# Fuzzy Q-Learning: A New Approach for Fuzzy Dynamic Programming

**Hamid R. Berenji**
Intelligent Inference Systems Corp.
Artificial Intelligence Research Branch, MS: 269-2
NASA Ames Research Center
Mountain View, CA 94035
*berenji@ptolemy.arc.nasa.gov*

***Abstract*—**
**Fuzzy Reinforcement Learning (FRL) involves "jump starting" reinforcement learning with fuzzy logic rules. By using FRL, prior domain knowledge, which may be very approximate and imprecise, can be expressed in terms of fuzzy rules and refined later through the learning process.**

**In this paper, we develop a new algorithm called Fuzzy Q-Learning (or FQ-Learning) which extends Watkin's Q-Learning method. It can be used for decision processes in which the goals and/or the constraints, but not necessarily the system under control, are fuzzy in nature. An example of a fuzzy constraint is: "the weight of object $A$ must not be *substantially* heavier than $w$" where $w$ is a specified weight. Similarly, an example of a fuzzy goal is: "the robot must be in the *vicinity* of door $k$". We show that FQ-Learning provides an alternative solution to this problem which is simpler than the Bellman-Zadeh's fuzzy dynamic programming approach. We apply the algorithm to a multistage decision making problem and a navigation task.**

## I. Introduction

In recent years, there has been an increased effort to develop intelligent systems which can perform in real-time a variety of tasks including control, planning, and decision making. This includes the development of a number of new algorithms, such as a family of Reinforcement Learning techniques, and also, re-evaluation and extension of some older techniques such as Dynamic Programming [3].

Reinforcement learning is an effective way of doing learning in domains in which there is not enough supervised training data available and the reinforcements from the environment appear only after a long sequence of actions. In recent years, incremental improvements have been made in extending the theory for control applications. In particular, Sutton's Temporal Difference method (TD algorithm) has been extended by Tesauro [12, 11] and applied in a backgammon-playing program, which performed better than many human expert backgammon players. A technique closely related to the TD algorithms is Watkin's Q-Learning method which has also been extensively studied in recent years. In particular, Lin [7] has combined Q-Learning with backpropagation and has applied it to robot control. Watkins and Dayan [13] have reported on formal convergence results of Q-learning. Priorotized Sweeping technique has been suggested by Moore and Atkeson [8] which requires learning an action model for the environment and uses all previous experiences to both guide the exploration of state-space and prioritize dynamic programming sweeps.

Non-adaptive fuzzy systems use fixed membership functions for their fuzzy labels and these functions do not change. Also, the structure of a fuzzy inference system remains unchanged. On the contrary, adaptive fuzzy systems must be able to adjust their labels definitions and possibly their structures over time. For example, the GARIC architecture [5] uses reinforcement learning and refines its membership functions by adjusting the location of the center, left and right spreads.

In this paper, we first discuss Markovian decision processes and Dynamic Programming. Next, the basics of temporal difference and Q-Learning methods are discussed, followed by a discussion on fuzzy dynamic programming. Then we present a new algorithm for Fuzzy Q-Learning and demonstrate its application to a multistage decision making problem in a fuzzy environment. Finally, we apply the algorithm to a navigation task involving obstacles.

## II. Markovian Decision Processes and Dynamic Programming

In a Markovian decision process, the transition between the current state of a system and its next state depends only on the current state and the action taken at that state. This is known as the Markov property. At each time step, the current state of the system is evaluated and an action is selected based on this evaluation. Assuming a transition function $f$, a finite set of discrete states $X$, and a finite set of discrete actions $U$, we have $x_{t+1} = f(x_t, u_t)$. If the Markov property holds in a system, then the transitions between its states can be modeled by using either a deterministic or a stochastic transition function.

Dynamic Programming (DP), introduced by Bellman [3], is a strong mathematical technique for making a sequence of interrelated decisions. It successively approximates optimal evaluation functions and the decision rules associated with them. Several authors (e.g., [1, 14]) have recently discussed its strong relation to a family of reinforcement learning methods.

## III. Reinforcement Learning

Reinforcement learning techniques assume that, during the learning process, no supervisor is present to directly judge the quality of the selected control action and instead, the final evaluation of a process is known after a long sequence of actions. The reinforcements received by the learning system can only be used to learn how to predict the outcome of the selected actions. Barto, Sutton, and Anderson [2] used two neuron-like elements to solve the learning problem in cart-pole balancing. In this approach, the state-space is partitioned into non-overlapping smaller regions and then credit assignment is performed on a local basis. However, in fuzzy reinforcement learning, the partitions of the state space can overlap leading to the use of fuzzy partitions in the antecedents and consequents of fuzzy rules. The reinforcements from the environment are then used to refine the fuzzy membership functions in the rules.

Temporal Difference (TD) [9] and Q-learning [14] methods can be used to learn how to select an appropriate continuous or discrete action, respectively. The Temporal Difference method and Q-Learning are simillar in terms of using a distal teacher for learning. In TD, two fundamental memory structures, one for evaluation and another for policy, are kept. However, Q-learning collapses these two and only maintains a structure which is a cross between an evaluation function and a policy [10].

In our previous work on the GARIC architecture [5], we have used a form of TD called Fuzzy Temporal Difference (FTD) method. The system learns to refine the labels used in its fuzzy rules through repeated trials. Based on the external reinforcements (such as failure signals) and a comparison between the goodness of the current state with the previous state, an internal reinforcement is generated which is used later to modify the action selected by the initial system. For example, if the value of the internal reinforcement is high, then the final action will be slightly different from the one recommended by the action selection network. However, if the internal reinforcement is low, then GARIC changes the selected action to a larger extent. In this paper, we show that Q-Learning can also be used for fuzzy reinforcement learning. In the following sections, we first describe Bellman-Zadeh's approach to fuzzy dynamic programming and then describe a new algorithm for Fuzzy-Q Learning.

## IV. Fuzzy Dynamic Programming

Bellman and Zadeh [4] have introduced a method for decision making in a fuzzy environment in which the goals and/or the constraints are fuzzy. In this method, a decision is defined as the fuzzy set of alternatives which results from the intersection of goals and constraints:

$$D = G \cap C \tag{1}$$

where $G$ is a fuzzy goal, $C$ is a constraint, and $D$ represents the decision. In terms of their associated membership functions, this corresponds to:

$$\mu_D = \mu_G \wedge \mu_C \tag{2}$$

where "$\wedge$" represents an "and" operator. Assuming $n$ goals $G_1, G_2, \cdots, G_n$ and $m$ constraints $C_1, C_2, \cdots, C_m$, 1 and 2 are generalized as the following:

$$D = G_1 \cap G_2 \cap \cdots \cap G_n \cap C_1 \cap C_2 \cap \cdots \cap C_m \tag{3}$$

$$\mu_D = \mu_{G_1} \wedge \mu_{G_2} \cdots \wedge \mu_{G_n} \wedge \mu_{C_1} \wedge \mu_{C_2} \cdots \wedge \mu_{C_m} \tag{4}$$

Assuming that the definition of "and" is left open, Bellman and Zadeh provide a broad definition of the concept of a decision:

Decision = Confluence of Goals and Constraints

The previous definitions can be expanded to include a multi-stage decision process. Assume a time-invariant finite-state deterministic system where the state of the system, $x_t$, ranges over a finite set $X = x_1, x_2, \cdots, x_n$ and the input (actions) ranges over a finite set $U = u_1, u_2, \cdots, u_n$. Then the next state of the system is given by

$$x_{t+1} = f(x_t, u_t) \tag{5}$$

in which $f$ is a given function for generating the successor of $x_t$ given the input (action) $u_t$. For simplicity, we assume $f$ is deterministic although a stochastic system can also be modeled using a probability distribution $P(x_{t+1}|x_t, u_t)$.

### *A. A Multi-stage Decision Process*

In this section we consider a fuzzy dynamic programming problem (first discussed by Bellman-Zadeh [4]) where the goal is a subset of states $X$ called the *termination set*, $T$. The process terminates whenever a state belonging to the termination set is reached. The state transition occurs as in (5) and the state of the system ranges over $X = \{x_1, \cdots, x_l, x_{l+1}, \cdots, x_n\}$ where $T = \{x_{l+1}, \cdots, x_n\}$ is the termination set. These states are interpreted as absorbing states and therefore will have $f(x_i, u_j) = x_i$ for all $u_j$ in $U$. The fuzzy goal is a subset of $T$ characterized by a membership function $\mu_G(x_N)$ where $N$ is the time that the process enters one of the goal states.

As before, for each initial state, a decision is defined as

$$D(x_0) = C(x_0) \cap C(x_1) \cap \cdots \cap C(x_{N-1}) \cap G \quad (6)$$

The above equation is then generalized to

$$D(x_t) = C(x_t) \cap C(x_{t+1}) \cap \cdots \cap C(x_{t+N-1}) \cap G \quad (7)$$

From this, Bellman and Zadeh show that

$$D(x_t) = C(x_t) \cap D(f(x_t, u_t)) \quad t = 0, 1, 2, \cdots \quad (8)$$

with the following membership functions of the sets in question

$$\mu_D(u_0, .., u_{N-1}|x_0) = \mu_C(u_0|x_0) \wedge \mu_D(u_1, .., u_{N-1}|f(x_0, u_0)) \quad (9)$$

Further details for this approach can be found in [4] where an approach is developed for first transforming the equations for policy selection to their compact matrix form and then solving the resulting family of functional equations. In the next section, we present a much simpler approach based on Fuzzy Q-Learning to solve the above problem.

## V. Fuzzy Q-Learning

Fuzzy Q-Learning extends Watkin's Q-learning method for decision processes in which the goals and/or the constraints, but not necessarily the system under control, are fuzzy in nature. An example of a fuzzy constraint is: "the weight of object $A$ must not be ***substantially*** heavier than $w$" where $w$ is a specified weight. Similarly, an example of a fuzzy goal is: "the robot must be in the ***vicinity*** of door $k$".

The Q-learning algorithm maintains an estimate $Q(x, a)$ of the value of taking action $a$ in state $x$. Similarly, in the FQ-Learning algorithm, we maintain an estimate $FQ(x, a)$ for taking action $a$ in state $x$, where the actions can have fuzzy constraints on them, and continuing with the optimal policy after a new state is reached. The value of a state can be defined as the value of the state's best state-action pair:

```
Initialize FQ values
Until FQ values converge or max. # of trials do {
  1. $x \leftarrow$ current state
  2. Select the action with the highest FQ. If multiple
     exist, select randomly among them.
  3. Apply action, observe the new state ($y$) and reward
     ($r$)
  4. Update $FQ(x,a) \leftarrow FQ(x,a) + \beta[(r + \gamma V(y)) \wedge$
     $C(x,a) - FQ(x,a)]$
     where $\wedge$ represents a conjunction or an "and" oper-
     ator (e.g., the "minimum") }
```

Figure 1: The FQ-Learning Algorithm

$$V(x) = Max_a \, FQ(x, a) \quad (10)$$

We define the FQ value as the expected value of the *confluence* of the immediate reinforcements plus the discounted value of the next state ***and*** the constraints on performing action $a$ in state $x$

$$FQ(x, a) = E\{(r + \gamma V(y)) \wedge \mu_C(x, a)\} \quad (11)$$

where $\gamma$ is a discount factor. The FQ values are updated according to:

$$FQ(x, a) \leftarrow FQ(x, a) + \beta[(r + \gamma V(y)) \wedge \mu_C(x, a) - FQ(x, a)] \quad (12)$$

where $\wedge$ represents a conjunction or an "and" operator (e.g., the "minimum") and $\beta$ is a learning rate.

The FQ-Learning method proceeds in the following manner. A state is selected and the FQ values for all the possible actions from that state are compared. The action with the maximum FQ value is selected and gets applied. As the result of applying this action, the system moves to a new state and an immediate reinforcement (if any) is received. The process continues until the FQ values converge to their optimal values or a maximum number of trials is exceeded. Figure 1 depicts this process.

### *A. Examples*

In this section, we apply FQ-Learning to a multistage decision making problem and a simple navigation task.

#### *A.1. A Multistage Decision Making Problem*

We use a modified version of a problem first discussed by Bellman and Zadeh [4] to provide insights into the above

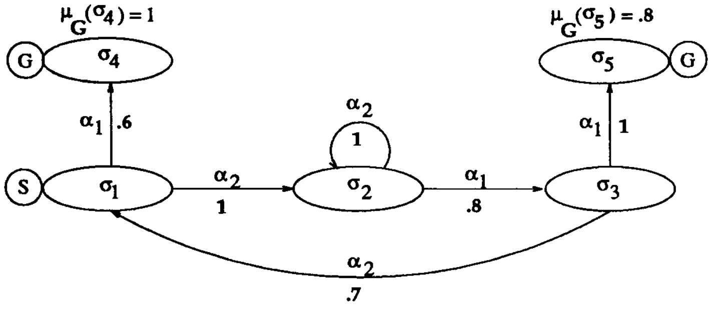

Figure 2: The starting position, all FQ values are set to their initial values (e.g., 0).

algorithm. As shown in Figure 2, there are five states in the system where $\sigma_4$ and $\sigma_5$ represent the goals. Associated with each goal is a fuzzy membership value which represents its desirability. The problem is to find an optimal policy for taking the best action at each state to reach the goals. In [4] these states were modeled as absorbing states. For our purpose in demonstrating FQ-Learning, we assume that whenever these states are reached, then the state of the system immediately transfers to the starting state which can either be fixed or randomly selected. In each state, two actions (i.e., $\alpha_1$ and $\alpha_2$) are possible. A fuzzy constraint on each state-action pair takes value in $[0, 1]$ where the lower the value, the harder is to take the action. These are also shown in the figure for each state-action pair.

Figure 3 demonstrates the results of applying the FQ-Learning algorithm to this problem. For state 2 and 3, FQ values are higher for selecting the correct action from early in the learning process. However, for state 1, until approximately the first 50 trials, higher FQ values are observed for selection of action $\alpha_1$. As shown in Figure 3(a), after approximately the first 50 trials, FQ values start to represent the correct action for that state. The final policy found in this example is:

| State | Policy |
|---|---|
| 1 | $\alpha_2$ |
| 2 | $\alpha_1$ |
| 3 | $\alpha_1$. |

### *A.2. A Navigation Task*

In this section, we demonstrate the performance of the FQ-Learning algorithm in a navigation task under the presence of obstacles. Consider the maze shown in figure 4 first discussed by Sutton [10]. The shaded squares are states which cannot be entered and represent obstacles. The starting state is marked "S" and the goal states are marked by "G". Here, we extend Sutton's example to allow for multiple goal states with different reinforcements and possible actions to have local constraints as well as global constraints. In this demonstration, reward is zero for all transitions except for those into one of the goal states for which it will take the reinforcement value associated with the goal. From each state, actions UP, DOWN, RIGHT, and LEFT are possible, and each action may have a constraint. For example, in our experiments, action "UP" had a .85 constraint on it. Actions were selected stochastically according to a Boltzmann distribution: $P(a|x) = e^{FQ(x,a)} / \sum_j e^{FQ(x,j)}$ where $j$ represents the total number of actions available at state $x$. Figure 4 shows the FQ values (a) during learning and (b) after their convergence.

## VI. Conclusion

The Fuzzy Temporal Difference (FTD) method as developed in GARIC keeps two separate action-state evaluation and action selection memory structures while FQ-Learning algorithm combines these two into a single structure. By keeping these memory structures separate, FTD can offer the following advantages over FQ-Learning:

1. Prior knowledge about action selection can be encoded simpler, for example, in terms of fuzzy logic control rules.
2. If prior knowledge also exists for state evaluation, then that can also be represented by fuzzy rules for guiding reinforcement learning.
3. FTD is more suitable for continuous domains such as control applications.

On the other hand, by collapsing the state-action evaluation and policy selection memory structures into one structure, FQ-Learning's search process is simpler and the algorithm is more appropriate for planning problems such as path planning with obstacles, multi-stage decision making, and solving dynamic programming problems.

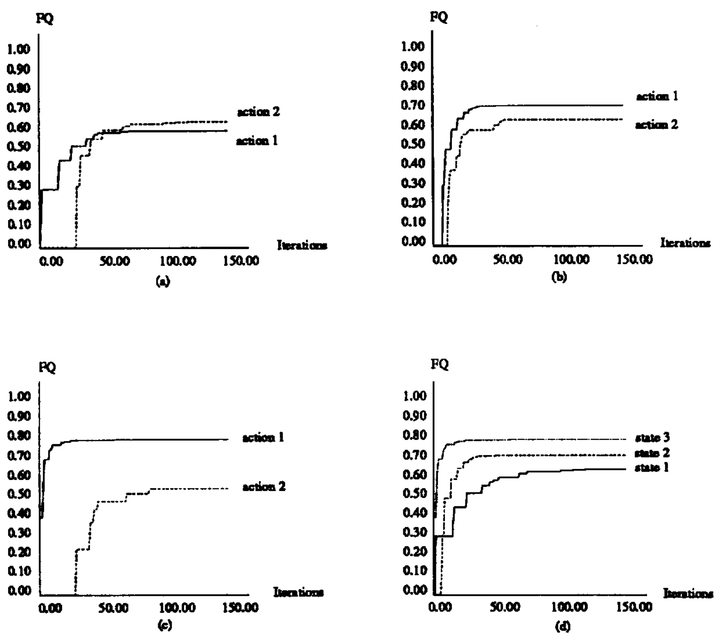

Figure 3: The convergence of FQ values for (a)- state 1, (b)- state 2, (c)- state 3, and (d)- for all three states (only the action with the higher FQ value is shown). After achieving a goal, the starting state is set randomly to a non-goal state.

The Fuzzy Temporal Difference method provides an excellent potential for use in *intelligent control* systems. It has been applied to challenging control problems such as the simulation of the Space Shuttle orbital operations [6]. FQ-Learning extends the applications of Q-Learning to fuzzy environments where fuzzy goals and fuzzy constraints can be modeled. We showed the application of this method to a typical multi-stage decision making problem and a simple navigation task. However, FQ-Learning suffers to some extent from the same type of difficulties as Q-Learning (e.g., slow learning rate) and its use in control problems has not yet been explored. Another interesting direction to be pursued later is an investigation of FQ-Learning's full potential in decision making and reasoning under uncertainty.

**Acknowledgements** I am thankful to Stan Malyshev for providing computer programming assistance for the experiments in FQ-Learning. Also, sincere thanks to John Bresina and Keith Swanson for many excellent discussions and useful suggestions related to Q-Learning. I am grateful to Silvano Colombano, Pandu Nayak and Stan Malyshev for reviewing an earlier draft of this paper.

## References

[1] A. G. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Submitted to AI Journal special issue on Computational Theories of Interaction and Agency*, 1993.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:834–846, 1983.

[3] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[4] R.E. Bellman and L.A. Zadeh. Decision-making in a fuzzy environment. *Management Science*, 17(4):B–141:B–164, 1970.

[5] H.R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks*, 3(5), 1992.

[6] H.R. Berenji, Y. Jani R.N Lea, P. Khedkar, A. Malkani, and Jeffrey Hoblit. Space shuttle attitude control by fuzzy logic and reinforcement learning. In

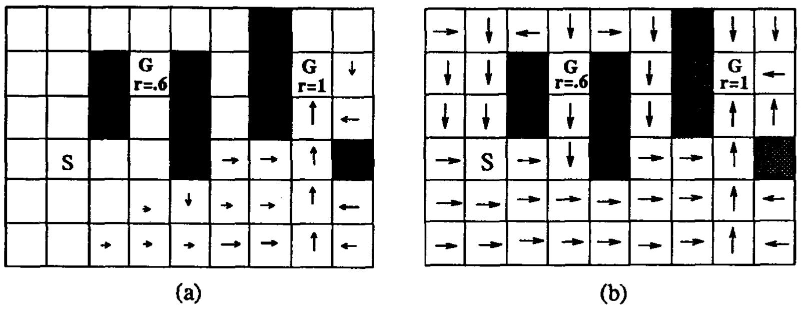

Figure 4: A simple navigation task. A trial is one trip from start state "S" to one of the goal states "G". An arrow represents the direction that the highest *FQ* value for a state recommends and its length corresponds to the *FQ* magnitude (a)- Early in the learning process after 1000 steps: many states lack strong *FQ* values (b)- After 100,000 steps and the convergence of the FQ values.

*Second IEEE International conference on Fuzzy Systems*, San Francisco, CA, March 1993.

[7] L.J. Lin. Programming robots using reinforcement learning and teaching. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 1991.

[8] A. Moore and C. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning*, page to appear.

[9] R.S. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[10] R.S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, 1990.

[11] G. Tesauro. Practical issues in temporal difference learning. *Machine Learning*, (8):257–277, 1992.

[12] G. Tesauro. Td-gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation*, 6(2):215–219, 1994.

[13] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, (8):279–292, 1992.

[14] C.J.C.H. Watkins. *Learning with Delayed Rewards.* PhD thesis, Cambridge University, Psychology Department, 1989.