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

Multiresolution State-Space Discretization Method for Q-Learning

Amanda Lampton and John Valasek
Texas A&M University, College Station, Texas 77843-3141

***Abstract*— For large scale problems Q-Learning often suffers from the Curse of Dimensionality due to large numbers of possible state-action pairs. This paper develops a multiresolution state-space discretization method for the episodic unsupervised learning method of Q-Learning, in which a state-space is adaptively discretized by progressively finer grids around the areas of interest within the state or learning space. Optimality of the learning algorithm is addressed by a cost function. Applied to a morphing airfoil with two morphing parameters (two state variables), it is shown that by setting the multiresolution method to define the area of interest by the goal the agent seeks, this method can learn a specific goal within ± 0.002, while reducing the total number of state-action pairs need to achieve this level of specificity by almost 90%.**

I. INTRODUCTION

For the computational reinforcement learning problem, the learned value or action-value function is generally a good representation of an agent's knowledge of the environment. The problem becomes more complex as the number of state variables needed to represent the environment increases, so discretizing the state and action spaces is a common way to cast a continuous state and action space problem as a reinforcement learning problem. Thus a simple learning problem can be easily discretized into a relatively small number of states. The number of states in the action-value function depends on how a problem is discretized, but there is a trade off. If the agent can only store knowledge in a small number of states, important details of the environment may be lost. If the agent can store knowledge in a very large number of states, details of the environment are captured quite well. The caveat is that the rate of convergence drops drastically as the number of states increases. Examples of state-space discretization include [1], which describes a space robot problem in which the orientation and the action set of the spacecraft has been discretized to facilitate learning, and [2], which describes quad-Q-learning in which a state-space is discretized and then sampled in a "divide and conquer" technique.

Reinforcement learning has also been used for the problem of shape changing or morphing. Reference [3] developed a methodology that combines Structured Adaptive Model Inversion (SAMI) with Reinforcement Learning to address the optimal shape change of an entire vehicle. The method learns the commands for two independent morphing parameters that produce the optimal shape, and the authors demonstrate learning of the required shape and morphing into it while accurately tracking a specified reference trajectory. The methodology is further improved upon by applying Sequential Function Approximation to generalize the learning from previously experienced quantized states and actions to the continuous state-action space.[4] This approximation scheme resulted in marked improvements in the learning as opposed to the previously employed K-Nearest Neighbor approach. All of these examples only have two independent degrees-of-freedom that must be learned, and manipulating more degrees-of-freedom creates a more complex problem with increased dimensionality and significant convergence issues.

This paper proposes and develops a multiresolution state-space discretization method that incorporates the convergence benefits of a coarse discretization of the state-space as well as the learning of the finer details, such as goal location, of a fine state-space discretization. The method mimics the natural tendency of people and animals to learn the broader goal before focusing in on more specific goals. This method is applied to the morphing airfoil architecture developed in References [5], [6], and [7]. Reinforcement learning is used to learn the commands that produce the optimal shape based on airfoil lift coefficient. The levels of discretization of the state-space are tuned such that good convergence and attention to detail is achieved. The contribution of this paper is a new discretization method that allows the learning to converge quickly while still maintaining a high level of detail around areas of high information content in the environment.

II. REINFORCEMENT LEARNING

Reinforcement learning (RL) is a method of learning from interaction between an agent and its environment to achieve a goal. The learner and decision-maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. The agent interacts with its environment at each instance of a sequence of discrete time steps, $t = 0, 1, 2, 3....$ At each time step t, the agent receives some representation of the environment's state, $s_t \in S$, where S is a set of possible states, and on that basis it selects an action, $a_t \in A(s_t)$, where $A(s_t)$ is a set of actions available in state s(t). One time step later, partially as a consequence of its action, the agent receives a numerical reward, $r_{t+1} = R$, and finds itself in a new state, s_{t+1}. The mapping from states to probabilities of selecting each possible action at each time step, denoted by π is called

Graduate Research Assistant, Vehicle Systems & Control Laboratory, Aerospace Engineering Department, Student Member AIAA, alampton@tamu.edu.

Associate Professor and Director, Vehicle Systems & Control Laboratory, Aerospace Engineering Department, Associate Fellow AIAA, valasek@tamu.edu.

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

the agent's policy, where $\pi_t(s,a)$ indicates the probability that $a_t = a$ given $s_t = s$ at time t. Reinforcement learning methods specify how the agent changes its policy as a result of its experiences. The agent's goal is to maximize the total amount of reward it receives over the long run.

Q-Learning, a reinforcement learning algorithm, is a form of the successive approximations technique of Dynamic Programming, first proposed and developed by Watkins [8] that learns the optimal value functions directly, as opposed to fixing a policy and determining the corresponding value functions, like Temporal-Differences. It is the first provably convergent direct adaptive optimal control algorithm. Here we use a 1-step Q-learning method, which is a common off-policy Temporal Difference (TD) control algorithm. In its simplest form it is defined by

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left\{ r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right\} \tag{1}$$

The Q-learning algorithm is illustrated as follows:[9]

Q-Learning()

- Initialize $Q(s,a)$ arbitrarily
- Repeat (for each episode)
  - Initialize s
  - Repeat (for each step of the episode)
    - Choose a from s using policy derived from $Q(s,a)$ (e.g. ϵ-Greedy Policy)
    - Take action a, observe r, s'
    - $Q(s,a) \leftarrow Q(s,a) + \alpha \left\{ r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right\}$
    - $s \leftarrow s'$
  - until s is terminal
- return $Q(s,a)$

The agent learns the greedy policy, and as the learning episodes increase, the learned action-value function $Q(s,a)$ converges asymptotically to the optimal action-value function $Q^*(s,a)$. The method is an off-policy one as it evaluates the target policy (the greedy policy) while following another policy. The policy used in updating $Q(s,a)$ can be a random policy, for example, with each action having the same probability of being selected.

If the number of the states and the actions of a RL problem is a small value, its $Q(s,a)$ can be represented using a table, where the action-value for each state-action pair is stored in one entity of the table. For RL problems with states on continuous domains, it is impossible to enumerate the action-value for each state-action pair because there are an infinite number of state-action pairs. One commonly used solution is to artificially quantize the states into discrete sets, thereby reducing the number of state-action pairs the agent must visit and learn. The goal in doing this is to reduce the number of state-action pairs while maintaining the integrity of the learned action-value function. This paper realizes this goal with a multiresolution state-space discretization method that keeps the number of state-action pairs manageable and the details intact, even when more state variables are added.

III. LEARNING ON A 2- AND N-DIMENSIONAL CONTINUOUS DOMAIN

Q-learning on a continuous domain quickly becomes intractable when one considers that convergence of the algorithm to the optimal action-value function is only guaranteed if the agent visits every possible state an infinite number of times.[8] An agent would therefore visit an infinite number of states using an infinite number of actions an infinite number of times. Add in the fact that the states can be defined by anywhere from 1 to N continuous variables and the so-called "Curse of Dimensionality" becomes a significant problem.

One way to cope with the inherent complexity of a continuous domain learning problem is to discretize the state-space by overlaying a pseudo-grid. The essential ideas of this concept can be best introduced in terms of a 1-dimensional problem. The notation can then be generalized for the 2- and N-dimensional problems. For the 1-dimensional problem the state-space can be represented by a line as seen in Fig. 1. An arbitrary set of vertices $\left\{ {}^1X, {}^2X, \ldots, {}^kX, \ldots \right\}$ are introduced at a uniform distance h apart. Ideally, h is chosen such that a vertex lies on both end points of the state-space. In the learning algorithm the agent is only allowed to visit the overlaying vertices and their corresponding states. This technique effectively reduces the state-space from infinity to a finite number of states, thus rendering the problem more manageable.

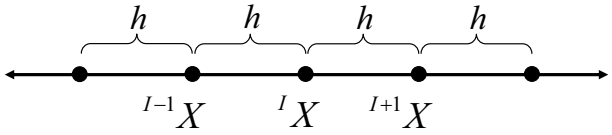

Fig. 1. 1-Dimensional State Space with Overlaying Pseudogrid

To further simplify the problem, we restrict what actions the agent may take. When the agent is at the I^{th} vertex $X = {}^IX$, it may only move to ${}^{I-1}X$ or ${}^{I+1}X$. Now the problem only has two possible actions rather than an infinite number, which further reduced the problem complexity.

Let L denote the length of the continuous domain. As per our formulation there are

$$N_{V_1} = \frac{L}{h} + 1 \tag{2}$$

vertices, where N_V is the number of vertices, and 2 actions. Therefore, there are only

$$N_1 = 2\left(\frac{L}{h} + 1\right) \tag{3}$$

state-action pairs, where N is the number of state-action pairs. The 2-dimensional problem can be represented in a similar manner. In this case the state-space is represented by Fig. 2. An arbitrary set of vertices $\left\{ {}^{11}X, {}^{12}X, \ldots, {}^{ij}X, \ldots \right\}$ are again introduced at uniform distances h_{x_1} or h_{x_2} apart. The actions available to the agent are again restricted as in the 1-dimensional case. For the 2-dimensional case, when the

agent is at the IJ^{th} vertex $X = {}^{IJ}X$, it may only move to vertices ${}^{(I-1)J}X$, ${}^{(I+1)J}X$, ${}^{I(J-1)}X$, and ${}^{I(J+1)}X$, a total of 4, or $2 * 2$, actions.

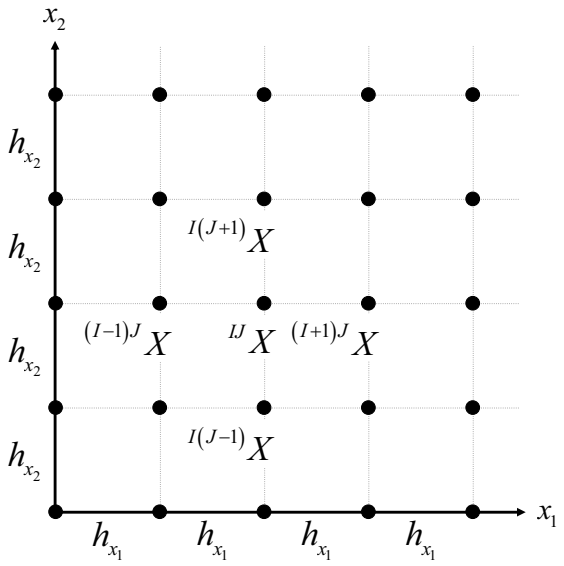

Fig. 2. 2-Dimensional State Space with Overlaying Pseudogrid

This problem is more complex than the previous one, yet it is still simpler than a 2-dimensional continuous state-space problem. For this 2-dimensional discrete case, let L_{x_1} and L_{x_2} denote the length in the x_1- and x_2-direction, respectively, of the continuous domain. This results in

$$\begin{aligned} N_{V_2} &= \left(\frac{L_{x_1}}{h_{x_1}} + 1\right)\left(\frac{L_{x_2}}{h_{x_2}} + 1\right) \\ &= \prod_{i=1}^{2}\left(\frac{L_{x_i}}{h_{x_i}} + 1\right) \end{aligned} \tag{4}$$

vertices. Therefore, there are

$$N_2 = 2 * 2\prod_{i=1}^{2}\left(\frac{L_{x_i}}{h_{x_i}} + 1\right) \tag{5}$$

state-action pairs. This 2-dimensional development is what will be used in the rest of this paper.

From here the formulation can be generalized to the N-dimensional case. For an N-dimensional continuous state-space, an arbitrary set of vertices $\{{}^{11\ldots1}X, {}^{11\ldots2}X, \ldots, {}^{NN\ldots N}X\}$ are introduced at uniform distances h_{x_1}, h_{x_2}, …, h_{x_N} apart. The actions are restricted to the two nearest vertices in any direction from the current vertex $X = {}^{IJ\cdots}X$, yielding a total of $2N$ actions available to the agent from any given vertex.

Now let L_{x_1}, L_{x_2}, …, L_{x_N} denote the length in the x_1-, x_2-, …, and x_N-directions, respectively. As a result there are

$$N_{V_N} = \prod_{i=1}^{N}\left(\frac{L_{x_i}}{h_{x_i}} + 1\right) \tag{6}$$

vertices. Therefore, there are

$$N_N = 2N\prod_{i=1}^{N}\left(\frac{L_{x_i}}{h_{x_i}} + 1\right) \tag{7}$$

state-action pairs.

Discretizing the domain in this way can greatly simplify a learning problem. Intuitively, the larger h_{x_i} is, the fewer the number of vertices, resulting in fewer visits by the agent necessary to learn the policy correctly. Special care must be taken, however, in the choice of h_{x_i} and the definition of the goal the agent attempts to attain. If the only goal state lies between vertices, then the agent will be unable to learn the actions necessary to reach the goal state. The "Curse of Dimensionality" can still become a problem when using this technique. As N increases, the number of state-action pairs increases quickly. Manipulation of h_{x_i} can alleviate some problems, but can eventually become overwhelmed. However, the number of state-action pairs remains finite. In this paper a 2-dimensional problem is analyzed.

IV. MULTIRESOLUTION DISCRETIZATION FOR N-DIMENSIONS

Discretizing a state-space for learning is beneficial in that it creates a finite number of state-action pairs the agent must visit. Generally, as the number of state-action pairs decreases, the rate of convergence increases.[6] However, fewer state-action pairs captures less detail of the environment. Also, using the method described in Section III limits the agent to the vertices. It is entirely possible that the goal the agent is seeking, or any other area of interest, does not lie on a vertex. This necessitates adding a range to the goal that encompasses one or more of the vertices in the state-space. These vertices within the goal range are pseudo-goals. (Fig. 3)

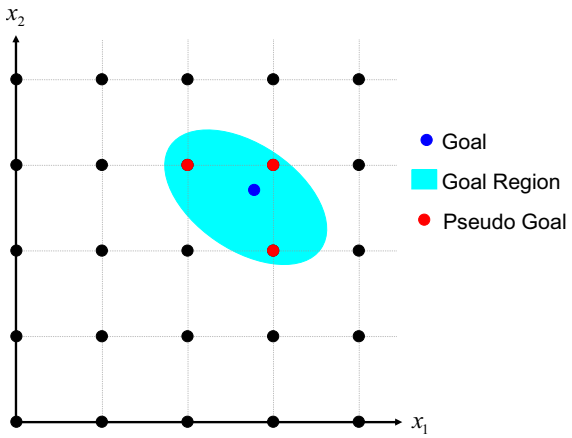

Fig. 3. Multiresolution State Space Discretization – Phase 1: Coarse Grid, Large Goal Range

As the agent explores the coarsely discretized state-space and garners rewards, it also notes the location of the pseudo-goals. Once learning on the current discretization has converged, the area surrounding and encompassing the psuedo-goals is re-discretized to a finer resolution such that

$h_{x_{i_2}} < h_{x_{i_1}}$, where the subscript 1 denotes the initial discretization, and subscript 2 denotes the second discretization. A new, smaller range is defined for the goal and learning begins anew in the smaller state-space. Fig. 4 shows the re-discretization of the state-space.

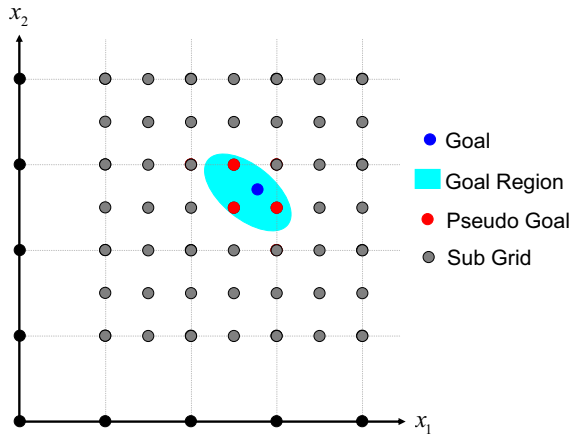

Fig. 4. Multiresolution State Space Discretization – Phase 2: Finer Grid, Smaller Goal Range

This method can then be generalized for the N-dimensional case. Let $L^j_{x_1}$, $L^j_{x_2}$, ..., $L^j_{x_N}$ denote the length in the x_1-, x_2-, ..., and x_N-directions, respectively, and the superscript j denote the resolution of the discretization in which 1 is the coarsest and M is the finest. The vertices for each resolution are then set at distances $h^j_{x_1}$, $h^j_{x_2}$, ..., $h^j_{x_N}$ apart. These terms effectively define the fineness of resolution level j. Eqs. 6 and 7 can then be modified to calculate the number of vertices and state action pairs for this method, as shown in Eqs. 8 and 9.

When the multiresolution learning is complete, there are

$$N_{V_N} = \sum_{j=1}^{M}\left(\prod_{i=1}^{N}\left(\frac{L^j_{x_i}}{h^j_{x_i}}+1\right)\right) - \sum_{j=1}^{M-1}\left(\prod_{i=1}^{N}\left(\frac{L^{j+1}_{x_i}}{h^j_{x_i}}+1\right)\right) \tag{8}$$

vertices. Therefore, there are

$$N_N = 2N\left(\sum_{j=1}^{M}\left(\prod_{i=1}^{N}\left(\frac{L^j_{x_i}}{h^j_{x_i}}+1\right)\right) - \sum_{j=1}^{M-1}\left(\prod_{i=1}^{N}\left(\frac{L^{j+1}_{x_i}}{h^j_{x_i}}+1\right)\right)\right) \tag{9}$$

state-action pairs. Notice the second term of each equation excises the duplicate vertices from one level of discretization to the next. Also note that if the full state-space were simply discretized by the finest level of $h^M_{x_i}$, there would be

$$N_{V_{N_{fine}}} = \prod_{i=1}^{N}\left(\frac{L^1_{x_i}}{h^M_{x_i}}+1\right) \tag{10}$$

vertices and

$$N_{N_{fine}} = 2N\left(\prod_{i=1}^{N}\left(\frac{L^1_{x_i}}{h^M_{x_i}}+1\right)\right) \tag{11}$$

state-action pairs. It can be shown that $N_{V_N} < N_{V_{N_{fine}}}$ and $N_N < N_{N_{fine}}$ by a significant amount, the magnitude of which is determined by the factor by which each subsequent descretization is reduced from the previous. It is known that the time to convergence for Q-learning increases exponentially as the complexity of the problem, i.e. state-action pairs, increases. This method reduces a learning problem to a series of smaller learning problems with relatively few state-action pairs, on the order of several orders of magnitude less. Rather than one large problem that will take a great deal of time to converge, there are several quickly converging smaller problems.

V. MORPHING AIRFOIL

For this problem the RL agent endeavors to learn, from its interaction with the environment, the optimal policy that commands the series of actions which change the morphing airfoil's shape toward an optimal one (Fig. 5). Optimality is defined by the airfoil lift coefficient. The environment is the resulting aerodynamics the airfoil is subjected to. It is assumed here that the RL agent has no prior knowledge of the relationship between actions and states, but the RL agent does know all possible actions that can be applied, and has accurate, real-time information of the morphing airfoil shape, the present aerodynamics, and the current reward provided by the environment.

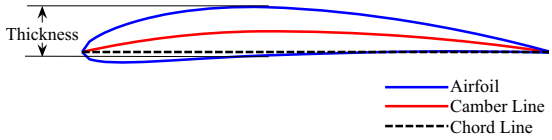

Fig. 5. Representative Airfoil

The airfoil aerodynamics are modeled with a constant strength doublet panel method with four degrees-of-freedom: Airfoil thickness, Camber, Location of maximum camber, and Airfoil angle-of-attack. Only thickness and camber degrees-of-freedom are used here. For simulation purposes, the morphing dynamics are assumed to be simple nonlinear differential equations. A full description of the airfoil model development can be found in Reference [5]. Validation and verification of the airfoil model can also be found in Reference [5].

A. Implementation

The aerodynamic model and the reinforcement learning agent interact significantly during both the learning stage, when the optimal shape is learned, and the operational stage, when the airfoil morphs from state to state. Initially, the reinforcement learning agent commands a random action from the set of admissible actions. As described in Section III, the admissible actions are restricted to movement to the two closest vertices in any given direction from the current vertex. For example, the agent chooses to move in the x_1-direction from vertex ${}^{IJ}X$ in the 2-dimensional problem. For the initial discretization, the two possible actions in the

TABLE I

DISTANCE BETWEEN ADJACENT VERTICES

Parameter	Value
$h^1_{x_1}$	0.50
$h^1_{x_2}$	0.50
g_f	0.20
M	3

TABLE II

AXIS DEFINITIONS

x_i	Definition
x_1	Thickness (%)
x_2	Camber (%)

x_1-direction are defined as follows

$$\begin{aligned} A^1_{11} &\equiv {}^{(I+1)J}X - {}^{IJ}X = h^1_{x_1} \\ A^1_{12} &\equiv {}^{(I-1)J}X - {}^{IJ}X = -h^1_{x_1} \end{aligned} \tag{12}$$

Eq. 12 can be summarized by saying the initial admissible actions in the x_1-direction are $A_1 = \pm h^1_{x_1}$. Similar relationships can be found for the x_2-direction. Admissible actions in the other direction is $A_2 = \pm h^1_{x_2}$. Each finer discretization is a predetermined factor, g_f, applied to the coarser discretization, such that $h^{j+1}_{x_i} = g_f h^j_{x_i}$. The number of levels of discretization or resolution as defined earlier is M. These parameters are listed in Table I, and the definitions of the x_i axes are defined in Table II.

To read these tables consider the x_1-direction, for example. The agent changes $\pm 0.50\%$ of the chord in thickness in this direction when $h_{x_1} = 0.50\%$.

The agent implements an action by submitting it to the plant, which produces a shape change. The reward associated with the resultant shape is evaluated. The resulting state, action, and reward set is then stored in a database. Then a new action is chosen, and the sequence repeats itself for some predefined number of episodes or until the agent reaches a goal state. Shape changes in the airfoil due to actions generated by the reinforcement learning agent cause the aerodynamics associated with the airfoil to change. The aerodynamic properties of the airfoil define the reward, as stated, and the structural analysis offers a constraint on the limits of the morphing degrees of freedom. Once the learning converges or the predefined number of learning episodes elapses, the state-space is reduced to the area around the area of interest, i.e. the goal, the range and admissible actions are redefined, and a new round of learning commences.

VI. NUMERICAL EXAMPLE

The numerical example demonstrates the multiresolution state-space discretization method for learning and updating the action-value function as the discretization becomes increasingly finer. The agent is allowed 5000 episodes with which to explore the state-space of thickness-camber combinations for each level of discretization. The reward the agent receives is calculated by

$$r = |g - c_{n-1}| - |g - c_n| \tag{13}$$

TABLE III

INITIAL AIRFOIL THICKNESS AND CAMBER LIMITS

Initial Limit	Lower	Upper
Thickness (% chord)	10	18
Camber (% chord)	0	5

TABLE IV

STATES AND STATE-ACTION PAIRS

	States	State-Action Pairs
Multiresolution	8881	35524
Single-Resolution	100651	402604

where r is the reward, g is the goal, and c is the metric. For this example, c is the lift coefficient, c_l, of the airfoil and $g := c_l = 0.2$. The area of interest is the goal and the associated initial range is ± 0.05. The initial boundary limits of the state-space are listed in Table III

Learning performance is measured in three ways. First the dimensionality of the multiresolution action-value function is compared with that of the full state-space discretized at the finest level. Next Monte Carlo simulation results are analyzed. Finally, the final value function and policy is considered.

A. Dimensionality

The number of states and state-action pairs for both the actual learned multiresolution problem and the hypothetical single-resolution problem described by Eqs. 10 and 11 are shown in Table IV.

The multiresolution method decreases the number of state-action pairs the agent must visit by an entire order of magnitude. This greatly simplifies the learning problem and encourages faster convergence.

B. Monte Carlo Simulation

The action-value function is analyzed by conducting a set of simulations using the learned function. The agent is initialized in a random, non-goal state, i.e. outside the target range, and allowed to exploit its knowledge to navigate through the domain to find the goal. The agent retains a 5% probability of choosing a random action and exploring the state-space. During the learning process the action-value function is recorded every 200 episodes. The simulation is run 500 times for each recorded action-value function to accrue enough data to get an accurate measure of the success or failure of the agent and the learning algorithm as the learning is refined. A success occurs when the agent navigates from the random initial state to a goal state without encountering a boundary. A failure occurs when the agent either encounters a boundary or gets "lost" and wanders around the state-space.

Fig. 6 shows the results of the Monte Carlo Simulation. Each set of 5000 episodes is for a different level of discretization, from coarsest to finest. The final star at 15000 episodes is for the simulation using the full power of the multiresolution state-space discretization method. The final range for the goal in this problem is 0.002. This figure shows

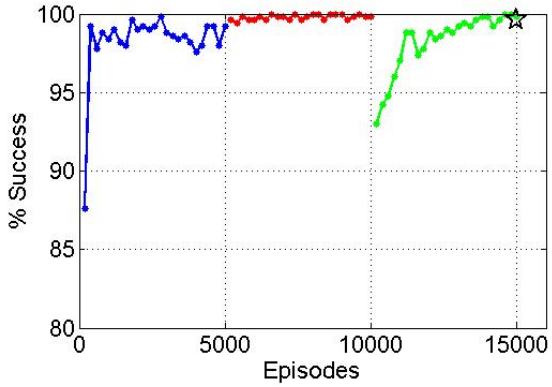

Fig. 6. Monte Carlo Simulation Results

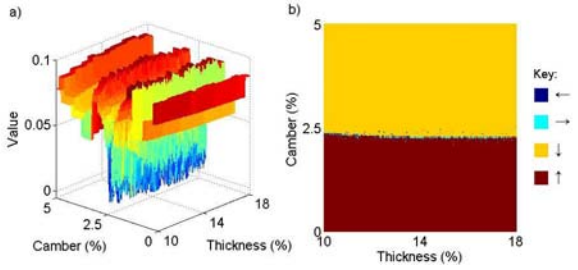

Fig. 7. Value Function (a) and Policy Representation (b)

that the first and second levels of discretization converge quickly. The third level takes several hundred episodes to converge above a 98% success rate. The multiresolution discretization converges within 15000 episodes. The final simulation shows that the agent can use its final set of information reach the goal with over a 99% success rate.

C. Value Function and Policy Analysis

It is helpful to consider a value function approximated from the learned action-value function. The value function can show how the learning differs when learning parameters or discretization is modified. The value function is calculated using the following equation.

$$V(s) \approx \max_a Q(s,a) \tag{14}$$

The policy can also be approximated by finding the action with maximum preference or action-value for each state. The policy is calculated using the following equation.

$$\pi(s) = \max_a Q(s,a) \tag{15}$$

Fig. 7 shows the final value function and greedy policy. Toward the upper and lower extremes of the camber axis, the function is very blocky. This is due to the large discretization during the initial learning. The sudden increase in value function as the camber approaches 2.5% is a result of learning on the second level of discretization. Notice the function is less blocky in this region. The minimal ridge in the middle of the function is effectively the goal of $c_l = 0.2$ and where the reward function approaches 0. This minimal ridge is reflected in the graphic of the greedy policy. This figure shows that change in camber has the greatest influence in affecting airfoil lift coefficient, enough such that thickness may not be a necessary morphing parameters in future iterations of this problem.

VII. CONCLUSIONS

The results show that the learning for the multiresolution method reaches a 98% success rate or greater within 200 episodes for the coarsest discretization and within 2500 episodes for the finest discretization. Conceptually, a problem with the full state-space discretized at the finest level would take many thousands or tens of thousands of episodes more to reach this level of convergence. This method is successful in greatly reducing the time for convergence, increasing the rate of convergence, and achieving a goal with the very small range of 0.002. This method essentially reduced the larger problem by almost 90%, making it a much more tractable learning problem.

VIII. ACKNOWLEDGMENTS

This work was sponsored (in part) by the Air Force Office of Scientific Research, USAF, under grant/contract number FA9550-08-1-0038. The technical monitor is Dr. William McEneaney. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research or the U.S. Government.

REFERENCES

[1] K. Senda, T. Matsumoto, Y. Okano, S. Mano, and S. Fujii, "Autonomous task achievement by space robot based on q-learning with environment recognition," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, no. AIAA-2003-5462, Austin, TX, 11-14 August 2003.

[2] C. Clausen and H. Wechsler, "Quad-q-learning," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 279–294, February 2000.

[3] J. Valasek, M. Tandale, and J. Rong, "A reinforcement learning - adaptive control architecture for morphing," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 4, pp. 174–195, April 2005.

[4] J. Valasek, J. Doebbler, M. Tandale, and A. Meade, "Improved adaptive-reinforcement learning control for morphing unmanned air vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Part B*, vol. 38, no. 4, pp. 1014–1020, August 2008.

[5] A. Lampton, A. Niksch, and J. Valasek, "Reinforcement learning of morphing airfoils with aerodynamic and structural effects," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, no. 1, pp. 30–50, January 2009.

[6] ——, "Reinforcement learning of a morphing airfoil-policy and discrete learning analysis," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, no. AIAA-2008-7281, Honolulu, HI, 18-21 August 2008.

[7] ——, "Morphing airfoil with reinforcement learning of four shape changing parameters," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, no. AIAA-2008-7282, Honolulu, HI, 18-21 August 2008.

[8] C. J. C. H. Watkins and P. Dayan, "Learning from delayed rewards," Ph.D. dissertation, University of Cambridge, Cambridge, UK, 1989.

[9] R. Sutton and A. Barto, *Reinforcement Learning - An Introduction*. Cambridge, Massachusetts: The MIT Press, 1998.