2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Fuzzy Ant Colony Optimization for Optimal Control

Jelmer van Ast, Robert Babuška, and Bart De Schutter

Abstract— **Ant Colony Optimization (ACO) has proven to be a very powerful optimization heuristic for Combinatorial Optimization Problems. While being very successful for various NP-complete optimization problems, ACO is not trivially applicable to control problems. In this paper a novel ACO algorithm is introduced for the automated design of optimal control policies for continuous-state dynamic systems. The so called Fuzzy ACO algorithm integrates the multi-agent optimization heuristic of ACO with a fuzzy partitioning of the state space of the system. A simulated control problem is presented to demonstrate the functioning of the proposed algorithm.**

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 in a distributed fashion. In an ACO algorithm, the metaphorical ants are agents programmed to find an optimal combination of elements of a given set that maximizes some utility function. The key ingredient in ACO and its biological counterpart 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 better trajectories. ACO represents the class of metaheuristic optimization methods that use the concepts of distributed optimization and pheromone maps in solving Combinatorial Optimization Problems [1].

This paper introduces an ACO-based algorithm for the automated design of optimal control policies for continuous-state dynamic systems. The algorithm combines the concepts of multi-agent optimization and fuzzy approximation of the state space in a novel approach.

This paper is structured as follows. Section II describes the relation of the subjects covered in this paper to the state of the art. In Section III, the ACO heuristic is briefly reviewed. Section IV presents some preliminaries on the control problem and the fuzzy partitioning of the state space. In Section V the Fuzzy ACO algorithm is introduced and described in detail. Section VI demonstrates the functioning of the Fuzzy ACO algorithm on a simple control problem and Section VII concludes this paper.

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)

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

II. Relation to the State of the Art

The state of the art related to the subjects covered in this paper can be summarized as follows.

1) The original ACO algorithm in the form of the Ant System has been introduced in [2] with the Ant Colony System in [3] and the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System in [4]. The basic ACO algorithm and its variants have successfully been applied to various optimization problems [5], [6], [7], [8]. A detailed description of ACO algorithms and its applications can be found in the survey [1] and the book [5].

2) One of the first real application of the ACO framework to optimization problems in continuous search spaces is described in [9] and [10]. Earlier application of the ant metaphor to continuous optimization appears in [11], with more recent work like the Aggregation Pheromones System in [12] and the Differential Ant-Stigmergy Algorithm in [13].

3) The first application of ACO to the automated design of optimal control policies for continuous-state dynamic systems has been developed by the authors of this paper in [14]. The method however is hampered by the large number of bins needed to quantize the state space in order to capture the dynamics of the original system. This *curse of dimensionality* is a phenomenon widely encountered when an originally continuous-state system needs to be represented using a finite number of quantized states.

4) There are only a few publications that combine ACO with the concept of fuzzy control [15], [16], [17]. In all three publications fuzzy controllers are obtained using ACO, rather than presenting an actual fuzzy ACO algorithm, as introduced in this paper.

This paper contributes to the above four categories of the state of the art. It contributes to 1) by further exploring the applicability of ACO and to 2) by presenting a novel approach to the optimization in continuous spaces using ACO. It presents an extension to the work mentioned in 3) by curbing the curse of dimensionality through the use of a parametric interpolation to retain a continuous state space with a finite number of parameters, typically much smaller than the number of quantized states that would be necessary to achieve similar accuracy. The interpolation method of choice is *fuzzy approximation*. Finally, the work in this paper is different from that described in 4) as it presents an ACO algorithm that operates on the membership *degrees* of the ants to find the optimal pheromone map corresponding to the fuzzy partitioning of the state space, rather than optimizing the membership *functions* themselves, or the linguistic rules.

978-1-4244-4524-0/09/$25.00 ©2009 AACC

III. Ant Colony Optimization

A. Framework for ACO Algorithms

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 $F : \mathcal{S} \rightarrow \mathbb{R}_+$ 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^*, or set of solutions $\mathcal{S}^*$, $s^* \in \mathcal{S}^* \subseteq \mathcal{S}$ 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 graph consisting of a set of vertices and a set of arcs connecting the vertices. A particular solution s consists of solution components, which are denoted by c_{ij} and which are pairs of a vertex and an arc. Here the subscript ij denotes that this solution component consists of a vertex i and an arc that connects this vertex to another vertex j. A particular solution s is thus a concatenation of solution components, and forms a tour 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[1], there are multiple terminal vertices, namely for each ant the terminal vertex is equal to its initial vertex, after visiting all other cities (i.e. vertices) exactly once. For the application to control problems, as considered in this paper, the terminal vertex corresponds to the desired steady-state. 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 solution over time and the heuristic variable provides a priori information about the quality of the solution component, i.e., the quality of moving to a node j from a node i. In the case of the traveling salesman problem, the heuristic variables typically represent the distance between the respective pair of cities. In general, 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 ACO.

B. The Ant System and Ant Colony System

The basic ACO algorithm works as follows. A set of M ants is randomly distributed over the vertices. The heuristic variables 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 c_{ij} 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:

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

with α and β determining the relative importance of η_{ij} and τ_{ij} respectively. The 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, each ant adds the associated solution component c_{ij} to its partial solution s_{p} until it reaches its terminal vertex and completes its candidate solution. This candidate solution is evaluated using the fitness function $F(s)$ and the resulting value is used to update the pheromone levels by:

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

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, which will be explained further on in this section. The pheromone deposit $\Delta\tau_{ij}(s)$ is computed as:

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

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. In this way it can be avoided that the algorithm prematurely converges to suboptimal solutions. In the next 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 pheromone levels encode the solution of the optimization problem.

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 $\mathcal{S}_{\mathrm{trial}}$[2]. This update rule is typical for the first ACO algorithm, called the Ant System (AS) [2]. Other update rules have shown to outperform the AS update rule. Most notably, the two most successful ACO variants in practice, the Ant Colony System (ACS) [3] and the $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System [4] respectively use the *Global Best* and the *Iteration Best* update rule. 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 computational complexity of the algorithm. An important

[1] 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.

[2] In ACO literature, the term trial is seldom used. It is rather a term from the reinforcement learning (RL) community [18]. In our opinion it is also a more appropriate term for ACO 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 ACO is iteration and the set of all candidate solutions found in each iteration is denoted as $\mathcal{S}_{\mathrm{iter}}$. In this paper, equivalently to RL, we prefer to use the word iteration to indicate one interaction step with the system.

element from the ACS algorithm is the local pheromone update rule, which occurs while iterating through the trial 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 index of the solution component 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 introduced Fuzzy ACO algorithm is based on the AS combined with the local pheromone update rule of ACS.

IV. Control Problem and Fuzzy Approximation

A. The Optimal Control Problem

Assume we have a nonlinear system, characterized by a state vector $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^{\mathrm{T}}$. Also assume that the state can be controlled by an input $\mathbf{u}$ and can be measured at discrete time steps, with a sample time T_{s}, and that the system dynamics in discrete-time can be denoted as:

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

with k the discrete time index. 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:

$$J = \sum_{k=0}^{K-1} \mathbf{e}^{\mathrm{T}}(k+1) Q \mathbf{e}(k+1) + \mathbf{u}^{\mathrm{T}}(k) R \mathbf{u}(k), \tag{4}$$

with $\mathbf{e}(k+1) = \mathbf{x}(k+1) - \mathbf{x}_{\mathrm{g}}$ the error at time $k+1$ and Q and R positive definite matrices of appropriate dimensions. The problem is to find a nonlinear mapping from states to input $\mathbf{u}(k) = g(\mathbf{x}(k))$ 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 quadratic cost function in (4) is minimized if the goal is reached in minimum time given the dynamics of the system and restrictions on the size of the input. The matrices Q and R balance the importance of speed versus the aggressiveness of the controller.

In the case of an ACO implementation, only a finite set of input values, called the action set $\mathcal{U}$, can be used. The goal is then actually to find the nonlinear mapping of states to the action set that minimizes (4).

B. Quantization Issues

For a system with a continuous-valued state space, optimization algorithms like ACO can only be applied if the state space is quantized. The most straightforward way to do this is to divide the state space into a finite number of bins, such that each state value is assigned to exactly one bin. These bins can be enumerated and used as the vertices in the ACO graph. It is easy to see that the state transitions are stochastic. Namely, each bin corresponds to a range of states and an input to the system may drive its state to one bin, or another. In [14], a variation to the AS algorithm is introduced that is capable of solving an optimal control problem in this manner. However, the number of bins needed to accurately capture the dynamics of the original system may become very large even for simple systems with only two state variables. Moreover, the time complexity of the ACO algorithm grows exponentially with the number of bins, making the algorithm infeasible for realistic systems.

A much better alternative is not to quantize the state space at all, but to approximate it by a smooth parameterized function approximator. In that case, there is still a finite number of parameters, but this number is typically much smaller compared to using crisp quantization. The universal function approximator that is used in this paper is the fuzzy approximator.

C. Fuzzy Approximation

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. Let A_i denote the membership functions for x_1, with a_i their centers for $i = 1, \dots, N_A$, 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 $i = 1, \dots, N_B$, with N_B the number of membership functions for x_2. Similarly, the membership functions can be defined for the other state variables in $\mathbf{x}$, but for the sake of notation, the discussion in this paper limits the number to two, without loss of generality. Note that in the example in Section VI, the order of the system is four.

The membership degree of A_i and B_i are respectively denoted by $\mu_{A_i}(x_1(k))$ and $\mu_{B_i}(x_2(k))$ for a specific value of the state at time k. The degree of fulfillment is computed by multiplying the two membership degrees:

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

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

$$\begin{aligned} \boldsymbol{\beta}(\mathbf{x}(k)) = [&\beta_{11}(\mathbf{x}(k)) \quad \beta_{12}(\mathbf{x}(k)) \quad \dots \quad \beta_{1N_B}(\mathbf{x}(k)) \\ &\beta_{21}(\mathbf{x}(k)) \quad \beta_{22}(\mathbf{x}(k)) \quad \dots \quad \beta_{2N_B}(\mathbf{x}(k)) \\ &\dots \quad \beta_{N_A N_B}(\mathbf{x}(k))]^{\mathrm{T}}, \end{aligned} \tag{5}$$

which is a vector containing β_{ij} for all combinations of ij. Each element will be associated to a vertex in the graph used by the Fuzzy ACO algorithm introduced in this paper. Most of the steps taken from the AS and ACS algorithms for dealing with the $\boldsymbol{\beta}(\mathbf{x}(k))$ vectors from (5) need to be reconsidered. This is the subject of the following section.

V. Fuzzy ACO for Optimal Control

A. Outline of the Algorithm

In the original paper on ACO for optimal control [14], the continuous state variables were quantized into a finite

number of bins, called the quantized states. All combinations of these quantized states for the different state variables corresponded to the nodes in the graph and the arcs corresponded to transitions from one quantized state to another. Because of the quantization, the resulted system was transformed into a stochastic decision problem. However, the pheromones were associated to these arcs as usual. In the fuzzy case, the state space is partitioned by membership functions, as described in Section IV-C and the combination of the indices to these membership functions for the different state variables correspond to the nodes in the construction graph. With the fuzzy interpolation, the system remains a deterministic decision problem, but the transition from node to node 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. This updating will be described in Sections V-E and V-F.

In [14], the vertex to vertex transitions of an ant are not deterministic. In Fuzzy ACO, an ant is not assigned to a certain vertex at a certain time, but to all vertices according to some degree of fulfillment at the same time and a transition from vertex to vertex is not trivial either. Because of this, a solution component c_{ij} does not contain pairs of vertex-next vertex, but of state-action. For this reason, a pheromone τ_{ij} is now denoted as τ_{iu} with i the index of the vertex (i.e. the corresponding element of $\boldsymbol{\beta}$) and u the action. For the sake of notation, no distinction will be made between the actual input $\mathbf{u}(k)$ and the index of the input (the action) u.

Similar to the definition of the vector of all degrees of fulfillment in (5), the vector of all pheromones for a certain action u at time k is denoted as:

$$\boldsymbol{\tau}_u(k) = \begin{bmatrix} \tau_{1u}(k) & \tau_{2u}(k) & \dots & \tau_{N_A N_B u}(k) \end{bmatrix}^{\mathrm{T}}.$$

B. Outline of a Trial

In the following sections, the fuzzy action selection and the local and global pheromone update are explained in more detail. Two more elements in this algorithm need special attention, namely the initialization of the ants and the determination whether an ant has reached the goal.

When using a Global Best pheromone update rule in an optimal control problem, all ants have to be initialized to the same state, as starting from states that require less time and less effort to reach the goal would always result in a better Global Best solution. Ultimately, initializing an ant exactly in the goal state would be the best possible solution and no other solution, starting from more interesting states would get the opportunity to update the pheromones in the global pheromone update phase. In order to find a control policy from *any* initial state to the goal state, the Global Best update rule cannot be used. Simply using all solutions of all ants in the updating, like in the original AS algorithm, the resulting algorithm does allow for random initialization of the ants over the state space and is therefore used in the Fuzzy ACO algorithm.

Regarding the terminal condition for the ants, with the fuzzy implementation, none of the vertices can be pointed out as being the terminal vertex. Rather there has to be defined a set of membership functions that can be used to determine to what degree the goal state has been reached. These membership functions can be used to express the linguistic fuzzy term of the state being *close to the goal*. If this has been satisfied, the ant has terminated its trial.

C. Parameter Setting

Some of the parameters are similarly initialized as with the ACS. The global and local pheromone trail decay factors are set to a preferably small value, respectively $\rho \in (0, 1)$ and $\gamma \in (0, 1)$. There will be no heuristic parameter associated to the arcs in the construction graph, so only an exponential weighting factor for the pheromone trail $\alpha > 0$ needs to be chosen. Increasing α leads to more biased decisions towards the one corresponding to the highest pheromone level. Choosing $\alpha = 2$ or 3 appears to be an appropriate choice. Furthermore, some control parameters for the algorithm need to be chosen, such as the maximum number of iterations per trial, $K_{\max}$, and the maximum number of trials, $T_{\max}$. The latter one can be set to, e.g., 100 trials, where the former one depends on the sample time T_{s} and a guess of the time needed to get from the initial state to the goal optimally, T_{guess}. A good choice for $K_{\max}$ would be to take about 10 times the expected number of iterations $T_{\mathrm{guess}} \cdot T_{\mathrm{s}}^{-1}$. Specific to the fuzzy implementation, the number of membership functions and the spacing over the state domain need to be determined. Furthermore, the pheromones are initialized as $\tau_{iu} = \tau_0$ for all i, u, where τ_0 is a small, positive value, which, according to [3] can be chosen as $\tau_0 = (n \cdot L_{\mathrm{guess}})^{-1}$, with n the number of nodes and L_{guess} a guess of the optimal tour length. Finally the number of ants M must be chosen large enough such that the complete state space can be visited frequently enough.

D. Fuzzy Action Selection

The action is chosen randomly according to the probability distribution:

$$p_c\{u|\boldsymbol{\beta}_c(k)\}(k) = \boldsymbol{\beta}_c^{\mathrm{T}}(k) \cdot \frac{\boldsymbol{\tau}_u^{\alpha}(k)}{\sum_{l \in \mathcal{U}} \boldsymbol{\tau}_l^{\alpha}(k)}, \tag{6}$$

where all operations are performed element-wise, except for the inner product $(\cdot)$. Note that when $\boldsymbol{\beta}_c$ contains exactly one 1 and for the rest only zeros, this would correspond to the crisp case, where the state is quantized to a set of bins and (6) then reduces to the original case in (1).

E. Local Pheromone Update

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

$$\begin{aligned} \boldsymbol{\tau}_u \leftarrow & \boldsymbol{\tau}_u(1-\beta) + ((1-\gamma)\boldsymbol{\tau}_u + \gamma\tau_0)\boldsymbol{\beta} \\ & = \boldsymbol{\tau}_u(1-\gamma\boldsymbol{\beta}) + \tau_0(\gamma\boldsymbol{\beta}), \end{aligned} \tag{7}$$

where all operations are performed element-wise.

As all ants update the pheromone levels associated with the state just visited and the action just taken in parallel,

one may wonder whether or not the order in which the updates are done matters, when the algorithm is executed on a standard CPU, where all operations are done in series. If it would matter, there would be a serious flaw in the algorithm. With crisp quantization, the ants may indeed sometimes visit the same state and with fuzzy quantization, the ants may very well share some of the membership functions with a membership degree larger than zero. We will show that in both cases, the order of updates in series does not influence the final value of the pheromones after the joint update. In the original, crisp case, the local pheromone update from (3) may be rewritten as follows:

$$\begin{aligned}\tau_{ij}^{(1)} \leftarrow & (1-\gamma)\tau_{ij} + \gamma\tau_0 \\ & = (1-\gamma)(\tau_{ij} - \tau_0) + \tau_0.\end{aligned}$$

After n updates, the pheromone level is reduced to:

$$\tau_{ij}^{(n)} \leftarrow (1-\gamma)^n(\tau_{ij} - \tau_0) + \tau_0, \tag{8}$$

which shows that the order of the update is of no influence to the final value of the pheromone level.

For the fuzzy case a similar derivation can be made. In general, after all the ants have performed the update, the pheromone vector is reduced to:

$$\boldsymbol{\tau}_u \leftarrow \prod_{c\in\{1,2,\ldots,M\}} \{1-\gamma\boldsymbol{\beta}_c\}(\boldsymbol{\tau}_u - \tau_0) + \tau_0, \tag{9}$$

where again all operations are performed element-wise. This result also reveals that the final values of the pheromones are invariant with respect to the order of updates. Furthermore, also note that when $\boldsymbol{\beta}_c$ contains exactly one 1 and for the rest only zeros, corresponding to the crisp case, the fuzzy local pheromone update from either (7) or (9) reduces to the original case in respectively (3) or (8).

F. Global Pheromone Update

The global pheromone update step is similar to (2) with the pheromone deposit defined as:

$$\Delta\tau_{ij} = \begin{cases} J^{-1}(s)\boldsymbol{\beta}(\mathbf{x}) & , \quad \text{if} \quad c_{ij} \in s \in \mathcal{S}_{\text{trial}} \\ 0 & , \quad \text{otherwise,} \end{cases}$$

with $J(s)$ the cost of the sequence of states and actions, according to (4).

As explained in Section V-B, for optimal control problems, the appropriate update rule is to use all solutions by all ants in the trial $\mathcal{S}_{\text{trial}}$. In the fuzzy case, the solutions $s \in \mathcal{S}_{\text{trial}}$ consist of sequences of states and actions and the states can be fuzzified so that they are represented by sequences of vectors of degrees of fulfillment $\boldsymbol{\beta}$. Instead of one pheromone level, in the fuzzy case a set of pheromone levels are updated to a certain degree. It can be easily seen that as this update process is just a series of pheromone *deposits*, the final value of the pheromone levels relates to the sum of these deposits and is invariant with respect to the order of these deposits. This is also the case for this step in the AS or ACS algorithm.

VI. Example: Navigation with Variable Damping

This section presents an example application of the Fuzzy ACO algorithm to a continuous-state dynamic system. The dynamic system under consideration is a simulated two-dimensional (2D) navigation problem and similar to the one described in [19]. Note that it is not our purpose to demonstrate the superiority of Fuzzy ACO over any other method for this specific problem. Rather we want to demonstrate the functioning of the algorithm.

A. Problem Formulation

A vehicle, modeled as a point-mass of 1 kg, has to be steered to the origin of a two-dimensional surface from any given initial position in an optimal manner. The vehicle experiences a damping that varies non-linearly over the surface. The state of the vehicle is defined as $\mathbf{x} = \begin{bmatrix} c_1 & v_1 & c_2 & v_2 \end{bmatrix}^{\mathrm{T}}$, with c_1, c_2 and v_1, v_2 the position and velocity in the direction of each of the two principal axes respectively. The control input to the system $\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^{\mathrm{T}}$ is a two-dimensional force. The dynamics are:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -b(c_1,c_2) & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -b(c_1,c_2) \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{u},$$

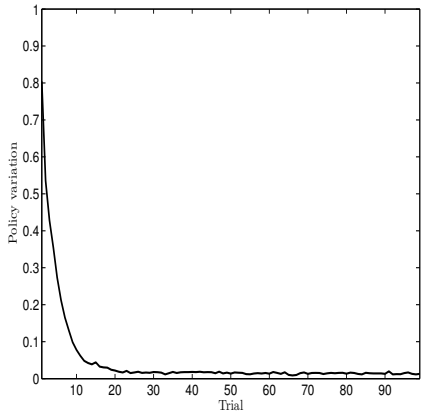
where the damping $b(c_1, c_2)$ in the experiments is modeled by an affine sum of two Gaussian functions, with means $(0, -2.3)$ and $(4.7, 1)$ and standard deviations $(2.5, 1.5)$ and $(1.5, 2)$ respectively. The damping profile can be seen in Fig. 1(b), where darker shading means more damping.

B. Fuzzy ACO Setup and Parameters

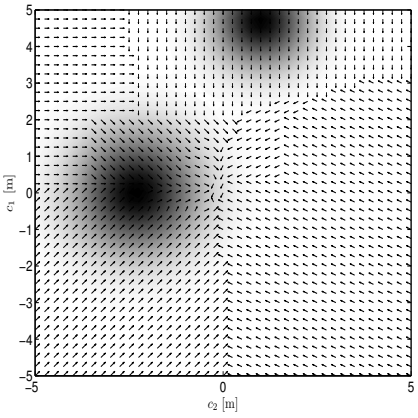
The cores of the membership functions of the positions c_1, c_2 are chosen to be $\{-5, -3.5, -2, -0.5, 0, 0.5, 2, 3.5, 5\}$ and those for the velocities v_1, v_2 are $\{-2, -0.5, 0, 0.5, 2\}$. The action set contains of 25 actions, namely the cross-product of the sets $\{-1, -0.5, 0, 0.5, 1\}$ for both dimensions. The local and global pheromone decay factors are respectively $\gamma = 0.01$ and $\lambda = 0.1$. Furthermore, $\alpha = 3$ and the number of ants is 2000. The sampling time is $T_s = 0.2$ and the ants are randomly initialized over the complete state space at the start of each trial. An ant terminates its trial when its position and velocity in both dimensions are within a bound of ± 0.25 and ± 0.05 from the goal respectively.

C. Simulation Results

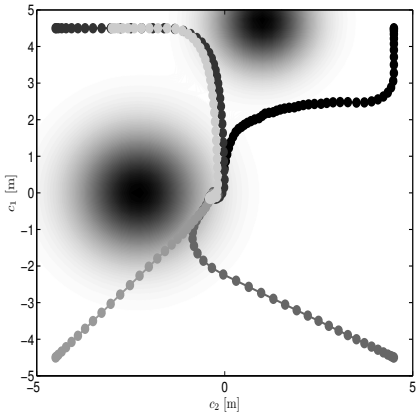
The convergence of the Fuzzy ACO algorithm is depicted in Fig. 1(a). It shows that the relative variation of the policy is already very low after about 20 trials. A slice of resulted policy for zero velocity is depicted together with the damping profile in Fig. 1(b). The policy shows the mapping of the positions in both dimensions to the input on a fine grid. Fig. 1(c) presents the trajectories of the vehicle for various initial positions and zero initial velocity. It shows that the vehicles manage to drive quickly to the goal, while avoiding the regions of stronger damping to a certain extent. However, the trajectories are only close to optimal. Especially for the



(a) Convergence of the algorithm in terms of the fraction of cores of the membership functions for which the policy changed at the end of the trial.



(b) A slice of the resulting policy for zero velocity. It shows the control input for a fine grid of positions. The damping profile is shown, where darker shades mean more damping.



(c) Trajectories of the vehicle under the resulted policy for various initial positions and zero velocity. The markers indicate the positions at the sampling instances.

Fig. 1. Results.

case where the vehicle starts in the bottom-left corner, the optimality of the trajectory can be questioned as the vehicle drives straight to the goal, without avoiding the region of larger damping at all. These results demonstrate that the algorithm is capable of converging quickly, but only to a suboptimal policy with the settings used in the experiments.

VII. Conclusions and Future Work

This paper has introduced the Fuzzy ACO algorithm for optimal control problems, which combines the framework of the AS and ACS algorithms with a fuzzy partitioning of the state space. The applicability of this algorithm to optimal control problems with continuous-valued states is outlined and demonstrated on the non-linear control problem of two-dimensional navigation with variable damping. The results show convergence of the algorithm to a suboptimal policy that drives the vehicle to the goal for any initial state. Future research must further develop the algorithm to deal with suboptimality in a better way and to theoretically prove its convergence.

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, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[3] 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.

[4] T. Stutzle and U. Hoos, "MAX MIN Ant System," *Journal of Future Generation Computer Systems*, vol. 16, pp. 889–914, 2000.

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

[6] 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*, Big Island, HI, USA, October 2005, pp. 288–292.

[7] 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.

[8] 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)*, Singapore, November 2004, pp. 313–318.

[9] 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.

[10] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, no. 3, pp. 1155–1173, 2008.

[11] 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.

[12] 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)*, September 2005, pp. 880–887.

[13] 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.

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

[15] J. Casillas, O. Cordón, and F. Herrera, "Learning fuzzy rule-based systems using ant colony optimization algorithms," in *Proceedings of the ANTS'2000. From Ant Colonies to Artificial Ants: Second International Workshop on Ant Algorithms. Brussels (Belgium)*, September 2000, pp. 13–21.

[16] B. Zhao and S. Li, "Design of a fuzzy logic controller by ant colony algorithm with application to an inverted pendulum system," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, 2006, pp. 3790–3794.

[17] W. Zhu, J. Chen, and B. Zhu, "Optimal design of fuzzy controller based on ant colony algorithms," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*, 2006, pp. 1603–1607.

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

[19] L. Busoniu, D. Ernst, B. De Schutter, and R. Babuška, "Continuous-state reinforcement learning with fuzzy approximation," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 38, pp. 156–172, 2008.