AIAA Guidance, Navigation, and Control Conference and Exhibit
11-14 August 2003, Austin, Texas

AIAA 2003-5718

AIAA-83-GNC-49

Minimax Reinforcement Learning

Suman Chakravorty* David C. Hyland†
Department of Aerospace Engineering
University of Michigan, Ann Arbor

Abstract

In this paper, the minimax actor-critic algorithm is presented. This is the minimax equivalent of the actor-critic algorithm in the case of probabilistic dynamic programming. The convergence of the policies generated by the algorithm, to an optimal policy, is established. The algorithm is applied to an example involving a UAV navigating hostile territory. Further, error bounds are obtained for approximations involved in solving large scale minimax DP problems, specifically the case of state aggregation.

1 Introduction

Dynamic Programming (DP) provides a formal framework for sequential decision making under uncertainty [3, 4, 8, 14]. In the standard DP formulation, the state x of a discrete-time system evolves according to transition probabilities dependent on a decision/control u. The system incurs an incremental cost, $c(x,u)$, in taking a decision u at state x. The cost incurred by the system in following a policy μ , a sequence of control actions, $\{\mu_0, \mu_1, ...\}$, is the expected value of the weighted sum of the incremental cost incurred at every stage, i.e.,

$$P_\mu(x) = E(\sum_{k=0}^{\infty} \beta^k c(x_k, \mu_k)/x_0 = x). \quad (1.1)$$

If $\beta = 1$, the problem is termed a stochastic shortest path problem [4]. Problems in which $\beta < 1$ are termed as discounted problems. In such problems future costs are considered less important than immediate costs. In this paper we shall be considering discounted problems. The central construct of the DP framework is the optimal "cost-to-go/reward/utility" function which is defined to be the infimum of the cost incurred in following a policy over the space of all possible policies, i.e.,

$$P^*(x) = \inf_\mu P_\mu(x). \quad (1.2)$$

It can be further shown that the optimal cost-to-go function is a fixed point of the DP operator B,[4, 7], where

$$BP(x) = \inf_{u \in A} \{c(x,u) + \beta E(P(y)/(x,u))\}, \quad (1.3)$$

y is the state at the next instant, given that control u is taken at state x, at the current instant. Two algorithms find widespread use in the numerical evaluation of the optimal cost-to-go function: the value iteration[3, 4] and the policy iteration[4, 11] algorithms. However, the policy / value iteration algorithms require that a model of the system be available, (i.e., the transition probabilities governing the dynamics of the system be known). In the absence of a model of the system, the cost-to-go function can be evaluated using simulation based methods. These methods "learn" the optimal cost-to-go function through repeated simulations and are known as Reinforcement Learning methods in literature. The reinforcement learning method based on value iteration is termed Q-learning [4, 18, 22] and that based on policy iteration is known as an actor-critic system [2, 4].

In this paper, we present the worst case variant of the standard DP problem. To motivate the use of a worst case formulation, consider the system shown in figure1. There are 3 states in the system: 1, 2 and G. The goal is to get to the state G from state 1. There are two control options at state 1, move fast or move slow. The fast move is shown by the dashed line. If the system moves fast from 1, it reaches the goal state G with a cost of 10. However, if it moves slow, it can reach the goal state with a cost of 1 with probability equal to 0.9. However, there is a probability of 0.1 that it reaches state 2 with cost of 1. Once in state 2, the system gets stuck and stays there forever incurring a cost of 1 at every instant. At the goal state the system remains forever

*Graduate Student Research Assistant, email: schakrav@engin.umich.edu
†Professor and Chairman, email: dhiland@engin.umich.edu

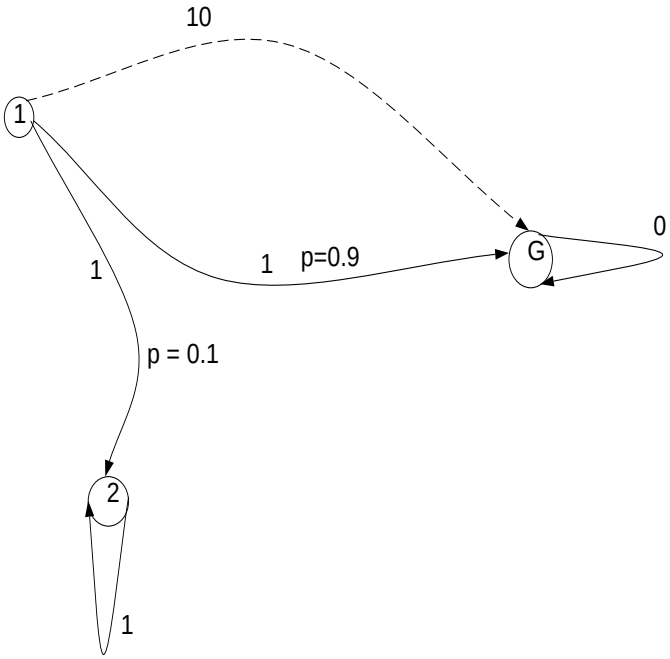

Figure 1: Motivating example for using minimax DP

without incurring any cost. According to tbe probabilistic formulation, the optimal action at state 1 is to move slow. However, note that this involves the risk of the system never reaching the goal state and getting stuck in state 2. According to the worst case formulation, the optimal action at state 1 is to move fast. This action involves no risk of getting stuck at state 2. Thus, the worst case formulation is a better way of solving the decision problem if we want to avoid risk. Under the minimax DP formulation, the dynamics of the system is defined by the next state sets, $\Gamma(x,u)$, which consist of all the states that can result from taking decision u at state x. We define the "worst case" cost-to-go incurred in following a policy μ as

$$J_\mu(x) = \sup_{\gamma_0,\gamma_1,..} \{\beta^k c(\gamma_k, \mu_k)\}, \qquad (1.4)$$

where $\gamma_0 = x$ and $\gamma_k \in \Gamma(\gamma_{k-1}, \mu_{k-1})$, i.e., the worst case cost that can result in following policy μ.
We define the optimal cost-to-go function J^*, in the same way as that for the standard DP problem. It can be shown that the optimal cost-to-go function is a fixed point of the worst case DP operator T, where

$$TJ(x) = \inf_{u\in A}\{c(x,u) + \beta \sup_{y\in\Gamma(x,u)} J(y)\}. \qquad (1.5)$$

Also, the worst case optimal-cost-to-go function can be evaluated using the value iteration and the policy iteration algorithms. The worst case approach to sequential decision making problems is also well-known in literature and are termed as risk-sensitive Markov Decision Processes(MDP) [5, 6, 20]. A notable feature of this formulation is the absence of a Markovian assumption on the underlying system dynamics. Therefore, this treatment remains valid for the case of a system in which the states can only be partially observed. The Reinforcement Learning methods based on minimax DP are termed Minimax reinforcement learning algorithms. It was introduced by Heger[9, 10]. Heger presented the minimax equivalent of the Q-lerning algorithm and termed it $\hat{Q}$-learning. More work on minimax reinforcement learning can be found in [15, 12]. [12] discusses the approximate solution of the minimax DP algorithm using state aggregation. It shows that the algorithm converges , however no error bounds on the approximation are obtained. Our main contribution in this paper is to introduce a minimax Actor-Critic algorithm and show its convergence. We also discuss the problem of function approximation in solving large scale DP problems. We consider the simplest form of function approximation , state aggregation, and provide bounds on the error that result from such an approximation. We would like to note that such results are known in the case of probabilistic formulation of DP [19]. This paper is orgainized as follows. Section 2 presents the minimax DP operator and establishes its basic properties and shows two methods of solution of the fixed point equation: value iteration and policy iteration. Section 3 presents the minimax actor-critic algorithm and shows its convergence. Section 4 presents a numerical example involving a UAV navigating hostile territory. Section 5 discusses the use of state aggregation in order to solve large scale DP problems and obtains error bounds on such approximations.

2 The Worst case / Minimax Dynamic Programming Operator

In this section, we present the worst case DP operator and mention its properties. Throughout this paper we shall be using the terms minimax / worst case DP interchangeably.
Let S denote the state space of the system. Let A denote the control/action space. The **incremental cost** incurred by the system in taking control u_k at state x_k is given by the non negative and bounded function $c(x_k, u_k)$. We denote by $c(x,u,y)$ the maximal incremental cost that can be incurred by the

system in making a transition from state x to state y under control action u.

Definition 2.1 *We denote the next state set of some x under control action u by $\Gamma(x,u)$ and define it as:*

$$\Gamma(x,u) = (y \in S | x \rightarrow^u y). \tag{2.1}$$

Please note that we shall be using the term "state" loosely in this paper and stress that the framework developed here is perfectly valid for incomplete state measurements too. Also note that there is no Markovian assumption on the dynamics of the system.

Definition 2.2 ***A control policy*** *is defined as a sequence of control actions i.e. a policy $\mu = (\mu(0), \mu(1), ..)$ where $\mu(k) \in A$ and is the control action chosen by the system at instant k.*

Consider any $x \in S$ and any policy μ. Note now that if policy μ were applied to the system starting at x, due to the uncertainity in the system, there is more than one path from x corresponding to μ. Let these paths be denoted by $\gamma = (\gamma_0, \gamma_1, ..)$, such that

$$\gamma_k \in \Gamma(\gamma_{k-1}, \mu_{k-1}), \tag{2.2}$$

$$\gamma_0 = x. \tag{2.3}$$

Definition 2.3 *The cost to go w.r.t path γ for policy μ from x is defined as*

$$J_\mu^\gamma(x) = \sum_{k=0}^{\infty} \beta^k c(\gamma_k, \mu_k). \tag{2.4}$$

Definition 2.4 *The cost to go from state x w.r.t policy μ as*

$$J_\mu(x) = \sup_\gamma J_\mu^\gamma(x). \tag{2.5}$$

Definition 2.5 *The optimal cost to go from state x is defined as*

$$J^*(x) = \inf_\mu J_\mu(x). \tag{2.6}$$

Let $B(S, \Re)$ denote the space of bounded real functionals with topology defined according to the sup norm, i.e., $||J||_\infty = \sup_{x \in S} |J(x)|$. Note that J_μ, J^* belong to $B(S, \Re)$, by definition.
We define the worst case DP operator T:

$$TJ(x) = \inf_{u \in A} (\sup_{y \in \Gamma(x,u)} (c(x,u,y) + \beta J(y))), \quad \forall x \in S. \tag{2.7}$$

The next proposition states that the minimax DP operator is a contraction mapping in the space of bounded real functionals under the supremum norm. The proofs of the propositions are straightforward extensions of the results in probabilistic DP [4].

Proposition 2.1 *The worst case DP operator T is a contraction mapping in $B(S, \Re)$ w.r.t the sup norm i.e*

$$||TJ - TJ^{'}||_\infty \leq \beta ||J - J^{'}||_\infty$$

Next, it can be shown that the minimax optimal cost function is the fixed point of the minimax DP operator.

Proposition 2.2 *The optimal cost to go from any state $x \in S$ satisfies*

$$TJ^*(x) = J^*(x). \tag{2.8}$$

By Propositon 2.1 and the contraction mapping theorem [13], there exists a unique fixed point of T in $B(S, \Re)$. Hence, it follows from Proposition 2.2 that the fixed point is J^*, the optimal cost to go.

Definition 2.6 *A stationary policy is a control policy under which the control action taken at a particular state is the same regardless of the instant it is taken.*

The next proposition states that the minimax DP operator, defined w.r.t a stationary policy, is also a contraction mapping and that cost function w.r.t the policy is its unique fixed point.

Proposition 2.3 *The operator T_μ defined by*

$$T_\mu J(x) = \sup_{y \in \Gamma(x, \mu(x))} (c(x, \mu(x)) + \beta J(y)), \forall\, x \in S, \tag{2.9}$$

where μ is any stationary policy, is a contraction operator and J_μ is its unique fixed point.

In the rest of this section we mention two methods for evaluating the otimal cost-to-go function, namely, the value iteration and the policy iteration algorithms. The presentation of the policy iteration algorithm follows in the lines of [7].

Proposition 2.4 Value Iteration : *Given any $J_0 \in B(S, \Re)$, the sequence $\{T^n J_0\}$ converges to the optimal cost-to-go function, J^*, under the sup norm.*

The **Policy Iteration** algorithm is represented as:
1) Choose some stationary policy μ_0.
2) Evaluate J_{μ_k}.
3) Find μ_{k+1} such that $TJ_{\mu_k} = T_{\mu_{k+1}}J_{\mu_k}$
4) If $||J_{\mu_k} - TJ_{\mu_k}|| > \epsilon(1-\beta)$, replace μ_k by μ_{k+1} and go to step 2, o.w. stop.

Definition 2.7 *An ϵ-optimal policy is defined to be a policy such that*

$$||J_\mu - J^*||_\infty \leq \epsilon \tag{2.10}$$

where J^* represents the optimal cost-to-go and J_μ represents the cost-to-go with respect to the policy μ.

With the above definitions, we have the following result.

Proposition 2.5 *Given any $\epsilon > 0$, the policy iteration algorithm terminates in an ϵ-optimal policy in a finite number of iterations.*

3 Minimax Adaptive Critics

In the previous section, we introduced the worst case DP operator and enumerated a few of its properties. We also considered two methods for finding the optimal cost-to-go function, namely, value and policy iterations. However, both these methods require that the system model be supplied to them. Thus, when the system model is unknown or partially known, these methods break down. In probabilistic reinforcement learning, this impasse is dealt with through the use of Q-learning and Adaptive critics. These are model-free learning techniques which require a simulation model of the system or can be implemented on-line. The Q-learning scheme is based on the value iteration algorithm while the adaptive critic scheme is based on the policy iteration algorithm. In this section, we present the minimax adaptive critic algorithm and prove its convergence. The minimax equivalent of the Q-learning scheme was introduced by [9, 10] and called $\hat{Q}$ learning. In this section, we assume that the state space and control space of the system is finite. Consider any fixed stationary policy μ. Let the the state space be denoted by $S = \{1, 2, ..i, ..N\}$. Recall that $c(i, u, j)$ represents the maximal incremental cost that a system can incur in transitioning from $i \to j$, under the control action u.

$\mathcal{A}$ **3.1** *The set of incremental costs that can be incurred by a system in making a transition $i \to j$ under control u is finite. The system has a non-zero probability of incurring any of these costs.*

$\mathcal{A}$ **3.2** *The simulation model is run under policy μ. It is assumed that every state $i \in S$ is visited infinitely often during the course of the simulation run. There is a non-zero probability for the system to transition from any $i \to j$, where $j \in \Gamma(i, \mu(i))$.*

Simulation Based Policy Evaluation:
1) Set $J_\mu^0 = 0$
2) At instant k , at state i_k , take control action $\mu(i_k)$ and simulate transition to i_{k+1}.
3) if $c_k(i_k, \mu(i_k)) + \beta J_\mu^k(i_{k+1}) > J_\mu^k(i_k)$ then $J_\mu^{k+1}(i_k) = c_k(i_k, \mu(i_k)) + \beta J_\mu^k(i_{k+1})$ else $J_\mu^{k+1}(i_k) = J_\mu^k(i_k)$
Under $\mathcal{A}$3.1 and $\mathcal{A}$3.2 we have that,

Proposition 3.1

$$J_\mu^k(i) \to J_\mu(i) \, \forall i \in S \tag{3.1}$$

Proof: If we have a bounded incremental cost function, it can be seen that the sequence $\{J_\mu^k(i)\}$ is a monotonically non-decreasing sequence and bounded above $\forall i \in S$. Hence, it follows that $J_\mu^k(i) \to J_\mu^\infty(i) \, \forall i$, where $J_\mu^\infty(i)$ is some real number. Consider some $i \in S$. Let Γ^i denote the set of times at which the state i is visited. Define,

$$G_\mu^\infty(i) = \max_{j \in \Gamma(i,\mu(i))} \{c(i, \mu(i), j) + \beta J_\mu^\infty(j)\}, \tag{3.2}$$

$$G_\mu^t(i) = c_t(i, \mu(i)) + \beta J_\mu^t(j). \tag{3.3}$$

. We have,

$$\begin{aligned} G_\mu^t(i) &\leq c(i, \mu(i), j) + \beta J_\mu^t(j) m \\ &\leq \max_{j \in \Gamma(i,\mu(i))} \{c(i, \mu(i), j) + \beta J_\mu^t(j)\}, \\ \leq \max_{j \in \Gamma(i,\mu(i))} &\{c(i, \mu(i), j) + \beta J_\mu^\infty(j)\} = G_\mu^\infty(i). \end{aligned} \tag{3.4}$$

The last inequality follows from the monotonicity of T_μ and noting that J_μ^∞ is the supremum of the sequence J_μ^t. Hence,

$$G_\mu^t(i) \leq G_\mu^\infty(i) \; \forall t. \tag{3.5}$$

Note that by definition, $\{J_\mu^t(i)\} \subseteq \{G_\mu^t(i)\}$, where, $t \in \Gamma^i$, which implies

$$\sup\{J_\mu^t(i)\} \leq \sup\{G_\mu^t(i)\}. \tag{3.6}$$

Thus, it follows that

$$J_\mu^\infty(i) \leq G_\mu^\infty(i) \tag{3.7}$$

. Consider some $t \in \Gamma^i$. Let

$$j^* = arg \max_{j \in \Gamma(i,\mu(i)} \{c(i, \mu(i), j) + \beta J_\mu^t(j)\}. \tag{3.8}$$

Then, it follows under assumptions3.1 and 3.2, that the transformation $i \rightarrow j^*$ takes place with the system incurring the maximal incremental cost $c(i,u,j^*)$ for some $\bar{t} > t$. Then,

$$\begin{aligned} J_\mu^\infty(i) &\geq J_\mu^{\bar{t}+1}(i) \geq G_\mu^{\bar{t}}(i) \\ &= c(i,\mu(i),j^*) + \beta J_\mu^{\bar{t}}(j^*) \\ &\geq \max_{j\in\Gamma(i,\mu(i))} \{c(i,\mu(i),j) + \beta J_\mu^t(j)\} \end{aligned} \tag{3.9}$$

Since $J_\mu^\infty(i) \leq G_\mu^\infty(i)$ from eq3.7, we have,

$$\begin{aligned} &|J_\mu^\infty(i) - G_\mu^\infty(i)| \\ \leq\ &| \max_{j\in\Gamma(i,\mu(i))} \{c(i,\mu(i),j) + \beta J_\mu^\infty(j)\} - \\ &\max_{j\in\Gamma(i,\mu(i)} \{c(i,\mu(i),j) + \beta J_\mu^t(j)\}| \\ &\leq \beta \max_{j\in\Gamma(i,\mu(i))} |J_\mu^\infty(j) - J_\mu^t(j)| \end{aligned} \tag{3.10}$$

Given any $\delta > 0$ there exists $t_\delta < \infty$ s.t $||J_\mu^\infty - J_\mu^{t_\delta}|| < \delta$, therefore

$$\begin{aligned} |J_\mu^\infty(i) - G_\mu^\infty(i)| &\leq \beta \max_{j\in\Gamma(i,\mu(i))} |J_\mu^\infty(j) - J_\mu^{t_\delta}(j)| \\ &\leq \beta||J_\mu^\infty - J_\mu^{t_\delta}|| = \beta\delta \end{aligned} \tag{3.11}$$

which implies that,

$$J_\mu^\infty(i) \geq G_\mu^\infty(i) - \beta\delta. \tag{3.12}$$

Noting that δ can be made arbitrarily small, it follows from eqs 3.7 and 3.12 that $J_\mu^\infty(i) = G_\mu^\infty(i)$. Also this holds for all $i \in S$. Hence, J_μ^∞ is the unique fixed point of T_μ from the contraction mapping theorem [13].

q.e.d

Note that in the special case of finite state space and control space, the number of stationary policies possible are finite. Hence it follows from the policy iteration algorithm(proposition 2.5) that there exists an optimal policy and that the policy iteration algorithm converges to this optimal policy. This shows that our algorithm for policy evaluation or in other words the CRITIC, given a stationary policy, finds the cost-to-go w.r.t the policy. However, in policy iteration, given μ_k and J_{μ_k}, we need to make a policy update μ_{k+1} such that

$$\mu_{k+1}(i) = arg\min_{u\in A}\{\max_{j\in\Gamma(i,u)} c(i,u,j) + \beta J_\mu(j)\}, \tag{3.13}$$

i.e., we need an ACTOR that can provide us with an improved policy given the cost-to-go function associated with the current policy. We propose a modification of the simulation based policy evaluation that allows us to carry out the policy update in the simulation framework, i.e, when we do not have a model of the system. We make following assumption and definition.

$\mathcal{A}$ **3.3** *During policy evaluation we assume that every control action $u \in A$ other than the policy is chosen randomly and infinitely often at every state $i \in S$.*

Now we define a γ-factor w.r.t any policy μ as

Definition 3.1 $\gamma_\mu(i,u) = \max_{j\in\Gamma(i,u)}\{c(i,u,j) + \beta J_\mu(j)\}$

Please note that these γ-factors are the worst case equivalents of the notion of Q-factors in traditional DP.($\hat{Q}$ factors in [9, 10]). Consider the following algorithm,

1) Initialise $\gamma_\mu^0(i,u) = 0$, i.e. initialise the gamma factors for all control-state pairs to 0.

2) suppose at instant k, control action u is taken at state i , we update the gamma-factor according to the following rule,

if $c_k(i,u) + \beta J_\mu^k(j) > \gamma_\mu^k(i,u)$

then $\gamma_\mu^{k+1}(i,u) = c_k(i,u) + \beta J_\mu^k(j)$

else $\gamma_\mu^{k+1}(i,u) = \gamma_\mu^k(i,u)$ Note that $J_\mu(i) = \gamma_\mu(i,\mu(i))$.

We term any update of $\gamma(i,u)$ where $u \neq \mu(i)$ as an ACTOR update and an update of $\gamma(i,\mu(i))$ as a CRITIC update.

Proposition 3.2 *Under the algorithm above,*

$$\gamma_\mu^k(i,u) \rightarrow \gamma_\mu(i,u)\, \forall\, i \in S,\, u \in A \tag{3.14}$$

Proof: Consider some $i \in S$,$u \in A$.

Then by definition,$\gamma_\mu^{t+1}(i,u) \geq c_t(i,u) + \beta J_\mu^t(j)$.

We can see that $\gamma_\mu^t(i,u)$ is monotonically non-decreasing and bounded. Therefore it converges to a limit, say $\gamma_\mu^\infty(i,u)$.

Define $G_\mu^t(i,u) = c_t(i,u) + \beta J_\mu^t(j)$

By definition, $\gamma_\mu^\infty(i,u) = sup\{\gamma_\mu^t(i,u)\}$. Also note that by definition $\{\gamma_\mu^t(i,u)\} \subseteq \{G_\mu^t(i,u)\}$.

Therefore, given any $\epsilon > 0$, there exist $t_1, t_2 < \infty$ s.t.

$$\begin{aligned} &\gamma_\mu^\infty(i,u).. \\ &\leq \gamma_\mu^{t_1}(i,u) + \epsilon \\ &= G_\mu^{t_2}(i,u) + \epsilon \leq c(i,u,j(t_2)) + .. \\ &\beta J_\mu^{t_2}(j(t_2)) + \epsilon \\ &\leq c(i,u,j(t_2)) + \beta J_\mu(j(t_2)) + \epsilon \\ &\leq \max_{j\in\Gamma(i,\mu(i))}\{c(i,u,j) + \beta J_\mu(j)\} + \epsilon \\ &= \gamma_\mu(i,u) + \epsilon \end{aligned} \tag{3.15}$$

By $j(t_2)$ we denote the state that the system transitions to from state i at instant t_2. Since ϵ can be arbitrarily small, it follows that

$$\gamma_\mu^\infty(i,u) \leq \gamma_\mu(i,u) \tag{3.16}$$

Let $j^* = arg\max_{j\in\Gamma(i,u)}\{c(i,u,j) + \beta J_\mu^t(j)\}$.
Let Γ^{iu} denote the set of times at which control u is taken at state i. Then given any $t \in \Gamma^{iu}$ there exists $\bar{t} > t$ s.t. the transformation $i \to j^*$ takes place with the system incurring the maximal incremental cost $c(i,u,j^*)$. It follows that

$$\begin{aligned}\gamma_\mu^{\bar{t}+1}(i) &\geq c(i,u,j^*) + \beta J_\mu^{\bar{t}}(j^*) \\ &\geq \max_{j\in\Gamma(i,u)}\{c(i,u,j) + \beta J_\mu^t(j)\}\end{aligned} \tag{3.17}$$

Since $\gamma_\mu^{\bar{t}}(i)$ is monotonically non decreasing and from eq3.16 we have that

$$\begin{aligned}&|\gamma_\mu^\infty(i) - \gamma_\mu(i,u)| \\ &\leq |\max_{j\in\Gamma(i,u)}\{c(i,u,j) + \beta J_\mu^t(j)\} \\ &-max_{j\in\Gamma(i,u)}\{c(i,u,j) + \beta J_\mu(j)\}| \\ &\leq \beta \max_{j\in\Gamma(i,u)}|J_\mu^t(j) - J_\mu(j)|\end{aligned} \tag{3.18}$$

Note that, given any $\delta > 0$, there exists $t_\delta < \infty$ s.t. $||J_\mu^{t_\delta} - J_\mu|| < \delta$, we have

$$\gamma_\mu^\infty(i,u) \geq \gamma_\mu(i,u) + \beta\delta \tag{3.19}$$

However δ can be made arbitrarily small, thus it follows from eqs.3.16 and 3.19 that $\gamma_\mu^\infty(i,u) = \gamma_\mu(i,u)$. Also, note that it holds for all control, state pairs. Thus, we have our result.
q.e.d
Its readily seen that the policy update stage is now reduced to solving the equation

$$\mu_{k+1}(i) = arg\min_u \gamma_{\mu_k}(i,u) \tag{3.20}$$

This allows to propose the following ACTOR-CRITIC algorithm:
1) choose some initial policy μ_0
2) evaluate γ_μ.
3) choose a new policy according to eq3.20.
4) If $||J_{\mu_k} - TJ_{\mu_k}|| > \epsilon(1-\beta)$, replace μ_k by μ_{k+1} and go to step 2 o.w. stop.
Note that in step4, $TJ_{\mu_k}(i) = arg\min_u \gamma_\mu(i,u)$.
Also in the above algorithm we may set $\epsilon = 0$. We know from proposition 3.1 and proposition3.2 that using algorithm3 , the cost-to-go function with respect to any policy can be evaluated using simulation and that the policy update can be carried out conveniently using the γ-factors generated by the algorithm. Also due to proposition2.5 we have that algorithm1 terminates with a policy that is ϵ-optimal. Hence using these two results we conclude that the simulation based policy iteration /ACTOR-CRITIC algorithm results in a policy that is optimal. We state this result as the following proposition.

Proposition 3.3 *The ACTOR-CRITIC algorithm converges to an optimal policy in a finite number of iterations.*

The actor critic algorithm assumes that every state is visited infinitely often and that every possible control action at that state is taken an infinite number of times. However note that the number of times that any control action other than the policy needs to be taken at a given state need only be a small fraction of the total number of times the state is visited. Hence the actor and the critic algorithms can be thought of as proceeding at two different time scales, the actor time scale being much greater than that of the critic time scale.Hence an ACTOR-CRITIC algorithm would typically be designed so that the ACTOR updates are relatively infrequent with respect to the CRITIC updates.

4 A Numerical Example

As an illustrative example, we consider the problem of a UAV (unmanned aerial vehicle) navigating in hostile territory. The speed of the UAV was assumed to be constant at a value v, and the control input to the UAV was the commanded heading angle,θ. The equations of motion of the system then can be written as

$$\dot{x} = Vcos\theta \tag{4.1}$$

$$\dot{y} = Vsin\theta \tag{4.2}$$

The continuous time problem was normalised so that the region of interest,(i.e., the region in which the UAV was confined to navigate), was a square of length 1 unit. The speed was taken as 0.05 units/sec. The problem was discretized (discrete time) by taking a sample time of 1sec. To get the problem into the discrete state space/ discrete control framework , we gridded the 1 unit square into 400 parts. Each grid had a representative point called the "exemplar". The control actions were discretised by requiring them to be one of the 8 directions on a map (N,NE,E,SE,S,SW,W,NW). The incremental cost function was assumed to be the square of the distance from the target which was chosen to be the

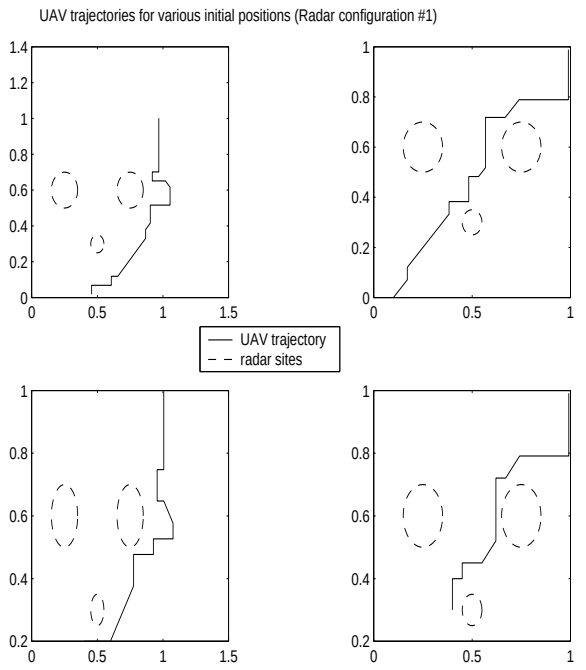

Figure 2: UAV trajectories for various initial positions(radar configuration 1)

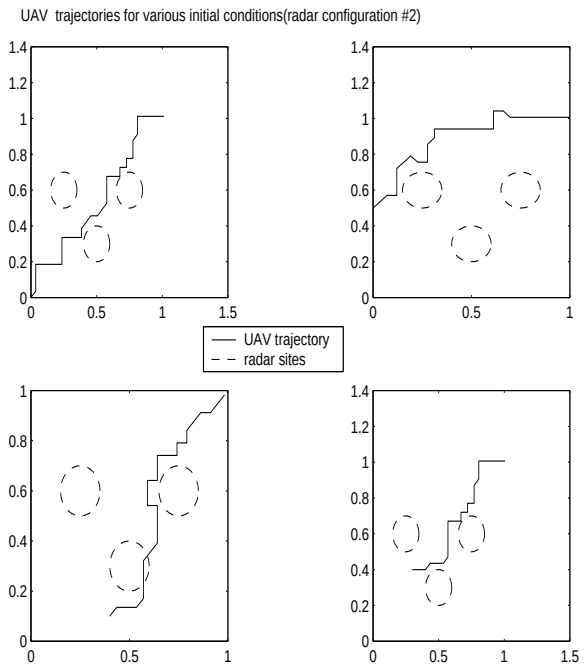

Figure 3: UAV trajectories for various initial positions(radar configuration 2)

point $(1,1)$. Also, if the UAV came inside the enemy radar or if it strayed from the 1x1 square, it was given a high positive cost. The incremental cost function for the dicrete state/control system was evaluated by finding the incremental cost incurred at the exemplar nearest to the current state. A discount factor of $\beta = 0.9$ was chosen for the problem. We used the simulation based policy iteration algorithm, presented in the previous section, in order to solve this problem. The policy evaluation step was carried out by randomly generating 15000 trajectories and evaluating the cost-to-go according to actor-critic algorithm. The trajectories were generated by randomly(uniformly) choosing an initial position and applying the current policy from that position till the UAV reached the target. The policy update was carried out using the γ-factors generated during policy evaluation as the policy evaluation step. As a choice for an initial heuristic policy, we chose a policy that would chose a heading angle that was closest to the slope of the straight line joining the UAV to the target. The dynamics of the UAV and the location of the enemy radar sites were assumed to be unknown to the UAV. We carried out the simulations for various configurations of the radar sites and the results are shown in the figures 1 and 2.

Figure 3 and figure 4 represent the optimal cost-to-go functions for the two different radar configurations.

It was noted during the course of our experiments with this example that in the regions which were not adequately explored, there was a corresponding deterioration of the performance of the UAV.

5 State Aggregation and Approximation Error

One of the biggest drawbacks of Dynamic Programming is the so called curse of dimensionality, i.e., as the dimension of the state space increases, the computational complexity increases exponentially. In the case of a finite state system, we still might have such a large number of states that DP might be an infeasible option. However, the state space of the problem can be reduced by aggregating states, i.e, reduce the original state space to a smaller one by partitioning it. However the question that immediately arises is: how close is the optimal cost-to-go function of the reduced system to the optimal cost-to-go function of the original system? In this section, we show that the ACTOR-CRITIC algorithm when

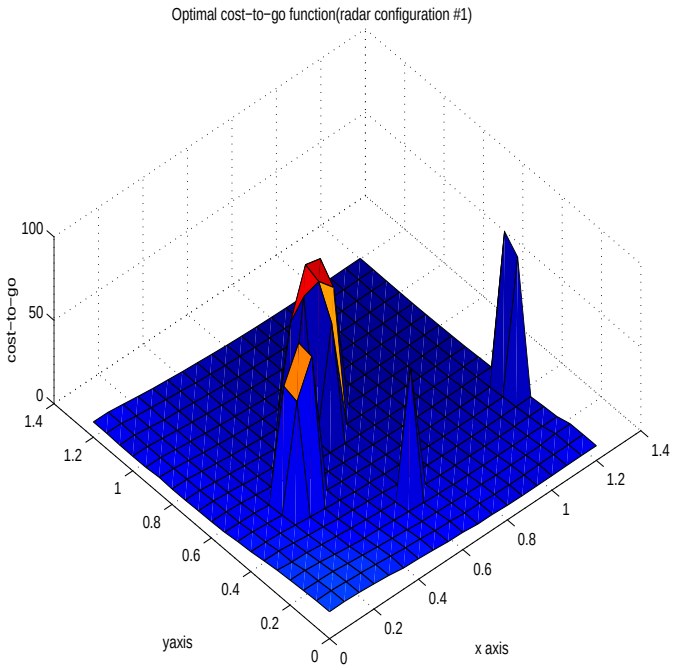

Figure 4: Optimal cost-to-go function(radar configuration 1)

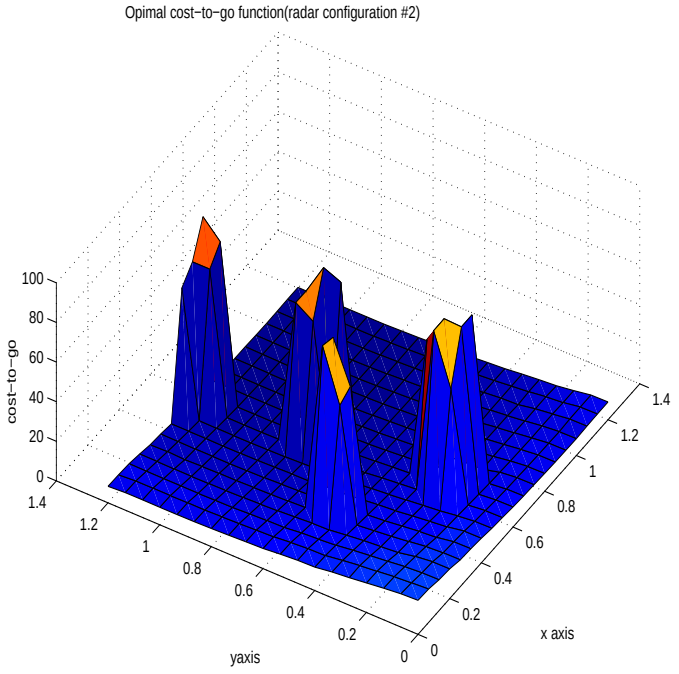

Figure 5: Optimal cost-to-go function(radar configuration 2)

run on a state aggregated system converges. However, we give an example which shows that unfortunately no guarantee can be given about the closeness of the optimal cost function of the reduced system to that of the original system.

Throughout this section, we assume that we are dealing with a system that has a finite state space. Let the state space of the syatem be denoted by $S = \{1, 2, ..N\}$. We partition the state space S into the sets $S_1, S_2, ...S_M$, where $M < N$. The ACTOR-CRITIC algorithm is now run on the reduced system, i.e., if the system makes the transition $i \rightarrow j$ under the control u, where $i \in S_i$ and $j \in S_j$, at time instant k, then the algorithm updates its γ-factors as:

$$\gamma_{k+1}(S_i, u) = max(c_k(i, \mu(i)) + \beta \bar{J}^k_\mu(j), \gamma_k(S_i, u))), \tag{5.1}$$

all the γ-factors being initialised to zero before a policy is evaluated. Here, i represents the actual state within S_i that the system is at time k and j is the state to which the system transitions.

Then, it trivially follows from Proposition 3.2, that

Proposition 5.1 *Given a stationary policy μ, and the ACTOR CRITIC algorithm modified as above, the gamma factors generated converge and the limit is given by*

$$\gamma_\mu(S_i, u) = \max_{j \in \Gamma(i,u)} (\bar{c}(S_i, u, S_j) + \beta \bar{J}_\mu(j))$$

where

$$\bar{J}_\mu(i) = \gamma_\mu(i, \mu(i))$$

$$\bar{c}(S_i, u, S_j) = \max_{i \in S} \max_{j \in \Gamma(i,u) \cap S_j} c(i, u, j)$$

However, nothing can be said about the closeness of the solution to the optimal cost-to-go function of the original system. To see this consider the 3-state system shown in figure 6. By closeness, we imply the following: if for the original system:

$$\max_{S_k} \max_{i,j \in S_k} |J^*(i) - J^*(j)| \leq \epsilon, \tag{5.2}$$

then what can be said about,

$$\max_{S_k} \max_{i \in S_k} |\bar{J}^*(S_k) - J^*(i)|. \tag{5.3}$$

The system consists of 3 states indexed 1,2 and 3. State 3 is a cost absorbing state, i.e., the system stays there once it gets there. There are two control actions possible at the other two states, STAY or MOVE. At state 1, if the system decides to move it incurs a cost of c_1 while if it chooses to stay it incurs an incremental cost of a_1. The costs

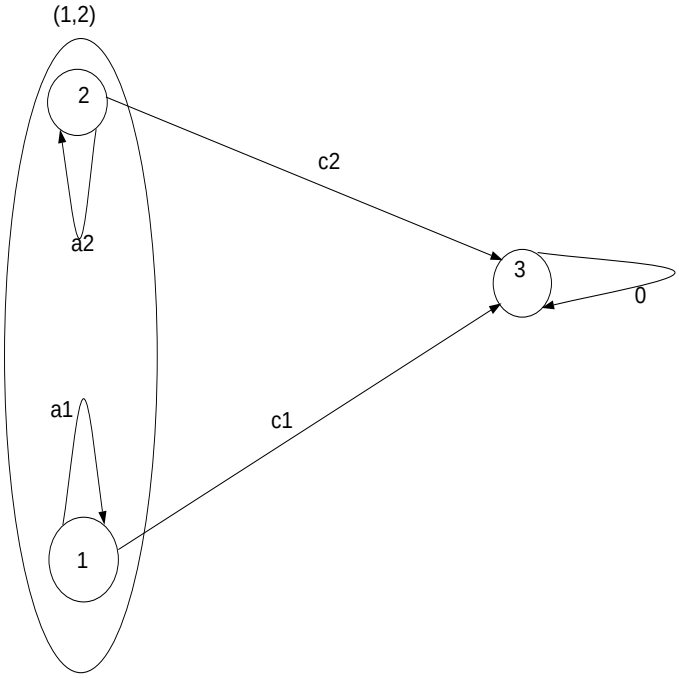

Figure 6: Example showing that errors can be arbitrarily large even for a perfect partition

for state 2 are similarly defined. Note that if the system decides to stay at either of states 1/2, then the system incurs a total cost of $\frac{a_1}{1-\beta}$ or $\frac{a_2}{1-\beta}$. Let $c_1 < \frac{a_1}{1-\beta}$,$c_2 > \frac{a_2}{1-\beta}$ and $c_1 = \frac{a_2}{1-\beta}$, i.e, the optimal policy at state 1 would be to move to state 3 while that at state 2 would be to stay there. Consider now the aggregated system $\{(1,2),3\}$. Suppose the system decides to stay at state(1,2) then it incurs a cost of c_2 while it incurs a cost of $\frac{a_1}{1-\beta}$ if it decides to move.(note that these are the worst case costs associated with the two control policies STAY and MOVE at the aggregated state (1,2) and is precisely what the ADAPTIVE CRITIC algorithm would find with respect to the selfsame policies). Note now that if we have $c_2 < \frac{a_1}{1-\beta}$, then the optimal policy at state(1,2), would be MOVE. The partition of the state space is perfect since there's no fluctuation of the optimal cost-to-go function within any of the partition, i.e., $\epsilon = 0$ in 5.2. However, we cannot give any error bounds since the approximation error in spite of the perfect state aggregation is $max(c_2 - \frac{a_2}{1-\beta}, c_2 - c_1)$, which is finite. Hence, just the assumption on the partition of the sytem as in 5.2 is not enough to assure any bounds on the approximation error.
The arbitrarily large approximation error above occurs because we try to approximate the optimal cost-to-go from a given state by the worst case cost from the aggregated state to which it belongs. The two can be different even for a perfect partition. However, suppose we approximate the cost-to-go from any state, by the worst optimal cost-to-go value of any state belonging to the partition containing it. In this example we have the optimal cost-to-go from state 1 as c_1 and that from 2 is $\frac{a_2}{1-\beta}$. Thuis if we approximate the optimal cost-to-go value from the aggregated state (1,2) by $max(c_1, \frac{a_2}{1-\beta})$, the error in approximation is zero. This is formalized in the following treatment. It follows the development in [19]. In the following the original system is assumed to have N states and the state aggregated system is assumed to have M states. Let $\tilde{V} : \Re^M \rightarrow \Re^N$ such that

$$\tilde{V}_i(J) = J_j \ \forall i \in S_j \tag{5.4}$$

The above equation is a compact form of representing the state aggregation scheme. In the rest of the section, we assume that the minimax DP operator T is known to us, i.e., the model of the system is present. We note that the developments generalize easily to the model-free case for both Q-learning as well as Adaptive Critics. Let T denote the minimax DP operator for the original system. Let Γ^j represent set of instants that the set S_j is visited.Let $\bar{J}(t)$ denote the estimate of the cost-to-go vector of the aggregate system at instant t.Consider some instant $t \in \Gamma^j$, let the state visited in S_j at that instant be i. Then the algorithm is stated as follows

$$\bar{J}_j(t+1) = max(T_i(\tilde{V}(\bar{J}(t))), \bar{J}_j(t)), \ t \in \Gamma^j$$
$$\bar{J}_j(t+1) = \bar{J}_j(t), o.w. \tag{5.5}$$

The following Proposition can be made about the sequence $\bar{J}(t)$

Proposition 5.2 *The sequence $\bar{J}(t)$ converges to $\bar{J}^*$ which is the unique vector satisfying the system of equations*

$$\bar{J}^*_j = \max_{i \in S_j} T_i(\tilde{V}(\bar{J}^*)) \tag{5.6}$$

Let J^* denote the optimal cost-to-go function for the original system and

$$\epsilon = \max_{S_k} \max_{i,j \in S_k} |J^*_i - J^*_j| \tag{5.7}$$

then

$$||\tilde{V}(\bar{J}^*) - J^*||_\infty \leq \frac{\epsilon}{1-\beta} \tag{5.8}$$

where $||.||_\infty$ refers to the supremum norm in $\Re^N$.

Proof: See appendix.
Note that the error bound has a factor of $(1-\beta)$

in the denominator which would tend to infinity as $\beta \to 1$. However, note that this is a worst case bound and most real systems would have much better performance than the worst case bound.

6 Conclusion

In this paper, we have presented a deterministic worst case treatment of uncertainty, applied to a sequential decision-making problem. This method has been variously called in literature as minimax / worst case DP. We have introduced minimax adaptive critics and shown the convergence of the learning algorithms associated with it. Also, we have presented bounds on the errors that result from approximating a higher order system by a lower order system, specifically the case of state aggregation. We think that this is a first step in the direction of function approximations in minimax reinforcement learning. Getting error bounds on such approximations is very important because any kind of practical application of reinforcement learning would typically involve some kind of function approximation architecture. Having said this, we should be guarded since this aspect of reinforcement learning has proved to be very difficult to tackle in the case of the probabilistic DP. However, we would like to stress the simplicity of all the learning algorithms associated with minimax reinforcement learning. This makes these methods particularly easy to implement. It may be argued that these methods would suffer from a lot of conservatism. However conservatism results in these methods when there is a great deal of uncertainty in the system. Such poorly identified systems would be very difficult to control. Therefore, the conservatism shown by these methods become particularly attractive when faced with the problem of on-line real time decision-making in expensive systems like aerospace systems. The pessimistic approach of these methods are particularly suitable since we can ill-afford to lose a million dollar aircraft or spacecraft. Another point of interest is that through the formulation of next state sets as mentioned in the paper, the case of incomplete state measurements can be satisfactorily treated by these methods.

A Appendix

In this appendix, we shall discuss multi-representation contractions and show how they are used to get the proof of proposition5.2.

In Dynamic Programming, we require the solution of a fixed point equation of the form $TJ^* = J^*$ where T represents the DP operator and J^* represents the optimal cost-to-go vector. We know that T is a contraction mapping and that J^* is its unique fixed point. One typical way of getting to this solution is to get the sequence $\{J^1, TJ^1, T^2J^1, ...\}$ which converges to J^* and is the method of succesive approximations. However if the dimension of the vector J^* , say n,is high then the computational complexity makes the calculation of the optimal cost-to-go vector inherently slow. One way of speeding up the calculation is by mapping the problem into a smaller space $\Re^m (m << n)$ which can be thought of as a paramter space. This can be done by defining the mapping $\tilde{V} : \Re^m \to \Re^n$ and the pseudo-inverse mapping $\tilde{V}^\dagger : \Re^n \to \Re^m$. The mapping $\tilde{V}$ can be thought of as a compact representation. The solution of the original problem can be approximated by finding the fixed point of the map $T^{'} : \Re^m \to \Re^m$ where $T^{'} = V^\dagger \circ T \circ \tilde{V}$. The hope is that $\tilde{V}(\bar{J}^*)$ is close to the fixed point J^* of T where $\bar{J}^*$ is the fixed point of $T^{'}$.We employ a vector norm for both $\Re^n, \Re^m$ and denote both by $||.||$ and make the following assumptions:

$\mathcal{A}$ **A.1** *The mapping T is a contraction with contraction coefficient $\beta \in [0,1)$ with respect to the norm $||.||$. Hence for all $J, J^{'} \in \Re^n$*

$$||TJ - TJ^{'}|| \leq \beta ||J - J^{'}|| \qquad \text{(A.1)}$$

The second assumption defines the relationship between $\tilde{V}$ and $\tilde{V}^\dagger$.

$\mathcal{A}$ **A.2** *(a) for all $\bar{J} \in \Re^m$*

$$\bar{J} = \tilde{V}^\dagger(\tilde{V}(\bar{J})) \qquad \text{(A.2)}$$

(b) there exists a $\beta^{'} \in [\beta, 1)$ such that, for all $\bar{J}, \bar{J}^{'} \in \Re^m$,

$$||\tilde{V}(\bar{J}) - \tilde{V}(\bar{J}^{'})|| \leq \frac{\beta^{'}}{\beta} ||\bar{J} - \bar{J}^{'}|| \qquad \text{(A.3)}$$

(c) for all $J, J^{'} \in \Re^n$,

$$||\tilde{V}^\dagger(J) - \tilde{V}^\dagger(J^{'})|| \leq ||J - J^{'}|| \qquad \text{(A.4)}$$

Part(a) forces $\tilde{V}^\dagger$ to be the pseudo-inverse of $\tilde{V}$.Part(b) ensures that points that are close in $\Re^m$ map to close points in $\Re^n$ while part(c) ensures the opposite direction.Let

$$\epsilon = \inf_{\bar{J} \in \Re^m} ||J^* - \tilde{V}(\bar{J})||$$

Under the above assumptions the following theorem holds[19],

Proposition A.1 *(a) We have*

$$||T^{'}(\bar{J}) - T^{'}(\bar{J}^{'})|| \leq \beta^{'}||\bar{J} - \bar{J}^{'}||$$

(b) If $\bar{J}^*$ is the fixed point of $T^{'}$, then

$$||J^* - \tilde{V}(\bar{J}^*)|| \leq \frac{\beta + \beta^{'}}{\beta(1-\beta^{'})}\epsilon \qquad (A.5)$$

Now we are ready to proceed with the proof of Proposition5.2.

Proof of Proposition5.2:
In our case we have

$$\tilde{V}_i(\bar{J}) = W_j, \ \forall i \in S_j \qquad (A.6)$$

$$\tilde{V}_j^{\dagger}(J) = \max_{i \in S_j} J_i, \qquad (A.7)$$

Now we shall verify the assumptions regarding the multi-representation contractions. The first assumption is satisfied since the DP operator T is a contraction mapping with a factor $\beta < 1$. Let $W = \{W_1, W_2, ..., W_m\}$.
Then

$$W_j = \tilde{V}_j^{\dagger}(\tilde{V}(W)) = \max_{i \in S_j} \tilde{V}_i(W) = W_j \qquad (A.8)$$

Hence we have that $\tilde{V}_j^{\dagger}(\tilde{V}(W)) = W$.
Next we have that if $i \in S_j$

$$|\tilde{V}_i(W) - \tilde{V}_i(W^{'})| \leq |W_j - W_j^{'}| \qquad (A.9)$$

which means that

$$||\tilde{V}(W) - \tilde{V}(W^{'})|| \leq ||W - W^{'}|| \qquad (A.10)$$

i.e. in this case $\beta^{'} = \beta$.
We also have that

$$|\tilde{V}_j^{\dagger}(J) - \tilde{V}_j^{\dagger}(J^{'})| = |\max_{i \in S_j} J_i - \max_{i \in S_j} J_i^{'}| \leq \max_{i \in S_j} |J_i - J_i^{'}| \qquad (A.11)$$

$\Rightarrow$

$$||\tilde{V}^{\dagger}(J) - \tilde{V}^{\dagger}(J^{'})|| \leq \max_j \max_{i \in S_j} |J_i - J_i^{'}| = ||J - J^{'}|| \qquad (A.12)$$

Hence all the assumptions of the multi representation contractioin theorem are satisfied . Therefore it follows that the operator $T^{'} = \tilde{V}^{\dagger} \circ T \circ \tilde{V}$ is also a contraction mapping with contraction factor $\beta^{'} = \beta$. Note that $\bar{J}_j(t)$ is a monotonically non-decreasing sequence and bounded above for all S_j. Hence it converges to some $\bar{J}_j^{\infty}$. Now we show that $\bar{J}^{\infty}$ is the fixed point of the operator T'.
Let

$$\lim \bar{J}(t) = \bar{J}^{\infty} \qquad (A.13)$$

$$\max_{i \in S_j} T_i(\tilde{V}(\bar{J}^{\infty})) = K_j^{\infty} \qquad (A.14)$$

Let Γ^j denote the set of times at which S_j is visited. From the fact that $\bar{J}_j(t)$ is a monotonically non-decreasing sequence,we have

$$\bar{J}_j(t) \leq \bar{J}_j^{\infty} \ \forall j \ \forall t \qquad (A.15)$$

Let

$$K_j(t) = T_{i(t)}(\tilde{V}(\bar{J}(t))) \ \forall t \in \Gamma^j \qquad (A.16)$$

$i(t) \in S_j$ denotes the state of the system at time t. Hence from the definition of the sequence $\bar{J}_j(t)$, it follows that

$$\{\bar{J}_j(t)\} \subseteq \{K_j(t)\} \qquad (A.17)$$

$$\Rightarrow$$

$$\sup\{\bar{J}_j(t)\} \leq \sup\{K_j(t)\} \qquad (A.18)$$

Given $\epsilon > 0$ there exists $t_\epsilon < \infty$,$t_\epsilon \in \Gamma^j$ s.t. $\sup\{K_j(t)\} \leq T_{i(t_\epsilon)}(\tilde{V}(\bar{J}(t_\epsilon))) + \epsilon$.
Note that $\bar{J}_1 \geq \bar{J}_2$ means that $\tilde{V}(\bar{J}_1) \geq \tilde{V}(\bar{J}_2)$. Hence from the monotonicity of t,

$$\begin{aligned} \sup\{k_j(t)\} &\leq T_{i(t_\epsilon}(\tilde{V}(\bar{J}(t_\epsilon))) + \epsilon \\ &\leq T_{i(t_\epsilon)}(\tilde{V}(\bar{J}^{\infty})) + \epsilon \leq K_j^{\infty} + \epsilon \end{aligned} \qquad (A.19)$$

Since the above relation holds for all $\epsilon > 0$, we conclude that $\sup\{K_j(t)\} \leq K_j^{\infty}$. Hence

$$\bar{J}_j^{\infty} \leq K_j^{\infty} \qquad (A.20)$$

Let $t < \infty$ be some arbitrary instant. Let $t + t_i$ represent the instant at which state $i \in S_j$ is visited by the system the first time after instant t.Let

$$t_\infty = \max_{i \in S_j}(t + t_i) \qquad (A.21)$$

which means that

$$\begin{aligned} \bar{J}_j^{\infty} \geq \bar{J}_j(t_\infty) &\geq \max_{i \in S_j} T_i(\tilde{V}(\bar{J}(t + t_i))) \\ &\geq \max_{i \in S_j} T_i(\tilde{V}(\bar{J}(t))) \end{aligned} \qquad (A.22)$$

The above relation follows from the monotonicity of the DP operator T.
Since $K_j^{\infty} \geq \bar{J}_j^{\infty}$, we get

$$\begin{aligned} |\bar{J}_j^{\infty} - K_j^{\infty}| &\leq |\max_{i \in S_j} T_i(\tilde{V}(\bar{J}(t))) - \max_{i \in S_j} T_i(\tilde{V}(\bar{J}^{\infty}))| \\ &\leq \max_{i \in S_j} |T_i(\tilde{V}(\bar{J}(t))) - T_i(\tilde{V}(\bar{J}^{\infty}))| \\ &\leq \max_{i \in S_j} |T\tilde{V}\bar{J}(t) - T\tilde{V}\bar{J}^{\infty}| \\ &\leq \beta |\bar{J}(t) - \bar{J}^{\infty}| \end{aligned} \qquad (A.23)$$

Let t_δ be such that for all $t > t_\delta, |\bar{J}(t) - \bar{J}^\infty| < \delta$. Then it follows that

$$|\bar{J}_j^\infty - K_j^\infty| < \beta\delta \tag{A.24}$$

$$\Rightarrow$$

$$\bar{J}_j^\infty \geq K_j^\infty - \beta\delta \tag{A.25}$$

Noting that δ can be made arbitrarily small, it follows form eqsA.20 andA.25 that $K_j^\infty = \bar{J}_j^\infty$. Also the above holds for all S_j. Hence it follows that $\bar{J}^\infty$ is the unique fixed point of the operator T'.
To obtain the error bounds on the approximation we use the second part of PropositionA.1.If the maximum fluctuation of J^* within any particular partition is less than ϵ then $\inf_{\bar{J}} ||\hat{V}(\bar{J}) - J^*||_\infty = \epsilon/2$. This is the case since the best a constant can approximate the cost-to-gos within a particular partition is going to be $\frac{\epsilon}{2}$. Hence substituitng $\frac{\epsilon}{2}$ in PropositionA.1(b), we get the desired result.
q.e.d

References

[1] A. G. Barto, S. J. Bradtke and S. P. Singh, "Learning to Act using RTDP", *Artificial Intelligence*, 72(1995), pp 81-138.

[2] A. G. Barto, R. Sutton and C. Anderson, "Neuron -like Elements that can solve difficult learning control problems", *IEEE trans. Syst. Man and Cyb.*, 13(1983), pp 835-846.

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

[4] D. P. Bertsekas, J. N. Tsitsiklis, *Neuro Dynamic Programming*, Belmont, Mass:Athena Scientific, 1997

[5] S. P. Coraluppi, S. I. Marcus, "Risk Sensitive and Minimax Control of discrete-time, finite sate MDP", *Automatica*, v35(1999),pp 301-309

[6] S. P. Coraluppi, S. I. Marcus, "Mixed risk-neutral / Minimax control discrete time finite state Markov Processes", *IEEE trans aut.con.*,v35 n.3,2000

[7] E. V. Denardo, "Contraction mappings in the theory underlying Dynamic Programming", *SIAM Review*, v9, n2, April 1967.

[8] D.P.Bertsekas, *Dynamic Programming: Deterministic and Stochastic models*, Englewood cliffs, NJ:Prentice Hall, 1987

[9] M. Heger, "Consideration of Risk in Reinforcement Learning", *Proceedings of the 11th Int. Conf. on Mach. Learn.*, pp 105-111,1994, Morgan Kaufman Publishers Inc., San Francisco, CA.

[10] M. Heger, "The Loss from Imperfect Value Functions in Expectation-based and Minimax-based Tasks", Machine Learning, 22, 1996, pp197-225

[11] R. A. Howard, *Dynamic Programming and Markov Processes*, Published jointly by the Technology Press of the Massachusetts Institute of Technology and Wiley, New York,1960

[12] G. Jiang, Wu Cang-Pu, G. Cybenko, "Minimax Based Reinforcement Learning with State Aggregation", *Proceeedings of the IEEE Conference on Decision and Control*, 1998, v2, pp 1236-1241.

[13] J. R. Munkres, *Topology: A First Course*, Prentice-Hall, Englewood Cliffs,NJ,1974

[14] S. M. Ross, *Introduction to stochastic dynamic programming*, Academic Press, NY, 1983.

[15] C. Szevespari, "Learning and Exploitation do not Conflict under Minimax optimality", *Machine Learning:ECML-97, 9th European Conference on Machine Learning*, Springer-Verlag, Berlin, Germany,1997,pp242-249

[16] C. Szevespari, M. Littman, "A Unified Analysis of Value Function based Reinforcement Learning Algorithms", *Neural Computation*, 11, 2017-2060 (1999)

[17] C Szevespari, M Littman, "Generalized Reinforcement Learning Model: Convergence and Application", *13th International Conference on Machine Learning*, 1996,Morgan Kaufman Publishers Inc., San Francisco, CA.

[18] J N Tsitsiklis, "Asynchronous Stochastic Approximations and Q-Learning", *Machine Learning*, 16(1994), pp 185-202

[19] J. N. Tsitsiklis and B. Van Roy, "Feature Based Methods for Large Scale Dynamic Programming", Machine Learning, v22, pp 59-94(1996).

[20] S. Verdu and H. V. Poor, "Abstract Dynamic Programming Models under Commutativity Conditions", *SIAM J Cont. optim.*, v25 n.4, 1987

[21] J. White and D.Sofge, eds, "Handbook of Intelligent control", Van Nostrand, 1992

[22] C. J. C. H. Watkins and P. Dayan, "Q-Learning", Machine Learning, 8(1992), pp 279-292