ECE64700: Homework IV

Bharath Keshavamurthy

I. Markov Decision Processes - An Energy Harvesting Radio Node

A. System Model

We consider a single energy harvesting radio node with an MDP agent running as a part of its decision engine solving for the optimal transmission policy considering channel fading states and battery energy levels.

The channel fading states g_k realize from the following set i.i.d over time with a uniform probability distribution $\mathbb{P}(g_k = \gamma_i) = \theta_i = \frac{1}{N}$.

$$\mathcal{G} \equiv \{\gamma_i,\ i = 1,\ 2,\ 3,\ ...,\ N\} \tag{1}$$

The battery states Q_k realize from the following set.

$$\mathcal{Q} \equiv \{mE_0,\ m = 0,\ 1,\ 2,\ 3,\ ...,\ M\} \tag{2}$$

The battery state update procedure is as follows.

$$Q_{k+1} = \min\{Q_k - w_k + a_k,\ ME_0\}$$

where,

$w_k \in \{0,\ E_0\}$ is the amount of energy spent by the radio node during transmission, and

$a_k \in \{0,\ E_0\}$ is modelled as a Bernoulli i.i.d process with $\mathbb{P}(a_k = E_0) = p$ as the success probability and $\mathbb{P}(a_k = 0) = (1-p)$ as the failure probability.

1) The State Space: The state space for the MDP agent is denoted by $\mathcal{S}$.

In time-slot k, we represent the state of the process as a 2-tuple or a pair $(Q_k,\ g_k)$ where,

$Q_k \in \mathcal{Q}$ is defined as the current battery level or the amount of energy available in the battery in time-slot k, and

$g_k \in \mathcal{G}$ is defined as the current state of the fading channel.

The dimensionality of the state space is given by $|\mathcal{S}| = MN = 110$ possible states.

Therefore, concretely,

$$\mathcal{S} \equiv \{(Q_k,\ g_k) \mid Q_k \in \mathcal{Q},\ g_k \in \mathcal{G}\} \tag{3}$$

2) The Action Space: The action space of the MDP agent is denoted by $\mathcal{U}$.

In time-slot k, the actions available to the MDP agent are,

$$u_k \;=\; \begin{cases} 0, \text{ if } Q_k \;=\; 0 \text{ and } g_k \in \mathcal{G} \\ \{1,\ 2,\ 3,\ ...,\ i\}, \text{ if } Q_k \;\neq\; 0 \text{ and } g_k \;=\; \gamma_i \in \mathcal{G} \end{cases} \tag{4}$$

If the battery is dead, the radio node cannot transmit anything. Hence, the only available action in states with $Q_k \;=\; 0$ is to remain idle, i.e. $u_k \;=\; 0$.

If there are units of energy in the battery (the units are discretized in steps of E_0 as it is evident from equation (2)), the actions of the radio node depends on the realization of the fading state of the channel.

3) The Transition Model: The transition model of the process if given by,

$$(Q_{k+1},\ g_{k+1}) \;=\; \begin{cases} (Q_k,\ g_{k+1}), \text{ if } u_k \;=\; 0 \text{ and } a_k \;=\; 0 \textbf{ with probability } \frac{1-p}{N} \\ (Q_k - E_0,\ g_{k+1}), \text{ if } u_k \neq 0 \text{ and } a_k \;=\; 0 \textbf{ with probability } \frac{1-p}{N} \\ (\min\{Q_k + E_0,\ ME_0\},\ g_{k+1}), \text{ if } u_k \;=\; 0 \text{ and } a_k \;=\; E_0 \textbf{ with probability } \frac{p}{N} \\ (Q_k,\ g_{k+1}), \text{ if } u_k \;\neq 0 \text{ and } a_k \;=\; E_0 \textbf{ with probability } \frac{p}{N} \end{cases} \tag{5}$$

4) The Reward Metric: The reward metric is denoted by $r_k \;=\; R(s_k,\ u_k)$.

In time-slot k, we model the reward to the MDP agent as the number of bits it could transmit over the channel given the process state $s_k \;=\; (Q_k,\ g_k) \in \mathcal{S}$, i.e. $r_k \;=\; u_k$.

B. The Problem Formulation

In order to solve for the optimal policy to be followed by the MDP agent residing in the decision engine of the energy harvesting radio node, we formulate the following problem which turns out to be the **Bellman Optimality Equation for Infinite Horizon Discounted Reward Problems**.

$$V^*(s) \;=\; \max_{u \in \mathcal{U}(s)} \; R(s,\ u) + \rho \sum_{s' \in \mathcal{S}} \; P_{s'|s,u} V^*(s') \tag{6}$$

where,

$\rho \;=\; 0.9$ is the discount factor,

$P_{s'|s,u}$ is the transition probability detailed in the transition model (5), and

$R(s,\ u)$ is the reward received in state s after taking action u.

C. Solution Methodology and Performance

1) The Algorithms: We employ the Value Iteration Algorithm and the Policy Iteration Algorithm to solve for the optimal policy as shown below.

- **Value Iteration**

Algorithm 1 The Value Iteration Algorithm for the given MDP agent

Result: The optimal policy μ^* and optimal value function $V^*(s), \forall s \in \mathcal{S}$

Initialization: Start with an initialization of the value function, $V_0(s) = 10^{-5}, \ \forall s \in \mathcal{S}$;

while $\forall s \in \mathcal{S}, V_{k+1}(s) \neq V_k(s)$ **do**

$$V_{k+1}(s) = \max_{u \in \mathcal{U}(s)} \left[u + \rho \sum_{s' \in \mathcal{S}} P_{s'|s,u} V_k(s') \right]$$

$$u_k(s) = \arg\max_{u \in \mathcal{U}(s)} \left[u + \rho \sum_{s' \in \mathcal{S}} P_{s'|s,u} V_k(s') \right]$$

end

Return $V^*(s), \ \forall s \in \mathcal{S}$ and $\mu^* = \{u^*(s) \mid s \in \mathcal{S}\}$ after convergence

- **Policy Iteration**

Algorithm 2 The Policy Iteration Algorithm for the given MDP agent

Result: The optimal policy μ^* and optimal value function $V^*(s), \forall s \in \mathcal{S}$

Initialization: Start with an arbitrary policy $\mu_0 = \{0 \ (idle), \ \forall s \in \mathcal{S}\}$ and an initialization of the value function, $V_0(s) = 10^{-5}, \ \forall s \in \mathcal{S}$;

while $\forall s \in \mathcal{S}, \mu_{k+1}(s) \neq \mu_k(s)$ **do**

Policy Evaluation:

$$V_{\mu_k}(s) = \left[\mu_k(s) + \rho \sum_{s' \in \mathcal{S}} P_{s'|s,u} V_{\mu_k}(s') \right]$$

Policy Improvement:

$$\mu_{k+1}(s) = \arg\max_{u \in \mathcal{U}(s)} \left[u + \rho \sum_{s' \in \mathcal{S}} P_{s'|s,u} V_{\mu_k}(s') \right]$$

end

Return $V^*(s), \ \forall s \in \mathcal{S}$ and μ^* after convergence

Please refer to the provided source code (*EnergyHarvestingNode.py*) for more details.

2) Results of the Algorithms:

- **Value Iteration**

  The optimal policy from the Value Iteration Algorithm is given below.

  '0_1': 0, '0_2': 0, '0_3': 0, '0_4': 0, '0_5': 0, '0_6': 0, '0_7': 0, '0_8': 0, '0_9': 0, '0_10': 0,

  '1_1': 0, '1_2': 0, '1_3': 0, '1_4': 0, '1_5': 0, '1_6': 0, '1_7': 7, '1_8': 8, '1_9': 9, '1_10': 10,

  '2_1': 0, '2_2': 0, '2_3': 0, '2_4': 0, '2_5': 5, '2_6': 6, '2_7': 7, '2_8': 8, '2_9': 9, '2_10': 10,

  '3_1': 0, '3_2': 0, '3_3': 0, '3_4': 0, '3_5': 5, '3_6': 6, '3_7': 7, '3_8': 8, '3_9': 9, '3_10': 10,

  '4_1': 0, '4_2': 0, '4_3': 0, '4_4': 4, '4_5': 5, '4_6': 6, '4_7': 7, '4_8': 8, '4_9': 9, '4_10': 10,

  '5_1': 0, '5_2': 0, '5_3': 3, '5_4': 4, '5_5': 5, '5_6': 6, '5_7': 7, '5_8': 8, '5_9': 9, '5_10': 10,

  '6_1': 0, '6_2': 0, '6_3': 3, '6_4': 4, '6_5': 5, '6_6': 6, '6_7': 7, '6_8': 8, '6_9': 9, '6_10': 10,

  '7_1': 0, '7_2': 0, '7_3': 3, '7_4': 4, '7_5': 5, '7_6': 6, '7_7': 7, '7_8': 8, '7_9': 9, '7_10': 10,

  '8_1': 0, '8_2': 2, '8_3': 3, '8_4': 4, '8_5': 5, '8_6': 6, '8_7': 7, '8_8': 8, '8_9': 9, '8_10': 10,

  '9_1': 0, '9_2': 2, '9_3': 3, '9_4': 4, '9_5': 5, '9_6': 6, '9_7': 7, '9_8': 8, '9_9': 9, '9_10': 10,

  '10_1': 1, '10_2': 2, '10_3': 3, '10_4': 4, '10_5': 5, '10_6': 6, '10_7': 7, '10_8': 8, '10_9': 9, '10_10': 10

  Here, $0\_1 : 0$ refers to the optimal action of remaining idle, i.e. $u_k = 0$, if the process state is $(Q_k = 0,\ g_k = \gamma_1)$.

  Similarly, $10\_7 : 7$ refers to the optimal action of sending the maximum allowed bits of 7 if the process state is $(Q_k = 10,\ g_k = \gamma_7)$.

  Some sample state value functions are:

  $(Q_k = 0,\ g_k = \gamma_1)$: 6.484672466785633

  $(Q_k = 2,\ g_k = \gamma_4)$: 17.801564174937106

  $(Q_k = 5,\ g_k = \gamma_2)$: 28.273127961666923

  $(Q_k = 10,\ g_k = \gamma_1)$: 37.55420277283401

  $(Q_k = 10,\ g_k = \gamma_{10})$: 46.55420700996909

- **Policy Iteration**

  The optimal policy from the Policy Iteration algorithm is given by,

  '0_1': 0, '0_2': 0, '0_3': 0, '0_4': 0, '0_5': 0, '0_6': 0, '0_7': 0, '0_8': 0, '0_9': 0, '0_10': 0,

  '1_1': 0, '1_2': 0, '1_3': 0, '1_4': 0, '1_5': 0, '1_6': 0, '1_7': 7, '1_8': 8, '1_9': 9, '1_10': 10,

  '2_1': 0, '2_2': 0, '2_3': 0, '2_4': 0, '2_5': 5, '2_6': 6, '2_7': 7, '2_8': 8, '2_9': 9, '2_10': 10,

  '3_1': 0, '3_2': 0, '3_3': 0, '3_4': 0, '3_5': 5, '3_6': 6, '3_7': 7, '3_8': 8, '3_9': 9, '3_10': 10,

  '4_1': 0, '4_2': 0, '4_3': 0, '4_4': 4, '4_5': 5, '4_6': 6, '4_7': 7, '4_8': 8, '4_9': 9, '4_10': 10,

  '5_1': 0, '5_2': 0, '5_3': 3, '5_4': 4, '5_5': 5, '5_6': 6, '5_7': 7, '5_8': 8, '5_9': 9, '5_10': 10,

  '6_1': 0, '6_2': 0, '6_3': 3, '6_4': 4, '6_5': 5, '6_6': 6, '6_7': 7, '6_8': 8, '6_9': 9, '6_10': 10,

  '7_1': 0, '7_2': 0, '7_3': 3, '7_4': 4, '7_5': 5, '7_6': 6, '7_7': 7, '7_8': 8, '7_9': 9, '7_10': 10,

  '8_1': 0, '8_2': 2, '8_3': 3, '8_4': 4, '8_5': 5, '8_6': 6, '8_7': 7, '8_8': 8, '8_9': 9, '8_10': 10,

  '9_1': 0, '9_2': 2, '9_3': 3, '9_4': 4, '9_5': 5, '9_6': 6, '9_7': 7, '9_8': 8, '9_9': 9, '9_10': 10,

  '10_1': 1, '10_2': 2, '10_3': 3, '10_4': 4, '10_5': 5, '10_6': 6, '10_7': 7, '10_8': 8, '10_9': 9, '10_10': 10

  Here, $0\_1 : 0$ refers to the optimal action of remaining idle, i.e. $u_k = 0$, if the process state is $(Q_k = 0,\ g_k = \gamma_1)$.

  Similarly, $10\_7 : 7$ refers to the optimal action of sending the maximum allowed bits of 7 if the process state is $(Q_k = 10,\ g_k = \gamma_7)$.

  Some sample state value functions are:

  $(Q_k = 0,\ g_k = \gamma_1)$: 6.484553119956839

  $(Q_k = 2,\ g_k = \gamma_4)$: 17.801464754820753

  $(Q_k = 5,\ g_k = \gamma_2)$: 28.27304768957006

  $(Q_k = 10,\ g_k = \gamma_1)$: 37.554155315117185

  $(Q_k = 10,\ g_k = \gamma_{10})$: 46.554160324453505

Please refer to the provided Console Logs for more details on the results of these two algorithms.

Although Value Iteration and Policy Iteration are comparable in their performance due to smaller state and action spaces in this process, we can see from the plots that Value Iteration is slightly faster than Policy Iteration.

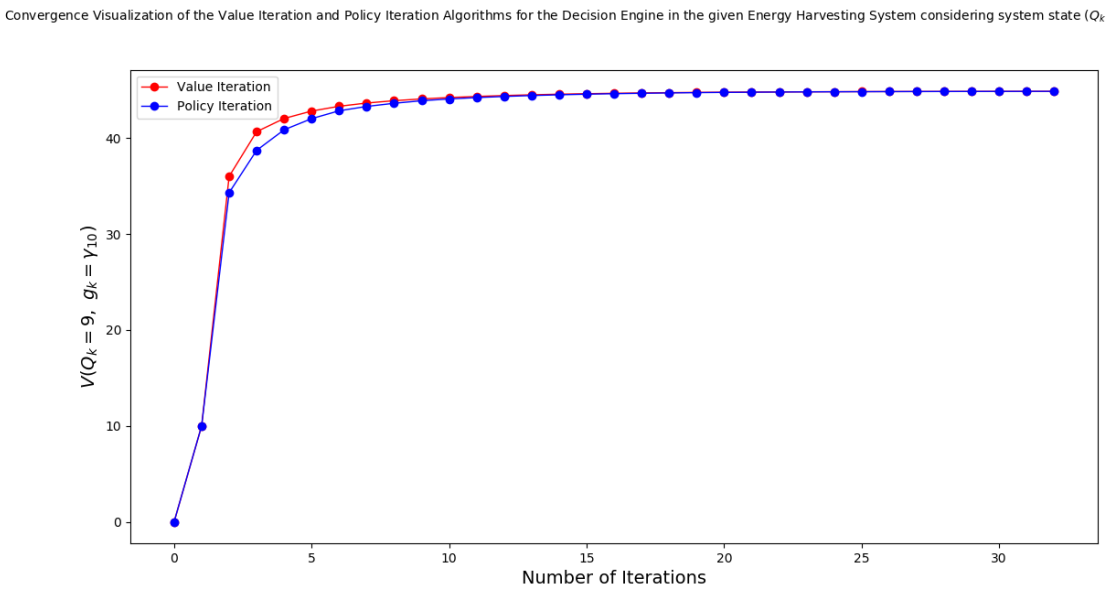

Fig. 1. Convergence Visualization of the Value Iteration and Policy Iteration Algorithms for the Decision Engine in the given Energy Harvesting System considering system state $(Q_k = 9,\ g_k = \gamma_{10})$

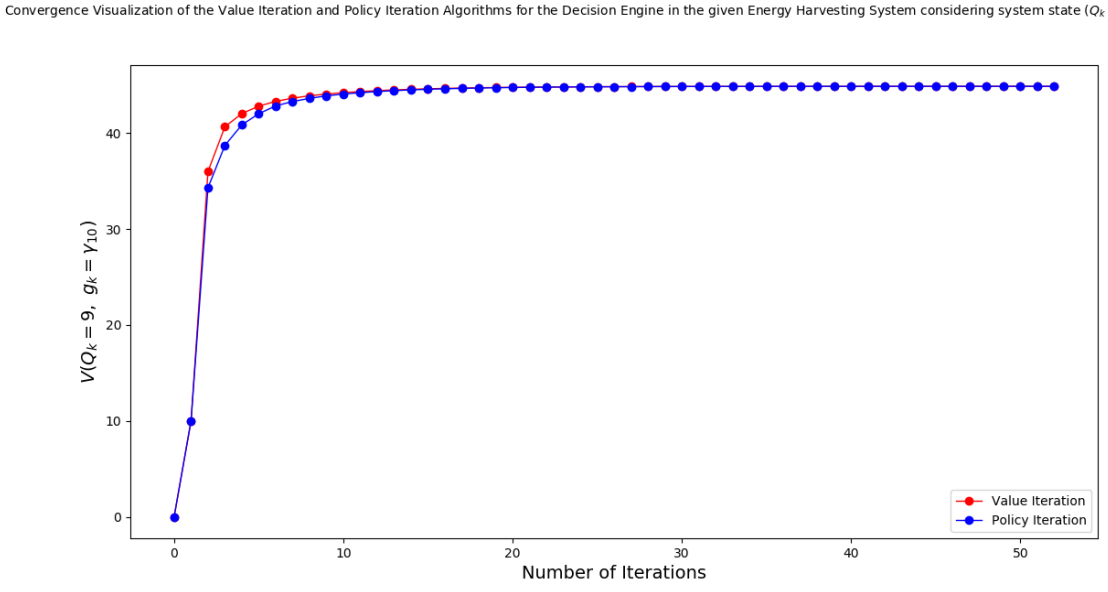

Fig. 2. Convergence Visualization of the Value Iteration and Policy Iteration Algorithms for the Decision Engine in the given Energy Harvesting System considering system state $(Q_k = 9,\ g_k = \gamma_{10})$ - Increased number of iterations to prove convergence

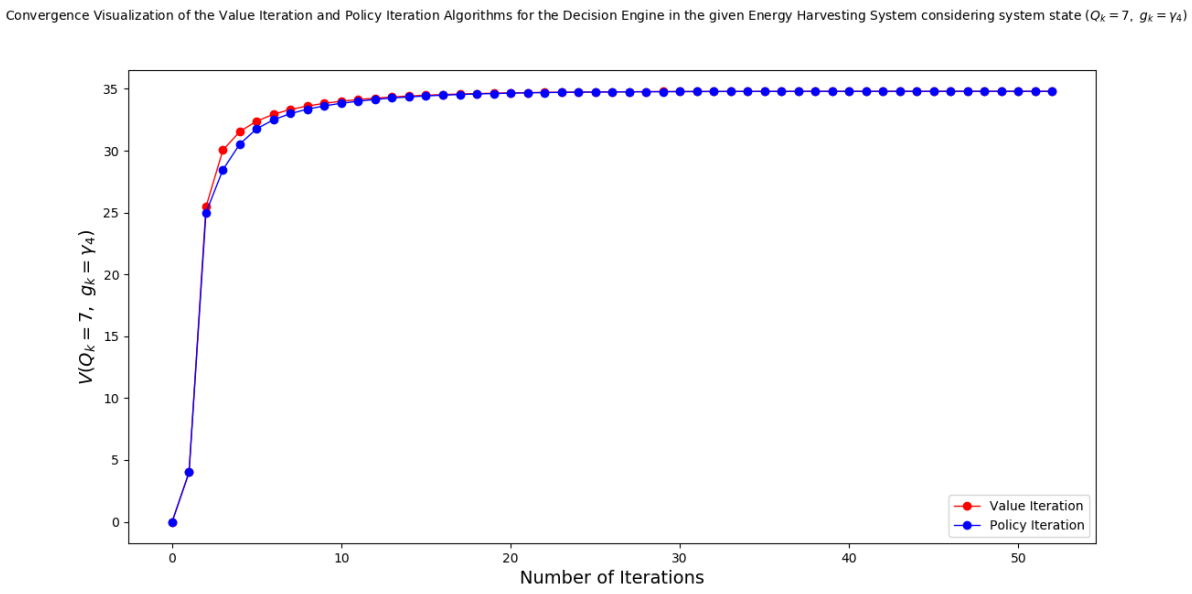

Fig. 3. Convergence Visualization of the Value Iteration and Policy Iteration Algorithms for the Decision Engine in the given Energy Harvesting System considering system state $(Q_k = 7,\ g_k = \gamma_4)$

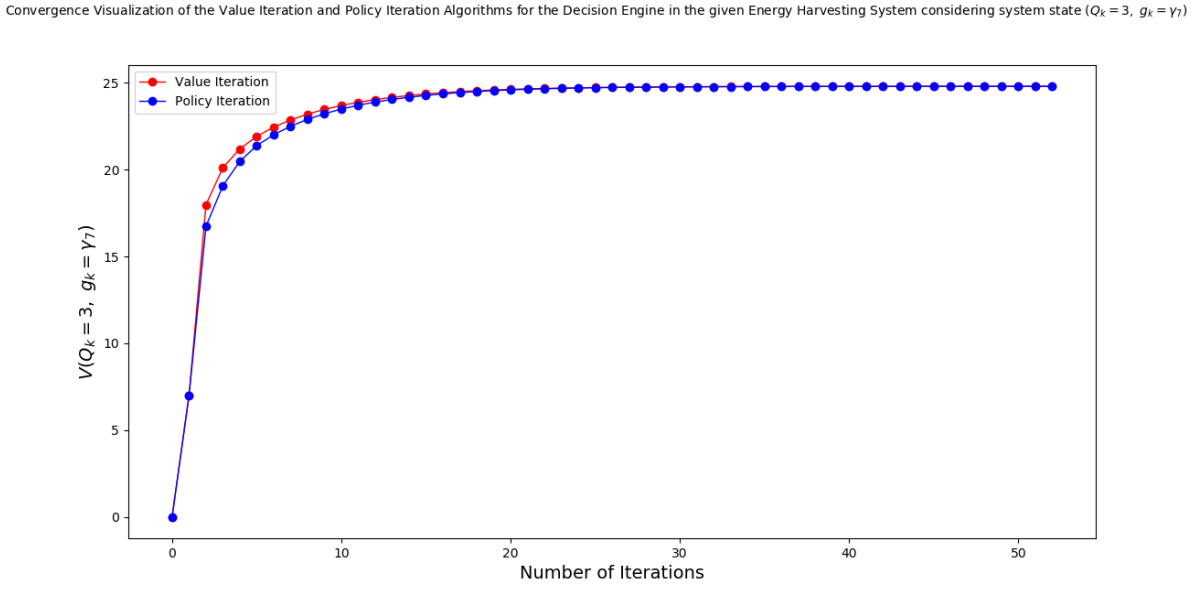

Fig. 4. Convergence Visualization of the Value Iteration and Policy Iteration Algorithms for the Decision Engine in the given Energy Harvesting System considering system state $(Q_k = 3,\ g_k = \gamma_7)$

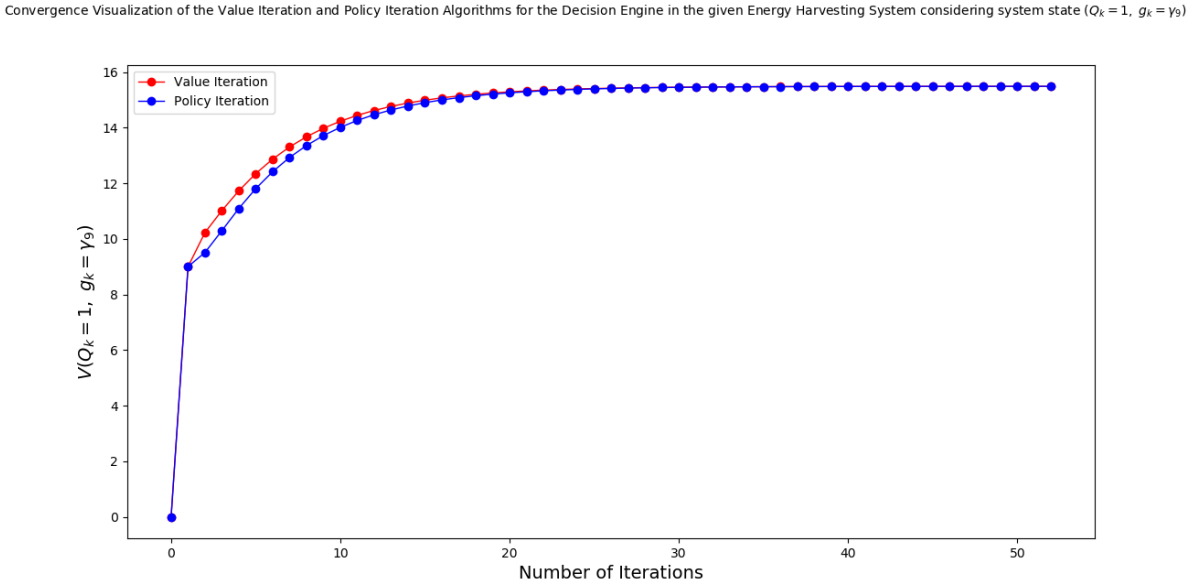

Fig. 5. Convergence Visualization of the Value Iteration and Policy Iteration Algorithms for the Decision Engine in the given Energy Harvesting System considering system state $(Q_k = 1,\ g_k = \gamma_9)$

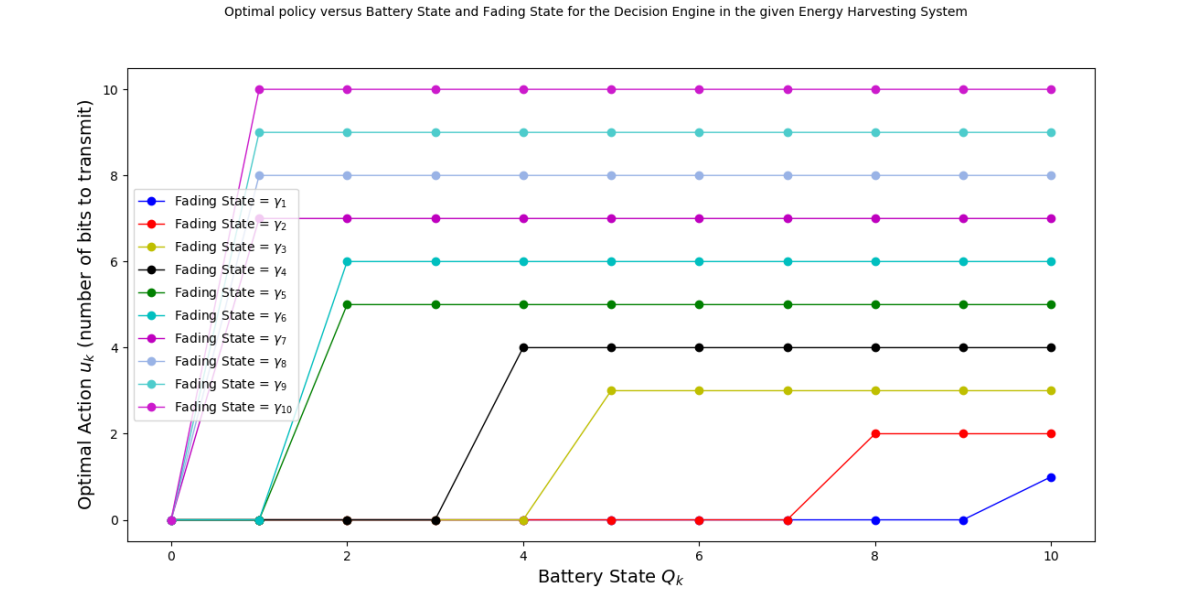

Fig. 6. Optimal policy versus Battery State and Fading State for the Decision Engine in the given Energy Harvesting System