Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

Multiresolution State-Space Discretization Method for Q-Learning with Function Approximation and Policy Iteration

Amanda Lampton and John Valasek, *Senior Member, IEEE*
Department of Aerospace Engineering
Texas A&M University
College Station, TX

***Abstract*—A multiresolution state-space discretization method is developed for the episodic unsupervised learning method of Q-Learning. In addition, a genetic algorithm is used periodically during learning to approximate the action-value function. Policy iteration is added as a stopping criterion for the algorithm. 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 method whereby a state-space is adaptively discretized by progressively finer grids around the areas of interest within the state or learning space. Policy iteration is added to prevent unnecessary episodes at each level of discretization once the learning has converged. Utility of the method is demonstrated with application to the problem of a morphing airfoil with two morphing parameters (two state variables). By setting the multiresolution method to define the area of interest by the goal the agent seeks, it is shown that this method can learn a specific goal within ± 0.002, while reducing the total number episodes needed to converge by 85% from the allotted total possible episodes. It is also shown that a good approximation of the action-value function is produced with 80% agreement between the tabulated and approximated policy, though empirically the approximated policy appears to be superior.**

***Index Terms*—Q-learning, Multiresolution, Genetic Algorithm, Function Approximation, Policy Iteration.**

I. Introduction

For the computational reinforcement learning problem, discretizing the state and action spaces and subsequently performing function approximation on the learned data is a common way to cast a continuous state and action space problem as a reinforcement learning problem. A simple learning problem can be easily discretized into a relatively small number of states. The learned value or action-value function is generally a good representation of the agent's knowledge of the environment. A problem becomes more complex as the number of state variables needed to represent the environment increases. The number of states in the action-value function depends on how a problem is discretized. There is a trade off, however. 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 Reference [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 Reference [2], which describes quad-Q-learning in which a state-space is discretized and then sampled in a "divide and conquer" technique.

Function approximation of the action-value function serves to replace the potentially large discrete lookup table with a set of continuous basis functions and weights. Generally, function approximation is performed on the final function with methods such as GLOMAP[3], Sequential Function Approximation (SFA)[4], [5], least-squares, *etc.* Other methods have been developed recently that actually learn using a least-squares approximation. Lagoudakis and Parr adapted these methods for Q-learning in the form of Least-Squares Temporal Difference Q-learning (LSTD-Q or LSQ) and extended to include policy iteration in the form of Least-Squares Policy Iteration (LSPI).[6] In these methods the action-value function or the value function is approximated with a set of basis functions chosen *a priori*, and essentially the weights are 'learned'.

This paper proposes 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 [7], [8], and [9]. 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 must be tuned such that good convergence and attention to detail is achieved. The contribution of this paper is to develop a new discretization method that allows the learning to converge quickly while still maintaining a high level of detail around areas of interest in the environment.

In addition, this paper proposes a new method of using a genetic algorithm to aid in function approximation of the action-value function during learning. By periodically computing the function approximation and monitoring the convergence of the policy represented by the approximated action-value function, not only can an approximation that preserves the policy be

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

found, so too can the number of learning episodes needed for convergence be kept to a minimum.

This paper is organized as follows. Section II describes the mechanics of reinforcement learning and how it is implemented in Q-learning in particular. Reinforcement learning learns the optimality relations between the aerodynamic requirements and the shape. The airfoil can then be subjected to a series of aerodynamics requirements and use the relations learned to choose a good shape for the current set of requirements. The method used to discretize a continuous learning domain is developed. Section III and Section IV describe the discretization method and the multiresolution state-space discretization method, respectively. Section V overviews the genetic algorithm used for function approximation, and Section VI describes how policy iteration is conducted in the algorithm. Section VII briefly describes the airfoil model used by the reinforcement learning agent. This section also describes how the airfoil model and the reinforcement learning agent are tied together to form a morphing airfoil. Section VII then takes the fully developed multiresolution state-space discretization method with the new additions of the genetic algorithm and policy iteration, applies it to the morphing airfoil, and interprets a numerical example generated from it. Finally, conclusions are drawn from the numerical example in Section VIII.

II. Rienforcement 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 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.[10] Q-learning learns the optimal value functions directly, as opposed to fixing a policy and determining the corresponding value functions, like Temporal-Differences. It automatically focuses attention to where it is needed, thereby avoiding the need to sweep over the state-action space. Additionally, it is the first provably convergent direct adaptive optimal control algorithm.

For the present research, the agent in the morphing airfoil problem is its RL agent. It attempts to independently maneuver from some initial state to a final goal state characterized by the aerodynamic properties of the airfoil. To reach this goal, it endeavors to learn, from its interaction with the environment, the optimal policy that, given the specific aerodynamic requirements, commands the series of actions that changes the morphing airfoil's thickness or camber toward an optimal one. The environment is the resulting aerodynamics the airfoil is subjected to. It is assumed that the RL agent has no prior knowledge of the relationship between actions and the thickness and camber of the morphing airfoil. However, the RL agent does know all possible actions that can be applied. It has accurate, real-time information of the morphing airfoil shape, the present aerodynamics, and the current reward provided by the environment.

The RL agent uses 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:[11]

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 $Q(s, a)$

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, with each action having the same probability of being selected. The other option is an ϵ-greedy policy, where ϵ is a small value. The action a with the maximum $Q(s, a)$ is selected with probability 1-ϵ, otherwise a random action is 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. Since the RL problem for the morphing vehicle has states (the shape of the airfoil) on continuous domains, it is impossible to enumerate the action-value for each state-action pair. In essence, 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 seeks to achieve this through a multiresolution state-space discretization method. For the problem at hand, this becomes most important for keeping the number of state-action pairs manageable and the details intact when more state variables are added in the form of other morphing parameters.

III. Learning on a 2-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.[10] 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 introduced for the 2-dimensional problems.

The 2-dimensional problem can be represented by a 2-dimensional plane as represented by Fig. 1. An arbitrary set of vertices $\left\{{}^{11}X, {}^{12}X, \ldots, {}^{ij}X, \ldots\right\}$ are introduced at uniform distances h_{x_1} or h_{x_2} apart. In the learning algorithm the agent is only allowed to visit the overlaying vertices and their corresponding states. 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. This technique effectively reduces the state-space from infinity to a finite number of states, thus rendering the problem more manageable.

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

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

vertices, where N_{V_2} is the number of vertices. Therefore, there are

$$N_2 = 2 * 2\prod_{i=1}^{2}\left(\frac{L_{x_i}}{h_{x_i}} + 1\right) \tag{3}$$

state-action pairs, where N is the number of 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.

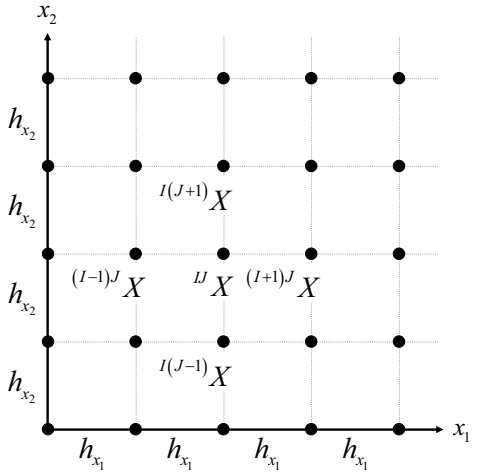

Fig. 1. 2-Dimensional State Space with Overlaying Pseudogrid

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.[8] 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. 2)

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. 3 shows the re-discretization of the state-space.

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

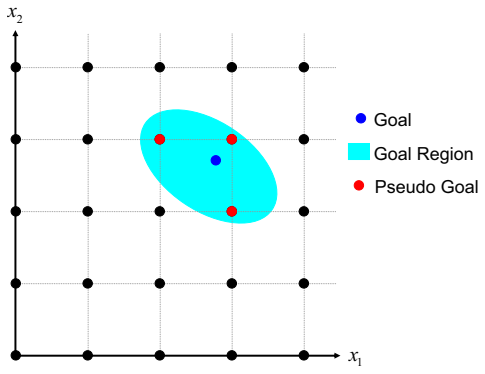

Fig. 2. Multiresolution State Space Discretization – Phase 1: Coarse Grid, Large Goal Range

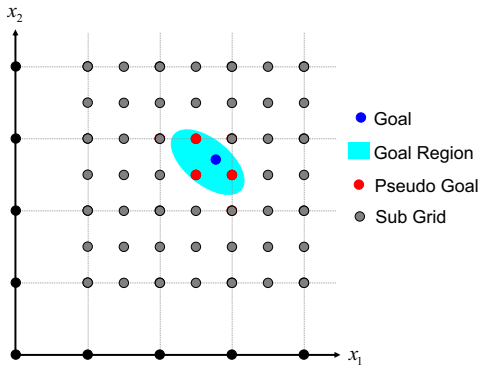

Fig. 3. Multiresolution State Space Discretization – Phase 2: Finer Grid, Smaller Goal Range

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 $h^j_{x_1}, h^j_{x_2}, \ldots, h^j_{x_N}$ apart. These terms effectively define the fineness of resolution level j. Eqs. 2 and 3 can then be modified to calculate the number of vertices and state action pairs for this method, as shown in Eqs. 4 and 5.

When the multiresolution learning is complete, there are

$$N_{V_N} = \sum_{j=1}^{M}\left(\prod_{i=1}^{N}\left(\frac{L^j_{x_i}}{h^j_{x_i}}+1\right)\right) - \sum_{j=1}^{M-1}\left(\prod_{i=1}^{N}\left(\frac{L^{j+1}_{x_i}}{h^j_{x_i}}+1\right)\right) \tag{4}$$

vertices. Therefore, there are

$$N_N = 2N\left(\sum_{j=1}^{M}\left(\prod_{i=1}^{N}\left(\frac{L^j_{x_i}}{h^j_{x_i}}+1\right)\right) - \sum_{j=1}^{M-1}\left(\prod_{i=1}^{N}\left(\frac{L^{j+1}_{x_i}}{h^j_{x_i}}+1\right)\right)\right) \tag{5}$$

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

$$N_{V_{N_{fine}}} = \prod_{i=1}^{N}\left(\frac{L^1_{x_i}}{h^M_{x_i}}+1\right) \tag{6}$$

vertices and

$$N_{N_{fine}} = 2N\left(\prod_{i=1}^{N}\left(\frac{L^1_{x_i}}{h^M_{x_i}}+1\right)\right) \tag{7}$$

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. Genetic Algorithm for Action-Value Function Approximation

In this algorithm, periodically during learning the action-value function is approximated using a genetic algorithm. The genetic algorithm does not attempt to determine the weights to be applied to some preselected set of basis functions as is generally done, but instead chooses the *basis functions themselves* and calculates the weights using least-squares.

Basis functions are often chosen by trial and error methods that can be time consuming for the user. For methods such as LSPI, it is not even determined if the selected basis functions are acceptable until after the algorithm has run to completion. By allowing the genetic algorithm to select the basis functions and using a well understood method to calculate the weights, this trial and error can be avoided. The only major limitation is the sets of basis functions the user gives to genetic algorithm to sift through. For this paper, the basis functions are simple polynomials, sines and cosines, radial basis functions, and cubic splines.

VI. Policy Iteration

The multiresolution discretization method provides a means of learning the action-value function, $Q^\pi(s,a)$, for a fixed policy, π, in progressively finer detail. Now policy iteration is integrated into the algorithm in addition to the genetic algorithm. Since Q-learning is used in this algorithm, the policy does not need to be represented explicitly or with any sort of model. The policy can be determined using that action-value function itself with the following relationship.

$$\pi(s) = \max_a Q(s,a) \tag{8}$$

The multiresolution algorithm with the genetic algorithm already integrated can then use a similar relationship shown in

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

Eq. 9 to determine the approximate policy.

$$\hat{\pi}(s) = \max_a \hat{Q}(s,a) \tag{9}$$

Function approximation and policy iteration occurs on a regular basis at predetermined intervals. This algorithm compares the current approximate policy, $\hat{\pi}^{(t)}(s)$, and the approximate policy calculated after the last interval, $\hat{\pi}^{(t-1)}(s)$. When these converge learning at the current level of discretization is discontinued. The final set of basis functions as determined by the genetic algorithm can then be used in lieu of the table form of the action-value function.

VII. NUMERICAL EXAMPLE

This method is applied to the morphing airfoil problem first developed in Reference [7]. Figure 4 shows a representative airfoil. To learn the shape changing procedure the reinforcement learning agent initially 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 x_1-direction are defined as follows

$$\begin{aligned} A^1_{11} &\equiv {}^{(I+1)J}X - {}^{IJ}X = h^1_{x_1} \\ A^1_{12} &\equiv {}^{(I-1)J}X - {}^{IJ}X = -h^1_{x_1} \end{aligned} \tag{10}$$

Eq. 10 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.

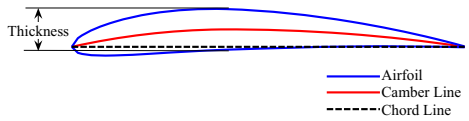

Fig. 4. Representative Airfoil

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 $h_{x_1} = 0.50\%$.

TABLE II
AXIS DEFINITIONS

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

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.

The purpose of the numerical example is to demonstrate the learning performance of the reinforcement learning agent utilizing the multiresolution state-space discretization method using the genetic algorithm for function approximation and employing policy iteration or comparison as the stopping criterion.

The agent is allowed a maximum of 5000 episodes with which to explore the state-space of thickness-camber combinations for each level of discretization. The agent is instructed to learn on 3 levels of discretization, which leads to a possible 15000 episodes total. The genetic algorithm and policy comparison are triggered every 200 episodes. The reward the agent receives is calculated according to the following equation:

$$r = |g - c_{n-1}| - |g - c_n| \tag{11}$$

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 $g := c_l = 0.3$. The area of interest is the goal, and the associated initial range is ± 0.05. The performance of the algorithm is analyzed by comparing the final value functions and policies represented by the discrete action-value function and the approximated action-value function. The policies are represented by Eqs. 8 and 9, respectively.

Fig. 5 and 6 show the visual representation of the value function and policy and approximated value function and policy, respectively. The key in Table III describes the significance of the colors for the policy representation in the figures. The genetic algorithm approximates the action-value function for each of the four actions separately. The final sets of basis functions determined by the genetic algorithm were 13^{th} degree polynomials for all four actions. Of course, the weights applied to the basis functions as determined by linear least-squares is different for each action. Both the discrete and approximated value functions in the figures show a definitive

TABLE III
POLICY COLOR KEY

Color	Direction
Blue	←
Cyan	→
Yellow	↓
Red	↑

ravine in the center of the camber axis. For this problem and reward function, this ravine indicates where the goal of $c_l = 0.30$ is located. Notice the general agreement between the discrete value function and the approximated value function. Looking closely at the policy shown in Fig. 5, it is seen that near the goal around a camber of 2.4% there are many patches in which the agent could get "stuck" and not quite reach the goal. This phenomenon is due to the agent not taking quite enough time to learn those areas. However, Fig. 6 shows that many of those patches are smoothed by the approximation, making it more likely that the agent will reach the goal. Quantitatively, there is an 80% agreement between the extracted discrete policy and the extracted approximated policy. A full set performance simulations should and will be run to determine the performance of the agent using both the discrete policy and the approximated policy.

The agent converged to the policy shown in the figures in 2200 episodes: 600 episodes at the first level of discretization, 1200 at the second, and 400 at the third, which is 85% fewer than the total 15000 episodes limit. The total computational time that elapsed while the agent learned is 6.74 hours. If the agent had been allowed to complete all 15000 episodes without any other stopping criteria, the agent would have a computational time upwards of 46 hours. This method saved almost 40 hours of computational time, and it produced an approximation of the action-value function without any additional post-processing.

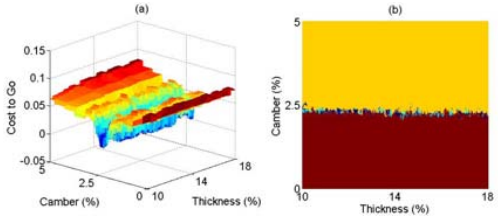

Fig. 5. Value Function (a) and Policy Representation (b)

VIII. CONCLUSIONS

A. Conclusions

The results show that the learning for this algorithm shows 80% agreement between the discrete policy and the approximated policy, showing good support for using the genetic algorithm to determine basis functions for function approximation using linear least-squares. The multi-resolution discretization portion of this method is successful in greatly reducing the time for convergence, increasing the rate of convergence, and

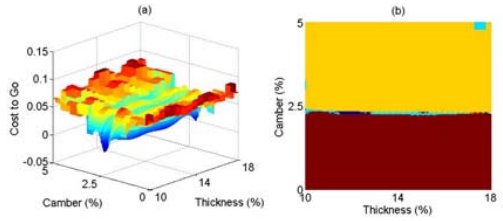

Fig. 6. Approximated Value Function (a) and Policy Representation (b)

achieving a goal with the very small range of 0.002 in fewer than the allotted number of episodes. This method reduced the total number of episodes by 85% from the total possible.

ACKNOWLEDGMENT

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 M. 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 National Science Foundation, 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] P. Singla, "Multi-resolution methods for high fidelity modeling and control allocation in large-scale dynamical systems," Ph.D. dissertation, Texas A&M University, College Station, TX, 2006.

[4] A. J. Meade, M. Kokkolaras, and B. A. Zeldin, "Sequential function approximation for the solution of differential equations," *Communications in Numerical Methods in Engineering*, vol. 13, no. 12, pp. 977–986, December 1997.

[5] M. Kokkolaras, A. J. Meade, and B. Zeldin, "Concurrent implementation of the optimal incremental approximation method for the adaptive and meshless solution of differential equations," *Optimization and Engineering*, vol. 4, no. 4, pp. 271–289, 2003.

[6] M. G. Lagoudakis and R. Parr, "Least-squares policy iteration," *Journal of Machine Learning*, vol. 4, no. 6, pp. 227–303, August 2004.

[7] 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.

[8] ——, "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.

[9] ——, "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.

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

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