Delft University of Technology

Delft Center for Systems and Control

Technical report 08-003

Ant colony optimization for optimal control*

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, "Ant colony optimization for optimal control," *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, Hong Kong, pp. 2040–2046, June 2008.

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/08_003.html`

Ant Colony Optimization for Optimal Control

Jelmer van Ast, Robert Babuška, Bart De Schutter
Delft Center for Systems and Control
Delft University of Technology
Mekelweg 2, 2628 CD Delft, the Netherlands

***Abstract*—Ant Colony Optimization (ACO) has proven to be a very powerful optimization heuristic for Combinatorial Optimization Problems (COPs). It has been demonstrated to work well when applied to various NP-complete problems, such as the traveling salesman problem. In this paper, an ACO approach to optimal control is proposed. This approach requires that a continuous-time, continuous-state model of the system, together with a finite action set, is formulated as a discrete, non-deterministic automaton. The control problem is then translated into a stochastic COP. This method is applied to the time-optimal swing-up and stabilization of a pendulum.**

I. Introduction

ANT Colony Optimization (ACO) is inspired by ants and their behavior of finding shortest paths from their nest to sources of food. Without any leader that could guide the ants to optimal trajectories, the ants manage to find these optimal trajectories over time, by interacting with their local environment. The ants initially search for food in a random fashion, but when they have found some, they return home while depositing chemicals, called pheromones. These pheromones attract other ants to follow the same path, and they in turn also deposit pheromones on their way back. Over time, this behavior leads to the emergence of paths, that can be shown to be near-optimal. ACO is used to indicate the class of metaheuristic optimization methods that use these concepts in solving Combinatorial Optimization Problems (COPs) [1].

The basic ACO algorithm and its variants, have successfully been applied to various optimization problems, such as the traveling salesman problem [2], job shop scheduling [3], optimal path planning for mobile robots [4], telecommunication routing [5], and load balancing [6], [7]. An implementation of the ACO concept of pheromone trails for real robotic systems is described in [8]. A survey of ACO and other metaheuristics to stochastic combinatorial optimization problems can be found in [9]. The first application of ACO to optimization problems in a continuous search space is outlined in [10]. More recent work presents and analyzes other extensions of ACO to continuous domains, like the Aggregation Pheromones System [11] and the Differential Ant-Stigmergy Algorithm [12]. An application of ACO to the continuous optimization problem of neural network training is outlined in [13].

This paper introduces a method for applying an ACO algorithm to the design of optimal controllers for continuous-time, continuous-state dynamic systems. In order to make such control problems suitable for ACO, the continuous model of the system is transformed to a non-deterministic discrete automaton. The main element in the automaton is the state-transition function. This function describes the probability distribution over the set of quantized states to which the system makes a transition, after a control action has been applied in the current state. The automaton then represents a Stochastic Combinatorial Optimization Problem (SCOP), suitable for the application of ACO. The model of the original system is reflected in the stochasticity of the problem, which the ants cannot sense directly. A special feature of the ACO algorithm as used in this paper, is that it does not contain any heuristic parameter. It is therefore very straightforward to apply, as there are fewer parameters that need to be set a priori, compared to the usual ACO algorithm. The effectiveness of this method is demonstrated by applying it to the control task of swinging up and stabilizing a pendulum. Noting that the ants in the algorithm do not know the model of the system, finding the optimal control policy is a challenging task. The results show however, that a near optimal controller is found quickly with the proposed ACO method.

The rest of this paper is structured as follows. In Section II, the ACO heuristic is briefly reviewed. Section III presents our main contribution of formalizing a control problem as a stochastic combinatorial optimization problem and describes the ACO algorithm to solve it. Section IV contains the results of the simulations of the pendulum by a controller obtained by ACO and Section V concludes this paper.

II. Ant Colony Optimization

ACO algorithms have been developed to solve hard combinatorial optimization problems [1]. 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 $F : \mathcal{S} \to \mathbb{R}^+$ is a fitness function assigning values to candidate solutions, where higher values correspond to better solutions. The purpose of the algorithm is to find the solution, or set of solutions, $s^* \in \mathcal{S}^* \subseteq \mathcal{S}$ that maximizes 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 as a construction graph consisting of a set of vertices and a set of edges connecting the vertices. A particular solution s consists of solution components, which are denoted by $c_{ij} \in \mathcal{C}$ and are pairs of a vertex and an edge. A particular solution s is thus a concatenation of solution components, and

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).
Bart De Schutter is also with the department of Marine and Transport Technology, Delft University of Technology.

forms a path from the initial vertex to the terminal vertex. How the terminal vertex is defined depends on the problem considered. For instance, in the traveling salesman problem, there are multiple terminal vertices, namely for each ant the terminal vertex is equal to its initial vertex. It will be shown that for the application to control problems, as considered in this paper, the terminal vertex corresponds to the desired steady-state. Associated with the edges are two values: a pheromone trail variable τ_{ij} and a heuristic variable η_{ij}. The pheromone trail represents the acquired knowledge about the optimal solution over time and the heuristic variable provides a priori information about the value of the solution component. Basically, the heuristic variables represent a short-term quality measure of the solution component, while the task is to acquire a concatenation of solution components that overall form the optimal solution. The pheromone variables basically encode the measure of the long-term quality of adding the solution component. The trade-off between these two parameters is important for the performance of the algorithm.

In words, the basic ACO algorithm works as follows. A set of M ants is randomly distributed over the vertices. Initially, the partial solutions s^{p} are empty and the pheromone and heuristic variables are set to some initial value. In each iteration, each ant decides based on some probability distribution, which vertex to add to s^{p} next. This function is typically defined as:

$$p(c_{ij}|s^{\mathrm{p}}) \quad = \quad \frac{\tau_{ij}^{\alpha}\eta_{ij}^{\beta}}{\sum_{c_{il}\in\mathcal{N}(s^{\mathrm{p}})}\tau_{il}^{\alpha}\eta_{il}^{\beta}}, \forall c_{ij}\in\mathcal{N}(s^{\mathrm{p}}), \qquad (1)$$

with $\mathcal{N}(s^{\mathrm{p}})$ the feasible neighborhood given the current partial solution and α and β determining the relative importance of η_{ij} and τ_{ij} respectively.

By moving from vertex i to vertex j, the ants add the associated solution component c_{ij} to their partial solution s^{p} until they reach their terminal vertex and complete their candidate solutions. These candidate solutions are evaluated using the fitness function $F(s)$ and the resulting values are used to update the pheromone values by:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho \sum_{s\in\mathcal{S}_{\mathrm{upd}}|c_{ij}\in s} F(s), \qquad (2)$$

with $\rho \in (0,1]$ the evaporation rate and $\mathcal{S}_{\mathrm{upd}}$ the set of solutions that are eligible to be used for the pheromone update. The pheromone values 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 values decrease by some factor in each iteration. In this way it is avoided that the algorithm prematurely converges to sub-optimal solutions. In the next iteration, each ant repeats the previous steps, but now the pheromone values are updated and can be used to make better decisions about which vertex to move to now. After some stopping criterion has been reached, the values of τ and η on the graph encode the solution, as they determine by (1) the path with the highest probability of being constructed step-wise from any initial vertex to the final vertex.

There exist various rules to construct $\mathcal{S}_{\mathrm{upd}}$, of which the most standard one is to use all the candidate solutions found in the trial. This update rule is called the Ant System (AS) update rule. Although other update rules have found to outperform AS in some applications [1], the method described in this paper will use the AS update rule, because of its easy implementation. Future research will show if other update rules might improve our method.

III. ACO for Optimal Control

A. Discrete Non-Deterministic Automata

Assume a continuous-time, continuous-state system with state vector $\mathbf{x}(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}} \in \mathbb{R}^n$ that must be controlled by a control policy derived from an ACO algorithm. Here, a control policy is a mapping from states to actions and the optimal control policy maps the states to actions such that the resulting sequence of state-action-next state triples yield the optimal solution of the problem. The system of differential equations describing the process to be controlled has to be converted to a discrete-time, quantized-state version before it can be cast as a SCOP. The dynamics are sampled with some fixed and properly chosen sample time h. The continuous state space of the system is quantized with a finite number of bins. Depending on the sizes and the number of these bins, portions of the state space will be represented with one and 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 represent these aspects of the quantization, the continuous model will be cast as a discrete non-deterministic *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
- $\phi : \mathcal{Q}\times\mathcal{U}\times\mathcal{Q} \rightarrow [0,1]$: a state transition function.

Given a discrete state vector $\mathbf{q} \in \mathcal{Q}$, a discrete input symbol $u \in \mathcal{U}$, and a discrete next state vector $\mathbf{q}' \in \mathcal{Q}$, the (Markovian) state transition function ϕ defines the probability of this state transition, $\phi(\mathbf{q}, u, \mathbf{q}')$, making the automaton *non-deterministic*. The probabilities over all states $\mathbf{q}'$ must each sum up to one for each state-action pair $(\mathbf{q}, u)$. An example of a non-deterministic automaton is given in Fig. 1. 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 state after one time step h that is either $q = 1$ or $q = 2$ with some probability.

The probability distribution function determining the transition probabilities reflect the system dynamics. This function is estimated from simulations of the system over a fine grid of combinations of initial states and inputs. This procedure is explained in the next section.

B. Conversion to a SCOP

As mentioned in Section III-A, the first step of formulating the control problem as a stochastic COP is to quantize the state

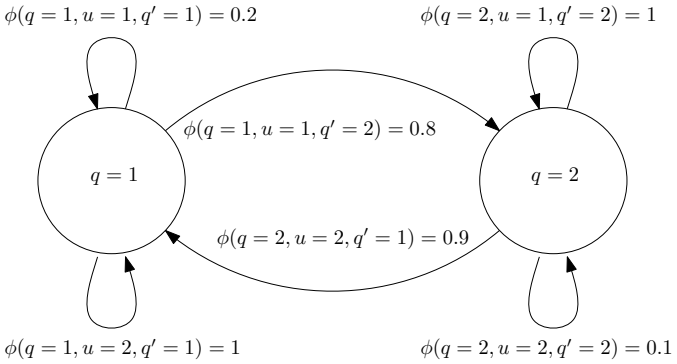


Fig. 1. An example of a non-deterministic automaton.

variables. The model is simulated for initial states on a fine grid, covering the complete domain of the state space, and for each of the inputs in $\mathcal{U}$. The resulting output state is stored for each of these combinations and mapped onto the quantization levels, predefined by $\mathcal{Q}$. These values are normalized, resulting in a probability distribution as described in Section III-A, called the transition function ϕ. This transition function will only be able to properly describe the system dynamics if the state vector describing the system is *Markovian* [14].

The goal of the SCOP is to find the correct sequence of states q and inputs u that leads to the goal state from any initial state in an optimal way, according to the fitness function. As the SCOP corresponds to a non-deterministic automaton, the sequence is actually the sequence that leads to the goal state with the highest probability. This paper will demonstrate in simulations that it is possible to find a solution to the SCOP using ACO for the pendulum swing-up and stabilization problem.

C. ACO Algorithm

The model of the system and the set of possible control actions is reflected in the structure of the automaton. In each state, the ants have to determine which action to choose from. It is not known to the ants to which state this action will take them, as in general there is a set of next states to which the ants can move, according to some probability distribution. At each trial, the ants are initialized randomly over the set of states $\mathcal{Q}$. The ants add the state-action pair to their constructed partial solutions. No heuristic values are associated with the vertices, as there is no a priori information available about the quality of solution components. This is implemented by setting all heuristic values to one. It can be seen that η disappears from (1) in this case. As the values of β and α tune the relative importance of the pheromones τ and the heuristics η, these are also eliminated. The probability of an ant k being in a state $q=i$ taking an action $u=j$ is now:

$$p_{ij}^k = \frac{\tau_{ij}}{\sum_{l\in\mathcal{U}_i}\tau_{il}}, \tag{3}$$

with $\mathcal{U}_i$ the action set the ant has to its disposal in state i. The choice of the next action thus only depends on the value of the pheromone τ_{ij} associated with this vertex. Note that (3) contains no constants that need tuning, making this algorithm much more straightforward to implement than the original ACO algorithm based on (1). The pheromones are initialized equally for all vertices and set to a small value, different from zero. In every trial, all ants construct their solutions until they either have reached the goal state, or the trial exceeds a certain pre-specified limit. The pheromones are then updated according to the following rule:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\sum_{k=1}^{M}\Delta\tau_{ij}^k, \quad \forall c_{ij}\in s\in\mathcal{S}_{\text{iter}},$$

with $\mathcal{S}_{\text{iter}}$ the set of all candidate solutions found in the trial. This type of update rule is comparable to the AS update rule. Future research will investigate what the effect of other types of update rules on the performance of this algorithm is. The value of $\Delta\tau_{ij}^k$ represents the fitness function $F(s)$ and reflects the amount of pheromone an ant k deposits on the vertices it has visited. This value is defined as follows:

$$\Delta\tau_{ij}^k = \frac{1}{\sqrt{(T_k-(1-h))h}} - \frac{1}{\sqrt{(T_{\max}-(1-h))h}}, \tag{4}$$

where T_k is the number of steps the ant k needed to reach the goal state, h is the sample time, used to express the time in seconds, and $T_{\max}$ is the maximum number of steps allowed in a trial. The value of $(1-h)$ is used to make the amount of pheromone deposit approximately equal to $1/h$ when the ant reaches the goal in just one step. The second term in (4) makes sure that the pheromones are not updated when the trial is stopped at the maximum number of time steps and the ant did not yet reach the goal. It is clear that the fitness function – the total amount of pheromones deposited – is maximized if all ants find the shortest path. A plot of the amount of pheromone deposit $\Delta\tau_{ij}^k$ as a function of trial length T_k is shown in Fig. 2.

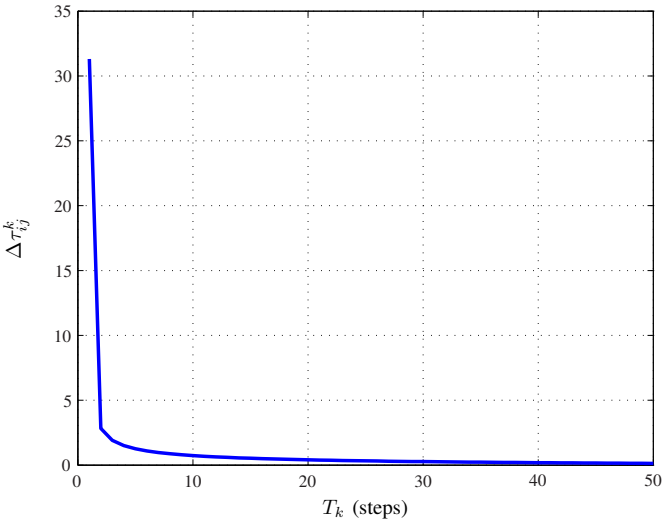


Fig. 2. Pheromone deposit $\Delta\tau_{ij}^k$ as a function of trial length T_k.

The number of ants can be taken equal to the number of states, but a smaller number is also allowed. The number

of ants largely influences the computation time required to determine the controller.

IV. ACO Optimal Control of a Pendulum

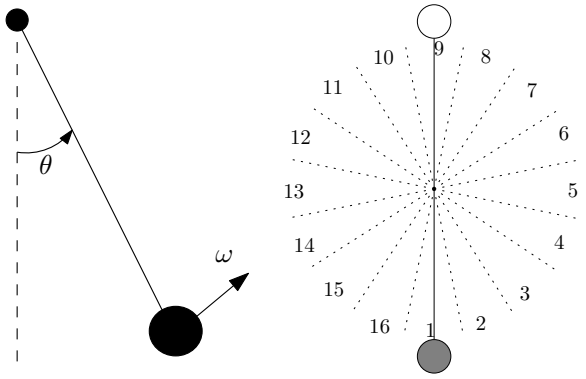
A. Pendulum Swing-Up and Stabilization

The 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. The torque is, however, limited such that it is not possible to move the pendulum to its upright position in one movement. A solution is to first swing the pendulum back and forth in order to accumulate energy to swing it up eventually. The pendulum problem is a nice abstraction of more complex robot control problems. The behavior can be easily analyzed, while the learning problem is challenging.

The non-linear state equations of the pendulum are given by:

$$J\dot{\omega} = Ku - mgR\sin(\theta) - D\omega, \tag{5}$$

with $\theta = x_1$ and $\omega = x_2$ the state variables, representing the angle and angular velocity of the pole respectively. The signs of the state variables are indicated in Fig. 3(a). Furthermore, u is the applied torque and the other parameters with their values as used in the simulations are listed in Table I.



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

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

Fig. 3. Schematic of the pendulum and quantization of its angle.

TABLE I
THE PARAMETERS OF THE PENDULUM MODEL AND THEIR VALUES USED IN THE EXPERIMENT.

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

The states θ and ω are quantized according to:

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

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

$$\mathcal{B}_\theta = \left\{ \frac{(2N_\theta - 1)\pi}{N_\theta}, \frac{\pi}{N_\theta}, \frac{3\pi}{N_\theta}, \ldots, \frac{(2N_\theta - 1)\pi}{N_\theta} \right\}$$
$$\mathcal{B}_\omega = \left\{ -\infty, -\omega_{\max} + \frac{0(2\omega_{\max})}{N_\omega - 2}, -\omega_{\max} + \frac{1(2\omega_{\max})}{N_\omega - 2}, \ldots, -\omega_{\max} + \frac{(N_\omega - 2)(2\omega_{\max})}{N_\omega - 2}, +\infty \right\},$$

in case of equidistant quantization levels. Here 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. 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.

B. Simulation Parameters

First, the non-linear state equation from (5) is transformed to a non-deterministic automaton, with a sample time of $h = 0.1\text{s}$, $N_\theta = 40$, $N_\omega = 41$, and $\omega_{\max} = 9\text{rad} \cdot s^{-1}$. The action set consists of only three actions, namely plus and minus the maximum torque of 0.8Nm and zero torque. The resulting function ϕ is a table with $1640 \times 1640 \times 3$ elements and describes the transition probabilities of the automaton. There are 500 ants initialized with a random state at the start of each trial. In each trial, the ants get 300 time steps to find the goal state, which is sufficiently long for the ants to swing up and stabilize the pendulum. The evaporation rate of the pheromones is fixed to $\rho = 0.1$. In order to measure the convergence, a success rate is defined as the fraction of the ants that reach the goal state within the duration of the trial.

C. Simulation Results

The algorithm converged quickly and monotonically to a success rate of 1 in about 25 trials as can be seen in Fig. 4.

The distribution of the number of steps that the 500 ants needed in order to reach the goal is shown in Fig. 5.

As the ants are randomly initialized over the set of states, some ants will need more steps to reach the goal than other ants. Furthermore, the decisions of the ants at each time step are stochastic, meaning that probably none of the ants chooses an optimal action at every state. However, the power of the ACO approach is that the optimal policy is not encoded by a single ant, but by the colony as a whole. Averaging over the colony reveals that the number of steps needed to reach the optimum from any initial state on the non-deterministic automaton is about 49. With a sampling time of $h = 0.1\text{s}$, this would correspond to 4.9s on the original system when the stochastic policy would be applied. In this research however, it is the purpose to derive a non-stochastic (greedy) policy.

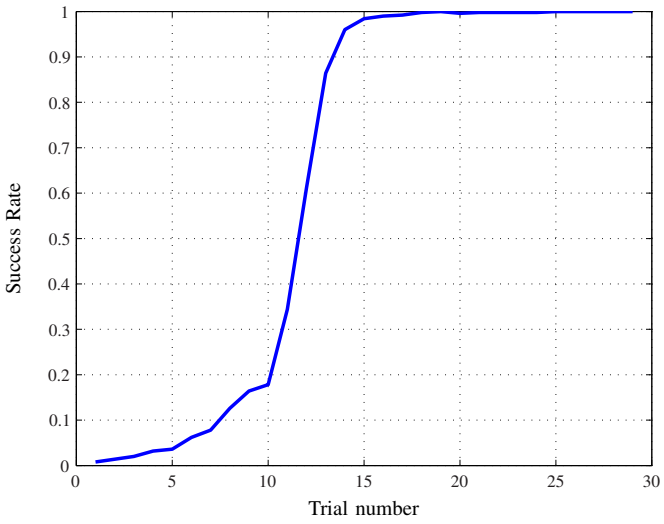


Fig. 4. Success rate: the fraction of the ants that reach the goal state within the time of the trial.

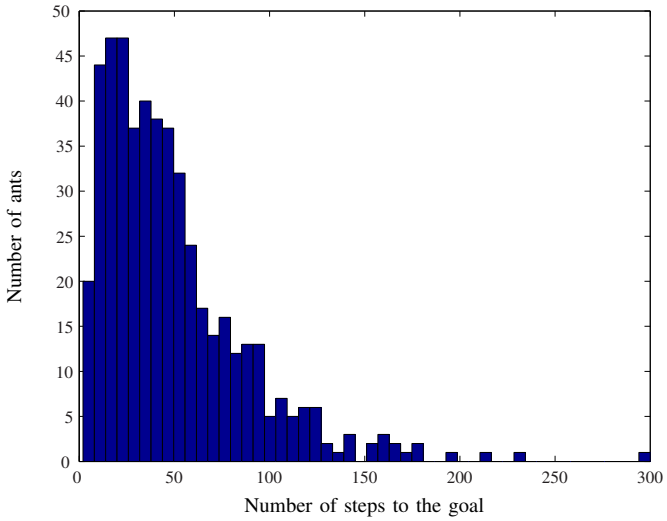


Fig. 5. Histogram showing the distribution of the 500 ants over the number of steps they needed to reach the goal.

The resulting greedy policy of the controller derived from the pheromone trail after convergence is depicted in Fig. 6. 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. There is a clear ridge indicating the region where there is a fine trade-off between the three actions needed to stabilize the pendulum near the goal. The policy far to the right of the ridge is to apply full positive torque. At the far left of the ridge, the optimal action is in general to apply full negative torque. This corresponds to destabilizing the pendulum in the downwards position. Close to the ridge, the optimal policy changes from maximal positive to maximal negative torque. This is to slow the pendulum down in order to balance it. At the ridge itself, the optimal policy is to apply no torque at all, thereby keeping the pendulum still in its goal state. The chaotic nature of the picture indicates that there is not much certainty about the optimal action in many of the states. This is a general phenomenon with motion systems, where in this case, the inertia of the rotating pendulum masks the clear effect of a certain action in some of the states. It turns out that there is no clear optimal action in these states.

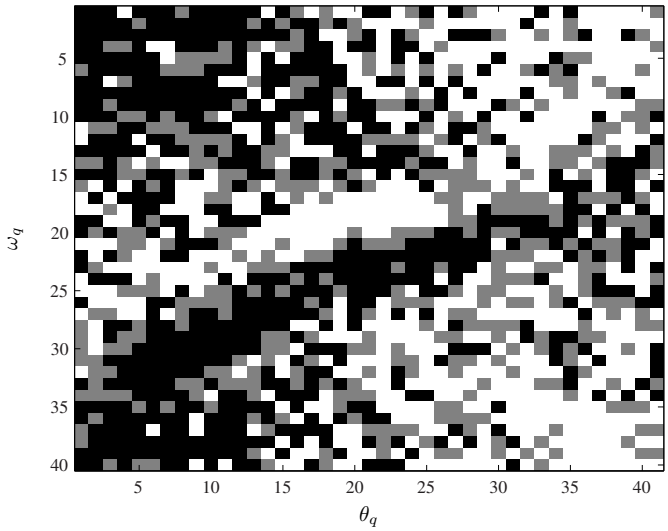
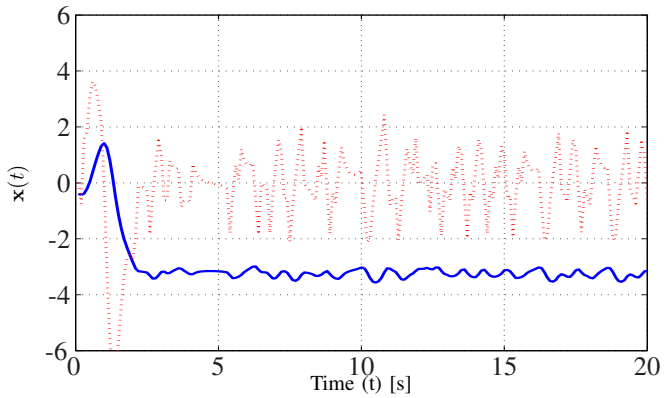


Fig. 6. Control policy. The shaded rectangles indicate the optimal action in the respective state. Black represents full negative torque, grey represents zero torque, and white represents full positive torque.

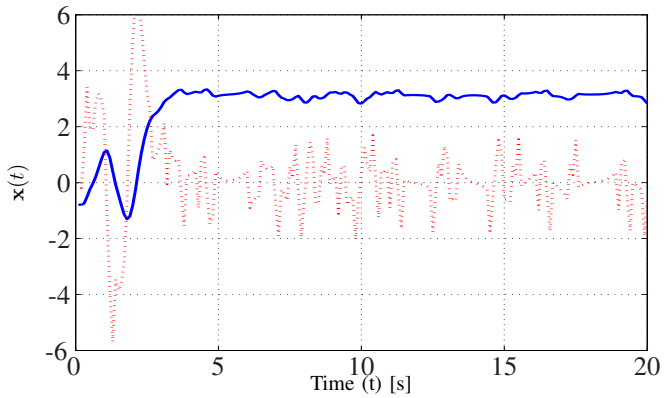
The behavior of the continuous model of the pendulum controlled by the policy from Fig. 6 is presented in Fig. 7 and 8. The figures show the behavior of the pendulum for a set of initial angles and zero velocity. Basically, the optimal policy is to swing the pendulum first to about 90 degrees on the other side and then swing it back and brake in time to stabilize it hanging upsidedown. Exhaustive policy search shows that an optimal policy is able to swing-up and stabilize the pendulum from a downward position with zero angular velocity in about $3.5s$. The figures show that the controller derived by the ACO algorithm controls the pendulum in a near time-optimal way. The little chattering around the instable equilibrium is due to the quantization of the state and action space, because it is almost impossible to reach an exact angle of π by only applying full positive, full negative, or zero torque in discrete states. It would of course be possible to allow for a larger action set. This would probably increase the 'smoothness' of the policy, but also rapidly increase the convergence time. Based on Fig. 7 and 8, the resulting behavior is thus considered to be very close to optimal.

V. Conclusions

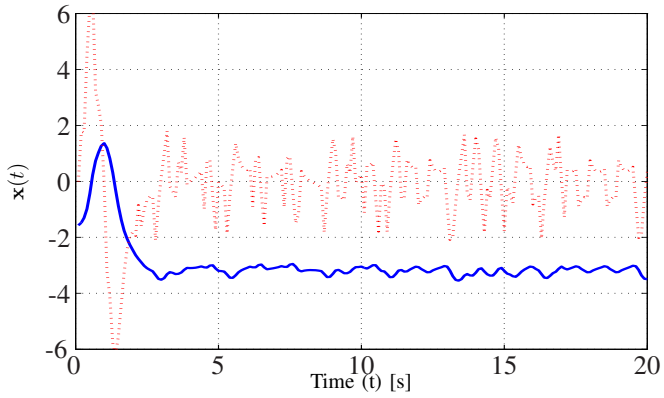
This paper has proposed a new method to the design of an optimal controller for continuous-time, continuous-state dynamic systems based on ACO. It has been described how the dynamics of the system can be transformed into a non-deterministic discrete automaton, where the original state



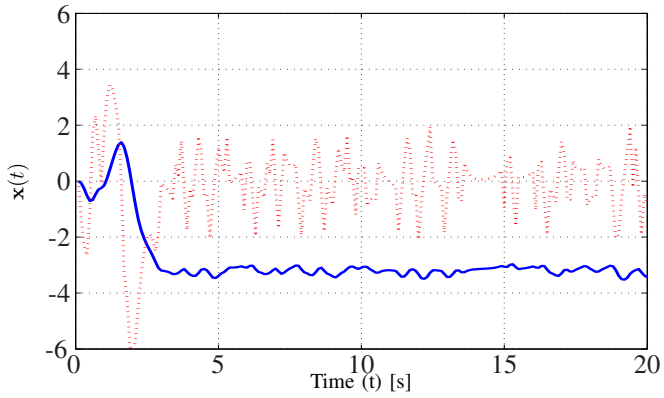
(a) $\mathbf{x}(0) = (-\frac{\pi}{8}, 0)^{\mathrm{T}}$



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

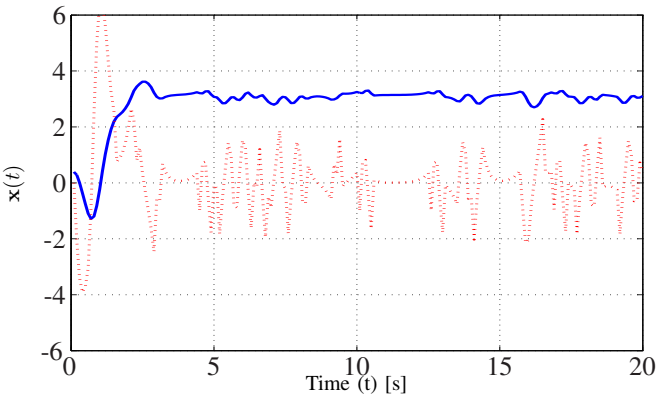


(c) $\mathbf{x}(0) = (-\frac{\pi}{2}, 0)^{\mathrm{T}}$

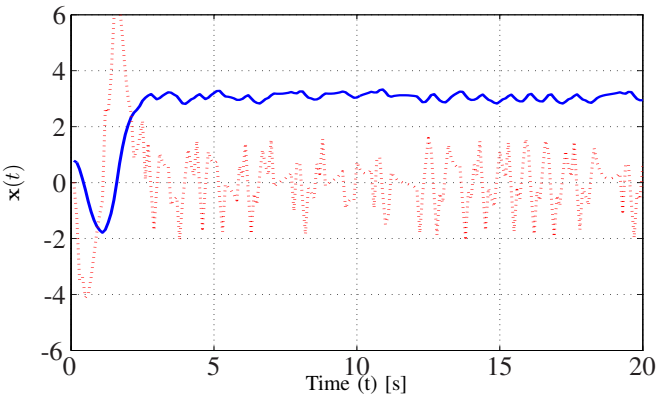


(d) $\mathbf{x}(0) = (0, 0)^{\mathrm{T}}$

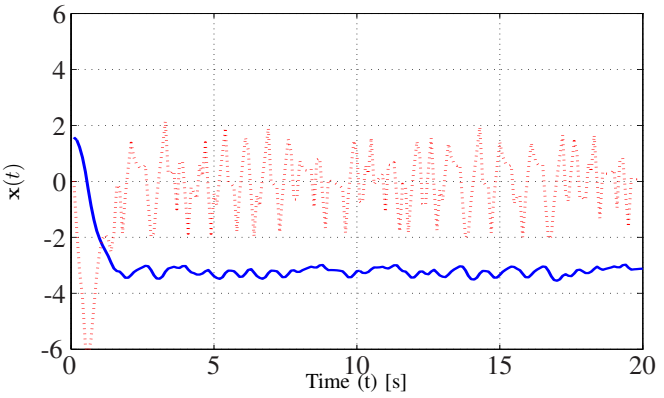
Fig. 7. Simulation of the original continuous system, controlled by the controller derived by the ACO algorithm after 25 trials. The figures show the behavior of the pendulum for the initial angles $\{-\frac{\pi}{8}, -\frac{\pi}{4}, -\frac{\pi}{2}, 0\}$[rad] and zero initial angular velocity. The solid lines represent the angle $\theta(x_1)$[rad] and the dashed lines represent the angular velocity $\omega(x_2)[\mathrm{rad} \cdot s^{-1}]$.



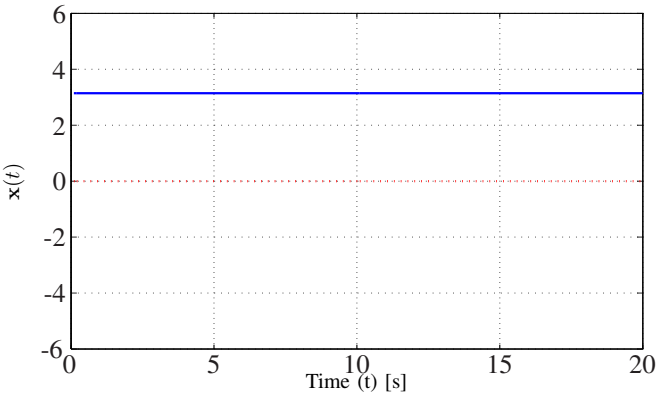
(a) $\mathbf{x}(0) = (+\frac{\pi}{8}, 0)^{\mathrm{T}}$



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



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



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

Fig. 8. Simulation of the original continuous system, controlled by the controller derived by the ACO algorithm after 25 trials. The figures show the behavior of the pendulum for the initial angles $\{+\frac{\pi}{8}, +\frac{\pi}{4}, +\frac{\pi}{2}, \pi\}$[rad] and zero initial angular velocity. The solid lines represent the angle $\theta(x_1)$[rad] and the dashed lines represent the angular velocity $\omega(x_2)[\mathrm{rad} \cdot s^{-1}]$.

space is quantized into a finite set of states and the transition function is a probability distribution over these states under an action from a discrete action set. The ACO algorithm does not require the assignment of heuristic values to the state-action pairs and is therefore straightforward to apply.

The proposed ACO algorithm has been applied to the control of the swing-up and stabilization of a pendulum. An optimal controller was found after only a few trials.

Future research will consist in comparing the ACO optimal control method to other learning-based methods, such as reinforcement learning, that are also able to find optimal controllers by interacting with the system, as well as to other ACO algorithms for continuous domains. These methods will be applied to a real setup investigating the practical applicability of the newly proposed method. Furthermore, the question of how to validate the optimality of the controller will be addressed. Other variations of the pheromone update rule in the ACO algorithm will also be investigated.

REFERENCES

[1] M. Dorigo and C. Blum, "Ant colony optimization theory: a survey," *Theoretical Computer Science*, vol. 344, no. 2-3, pp. 243–278, November 2005.

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

[3] P. K. Jain and P. K. Sharma, "Solving job shop layout problem using ant colony optimization technique," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, vol. 1, Big Island, HI, USA, October 2005, pp. 288–292.

[4] X. Fan, X. Luo, S. Yi, S. Yang, and H. Zhang, "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)*, vol. 1, Changsha, Hunan, China, October 2003, pp. 131–136.

[5] M. T. Islam, P. Thulasiraman, and R. K. Thulasiram, "A parallel ant colony optimization algorithm for all-pair routing in MANETs," in *Proceedings of the International Symposium on Parallel and Distributed Processing (IPDPS 2003)*, Nice, France, April 2003.

[6] Y. Hsiao, C. Chuang, and C. Chien, "Computer network load-balancing and routing by ant colony optimization," in *Proceedings of the IEEE International Conference on Networks (ICON 2004)*, vol. 1, Singapore, November 2004, pp. 313–318.

[7] K. M. Sim and W. H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 33, no. 5, pp. 560–572, September 2003.

[8] A. H. Purnamadjaja and R. A. Russell, "Pheromone communication in a robot swarm: necrophoric bee behaviour and its replication," *Robotica*, vol. 23, no. 6, pp. 731–742, 2005.

[9] L. Bianchi, M. Dorigo, L. M. Gambardella, and W. J. Gutjahr, "Metaheuristics in stochastic combinatorial optimization: a survey." IDSIA, Tech. Rep. 08, March 2006.

[10] G. Bilchev and I. C. Parmee, "The ant colony metaphor for searching continuous design spaces," in *Selected Papers from AISB Workshop on Evolutionary Computing*, ser. Lecture Notes in Computer Science, T. Fogarty, Ed., vol. 993. London, UK: Springer-Verlag, April 1995, pp. 25–39.

[11] 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, September 2005, pp. 880–887.

[12] 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)*, September 2007, pp. 157–164.

[13] 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. [Online]. Available: http://dx.doi.org/10.1007/s00521-007-0084-z

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