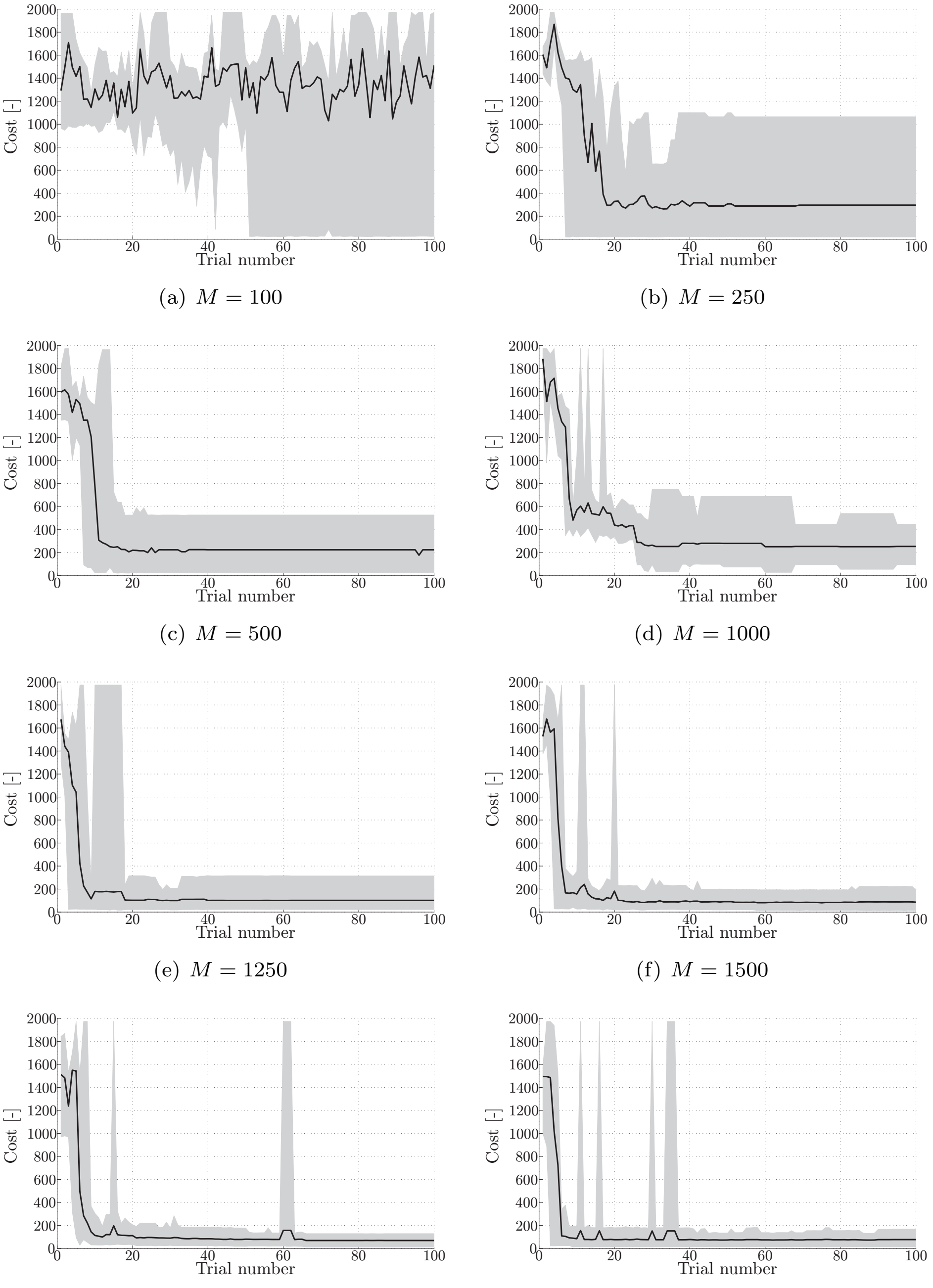
During the experiments, the quality of the control policy derived from the pheromone matrix is evaluated at each trial. For a set of initial states, homogeneously distributed over the state space, the trajectories starting from these states and following the current best policy are evaluated over the cost function (5). For the various experiments, with different numbers of ants, the plot of the progress of this cost throughout the experiments is shown in Figure 5. The solid line represents the mean of this cost over all trajectories and the boundaries of the shaded area correspond to the maximum and minimum cost from this set. The shaded area can be interpreted as being the confidence of the ants in finding the optimal policy.

It can be seen that for $M \geq 250$, the mean of the cost converges. For increasing numbers of ants, the converged mean of the cost slowly decreases. For $M \geq 1250$, it appears to have reached its minimum. The size of the shaded area also decreases with increasing M. The peaks in the shaded area are caused by some ants that could not reach the goal state within the maximum number of iterations in the given trial. These results show that for an increasing number of ants, the performance of the policy and the confidence of the ants that this policy is optimal increases. The convergence speed, however, does not seem to be much influenced by the number of ants. For $M \geq 250$, convergence occurs after about 25 to 50 trials.

During the experiments, we have also recorded the cost of the trajectories that the ants have followed. Figure 6 shows these results, with the solid line representing the mean and the shaded area representing the area bounded by the maximum and the minimum cost over the set of trajectories.

It can be observed that for the cases when the policy converges, the means of these costs converges as well, but the areas bounded by the maximum and minimum costs remain large. This indicates that the ants keep on exploring the state space and as such do not converge themselves to always following the found policy.

The resulting policy of the controller derived from the pheromone trail after 100 trials, when the simulation is finished, is depicted in Figure 7 for the experiments with $M = 250$ and $M = 2000$. In this figure, the optimal action as a function of the quantized state is depicted in a shaded grid, where black represents full negative torque, grey represents zero torque, and white represents full positive torque. In Figure 7(a), the resulting policy for $M = 250$ is much less structured than the



(a) $M = 100$

(b) $M = 250$

(c) $M = 500$

(d) $M = 1000$

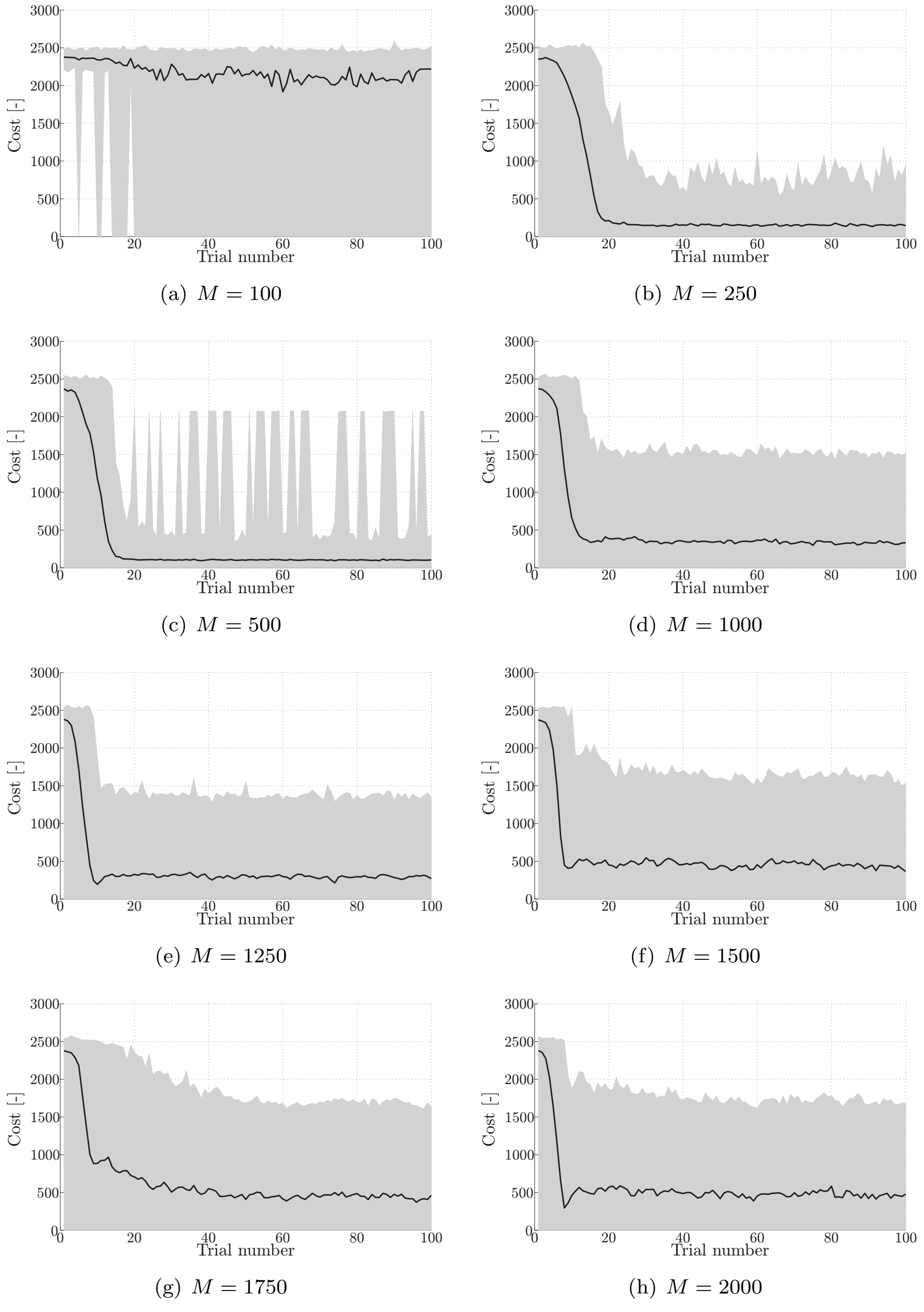
(e) $M = 1250$

(f) $M = 1500$

(g) $M = 1750$

(h) $M = 2000$

Fig. 5. During each experiment, after each trial, the policy is evaluated by computing the cost for a set of trajectories starting from a set of initial states, homogeneously distributed over the state space and following the policy. These figures show the progress of this cost over the trials in the experiments with different numbers of ants. The solid line is the mean of the cost and the boundaries of the shaded area correspond to the maximum and minimum cost over the set of trajectories.



(a) $M = 100$

(b) $M = 250$

(c) $M = 500$

(d) $M = 1000$

(e) $M = 1250$

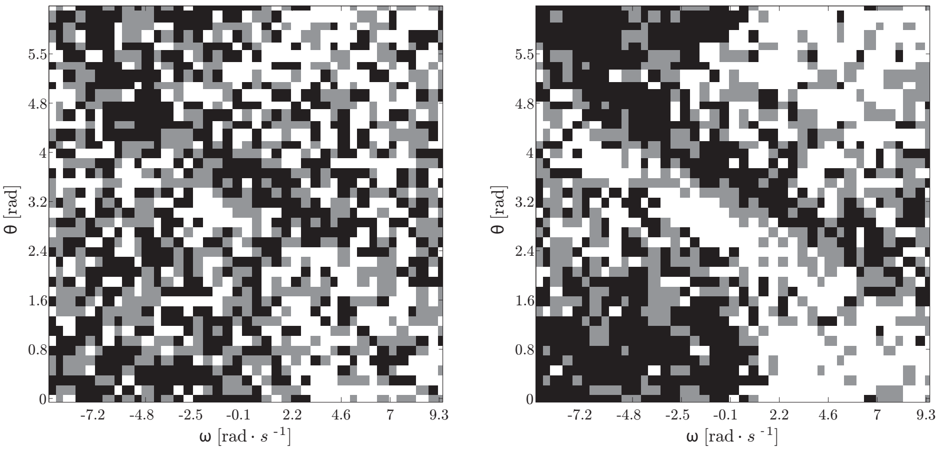
(f) $M = 1500$

(g) $M = 1750$

(h) $M = 2000$

Fig. 6. The cost of the trajectories followed by the ants during each trial. The solid line is the mean of the cost and the boundaries of the shaded area correspond to the maximum and minimum cost over the set of trajectories.

policy depicted in Figure 7(b) for $M = 2000$. It can be observed most clearly from Figure 7(b) that the policy is to apply the torque in the direction of movement for angles near 0, or 2π rad. This corresponds to a destabilizing action with the purpose of swinging the pendulum up higher by accumulating energy. There is a ridge on the diagonal near the goal state in π rad and 0 rad $\cdot\,\mathrm{s}^{-1}$, where the destabilizing action changes to a stabilizing action by applying the torque in the opposite direction of the movement. In the quantized goal state, the policy is to apply the zero torque action. There seems to be a chaotic pattern of states for which the policy is to apply the zero torque action throughout the white and black regions. It is a property of motion systems that the inertia masks the clear effect of a certain action and in particular the zero torque action in some of the states. It turns out that there is no clear optimal action in these states. With 250 ants, the policy contains many more of such states. Applying this policy to control the pendulum shows that the converged policy is too sub-optimal to swing up and stabilize the pendulum. The policy plots for the experiments with numbers of ants between 250 and 2000 are not shown here, but reveal an increasing ordered pattern from the plot in Figure 7(a) to the plot in Figure 7(b).



(a) $M = 250$

(b) $M = 2000$

Fig. 7. Control policy for the simulations with 250 and 2000 ants. The shaded rectangles indicate the optimal action in the respective state. Black represents the full negative torque, grey represents the zero torque, and white represents the full positive torque action.

The behavior of the original continuous model of the pendulum controlled by the control policy as depicted in Figure 7(b) is presented in Figure 8. The figures show the behavior of the pendulum for a set of initial angles and zero velocity. Starting from, e.g., the initial state $\mathbf{x}(0) = (0, 0)^{\mathrm{T}}$, the optimal policy is to swing the pendulum first to about $\frac{1}{2}\pi$ rad and then swing it back and brake in time to stabilize it hanging upside down. The little chattering around the unstable equilibrium is due to the quantization of the state and action space. It is almost impossible to reach an exact angle of π by only applying full positive, full negative, or zero torque in discrete states.

4.4. *Summary*

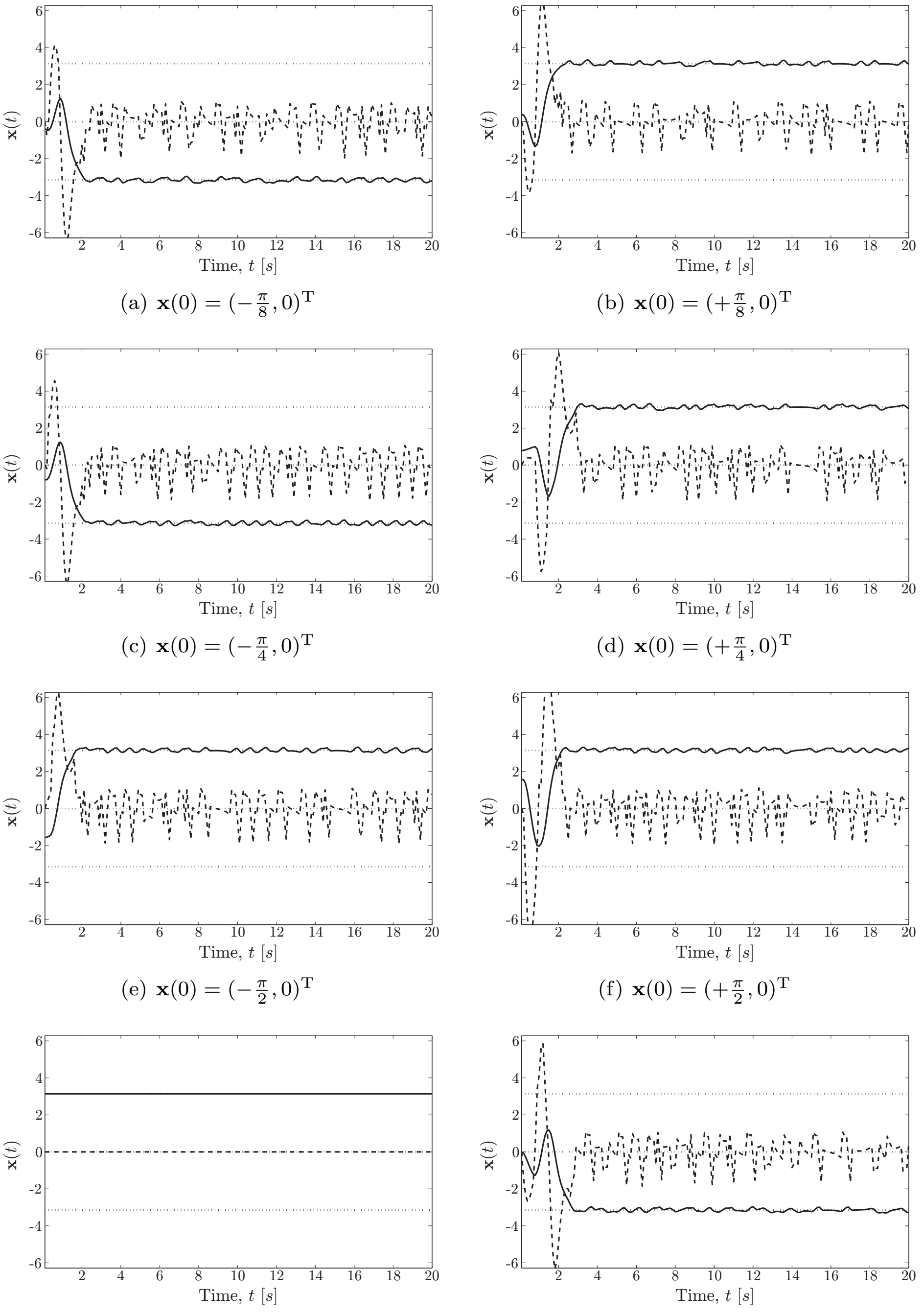
The results from Section 4.3 show that when the number of ants is low, the policy does not converge at all. For larger values of M, the mean of the cost associated with the pheromone matrix does converge, but the confidence remains low, meaning that the pheromone matrix still changes considerably at every trial. For even larger values of M, the mean of the cost associated with the pheromone matrix decreases and the confidence increases. This indicates that the ants have found a more optimal policy that does not vary too much with the progress of the algorithm. The convergence speed does not seem to depend much on the number of ants.

Interestingly, the ants themselves do not converge to a situation in which they all follow the optimal policy, even if the policy itself does converge. This indicates that the ants keep on exploring the state space, thereby potentially capable of tracking changes in the system dynamics.

5. Conclusions and Future Research

This paper has introduced a new method for the design of optimal control policies for continuous-time, continuous-state dynamic systems based on Ant Colony Optimization (ACO). We have discussed how a sampled and quantized version of such a system behaves as a stochastic automaton. The problem of finding an optimal control policy as such can be viewed as a stochastic combinatorial optimization problem. Our novel ACO algorithm is based on the combination of the Ant System and the Ant Colony System, but it has some particular characteristics. The action selection step does not include an explicit probability of choosing the action associated with the highest value of the pheromone matrix in a given state, but only uses a random-proportional rule. In this rule, the heuristic variable, typically present in ACO algorithms, is left out as no action can be marked as being favored over other actions in a particular state a priori. The absence of the heuristic variable makes our algorithm straightforward to apply. Another particular characteristic of our algorithm is that it uses the complete set of solutions of the ants in each trial to update the pheromone matrix, whereas most ACO algorithms typically use some form of elitism, selecting only one of the solutions from the set of ants to perform the global pheromone update. In a control setting, this is not possible, as ants initialized closer to the goal will have experienced a lower cost than ants initialized further away from the goal, but this does not mean that they must be favored in the update of the pheromone matrix.

Using basically standard settings of the parameters, we have demonstrated that the ants can quickly find a solution to the underactuated pendulum swing-up and



(a) $\mathbf{x}(0) = (-\frac{\pi}{8}, 0)^{\mathrm{T}}$

(b) $\mathbf{x}(0) = (+\frac{\pi}{8}, 0)^{\mathrm{T}}$

(c) $\mathbf{x}(0) = (-\frac{\pi}{4}, 0)^{\mathrm{T}}$

(d) $\mathbf{x}(0) = (+\frac{\pi}{4}, 0)^{\mathrm{T}}$

(e) $\mathbf{x}(0) = (-\frac{\pi}{2}, 0)^{\mathrm{T}}$

(f) $\mathbf{x}(0) = (+\frac{\pi}{2}, 0)^{\mathrm{T}}$

(g) $\mathbf{x}(0) = (\pi, 0)^{\mathrm{T}}$

(h) $\mathbf{x}(0) = (0, 0)^{\mathrm{T}}$

Fig. 8. Controller derived by the ACO algorithm applied to the original continuous system for different initial angles and zero initial angular velocity. In each plot, the solid line represents the angle θ (x_1), the dashed line represents the angular velocity ω (x_2), and the horizontal dotted lines indicate the desired goal state (with mod 2π).

stabilization problem. In particular, we have studied the influence of the number of ants on the performance of the algorithm. For the underactuated pendulum problem, we have shown that an increasing number of ants results in improved convergence of the policy. This is because larger numbers of ants are capable of sampling the state-action space much better. We have also shown that the convergence speed does not depend much on the number of ants. Another result shows that although the ants have jointly worked to converge the policy, they will themselves not converge to all following the found policy. This indicates that the ants are capable of constantly monitoring changes in the system dynamics and of adapting a converged policy to new circumstances.

The quantization of the state space introduces stochasticity in the system. Due to this, a particular policy applied to the system may result in different trajectories of the state. For the underactuated pendulum problem, we have demonstrated that regardless of this issue, the ants are capable of finding good policies for solving the control problem. However, with a different, more coarse quantization of the state space, the stochasticity may become too large and the probability of finding a control policy too low. Moreover, depending on the system dynamics, a smaller sampling time and finer quantization may be necessary to capture the system dynamics in the quantized domain properly. Such an increasing size of the quantization space increases the memory requirements of the algorithm and more ants may be needed to sample the quantized state space properly, which leads to an increase in the computational requirements of the algorithm as well.

Our current research focuses on using fuzzy membership functions for the parametric approximation of the state space in order to be able to maintain a continuous representation of the state in the algorithm. With such a continuous representation of the state, the state transitions are no longer stochastic. Furthermore, the number of parameters in the approximation is typically much smaller than the number of quantization levels needed to capture the system dynamics to the same extend. We expect that our ACO approach to optimal control will be much broader applicable with the fuzzy approximation of the state space. Future research must study to what extend this expectation can be met.

The ACO approach to optimal control, as described in this paper, currently lacks the possibility of incorporating prior knowledge of the system, or of the optimal control policy. However, sometimes prior knowledge of such kind is available and using it could be a way to speed up the convergence of the algorithm and to increase the optimality of the derived policy. Future research must focus also on this issue.

Furthermore, this paper does not compare the proposed algorithm to any other method in order to benchmark its performance. Generally, a comparison with an arbitrary other method on a single example is not meaningful, unless it is a well-established benchmark problem with a suite of results already published in the literature. A fair comparison to existing techniques is part of our plans for future research.

Finally, future research will focus on a theoretical analysis of the algorithm.

Acknowledgments

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

Alaykran, K., Engin, O., and Dyen, A. (2007). Using ant colony optimization to solve hybrid flow shop scheduling problems. *International Journal of Advanced Manufacturing Technology*, **35(5-6)**: 541–550.

Åström, K. J. and Wittenmark, B. (1990). *Computer Controlled Systems—Theory and Design*. Prentice-Hall, Englewood Cliffs, New Jersey.

Bianchi, L., Dorigo, M., Gambardella, L. M., and Gutjahr, W. J. (2006). Metaheuristics in stochastic combinatorial optimization: a survey. Technical Report 08, IDSIA, Manno, Switzerland.

Birattari, M., Caro, G. D., and Dorigo, M. (2002). Toward the formal foundation of Ant Programming. In *Proceedings of the International Workshop on Ant Algorithms (ANTS 2002)*, pages 199–201, Brussels, Belgium. Springer-Verlag.

Colorni, A., Dorigo, M., and Maniezzo, V. (1992). Distributed optimization by ant colonies. In Varela, F. J. and Bourgine, P., editors, *Towards a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 134–142, Cambridge, MA. MIT Press.

Dorigo, M. and Blum, C. (2005). Ant colony optimization theory: a survey. *Theoretical Computer Science*, **344(2-3)**: 243–278.

Dorigo, M. and Gambardella, L. (1997). Ant Colony System: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, **1(1)**: 53–66.

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

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. The MIT Press, Cambridge, MA, USA.

Fan, X., Luo, X., Yi, S., Yang, S., and Zhang, H. (2003). Optimal path planning for mobile robots based on intensified ant colony optimization algorithm. In *Proceedings of the IEEE International Conference on Robotics, Intelligent Systems and Signal Processing (RISSP 2003)*, pages 131–136, Changsha, Hunan, China.

Fox, B., Xiang, W., and Lee, H. P. (2007). Industrial applications of the ant colony optimization algorithm. *International Journal of Advanced Manufacturing Technology*, **31(7-8)**: 805–814.

Gambardella, L. M. and Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. In Prieditis, A. and Russell, S., editors, *Machine Learning: Proceedings of the Twelfth International Conference on Machine Learning*, pages 252–260, San Francisco, CA. Morgan Kaufmann Publishers.

Huang, R. H. and Yang, C. L. (2008). Ant colony system for job shop scheduling with time windows. *International Journal of Advanced Manufacturing Technology*, **39(1-2)**: 151–157.

Purnamadjaja, A. H. and Russell, R. A. (2005). Pheromone communication in a robot swarm: necrophoric bee behaviour and its replication. *Robotica*, **23(6)**: 731–742.

Sim, K. M. and Sun, W. H. (2003). Ant colony optimization for routing and load-balancing: survey and new directions. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, **33(5)**: 560–572.

Stützle, T. and Hoos, U. (2000). MAX MIN Ant System. *Journal of Future Generation Computer Systems*, **16**: 889–914.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA.

van Ast, J. M., Babuška, R., and De Schutter, B. (2008). Ant colony optimization for optimal control. In *Proceedings of the 2008 Congress on Evolutionary Computation (CEC 2008)*, pages 2040–2046, Hong Kong, China.

Wang, J., Osagie, E., Thulasiraman, P., and Thulasiram, R. K. (2009). HOPNET: A hybrid ant colony optimization routing algorithm for mobile ad hoc network. *Ad Hoc Networks*, **7(4)**: 690–705.