Proceedings of the Pakistan Academy of Sciences:
A. Physical and Computational Sciences: 57 (2): 81-88 (2020)
Copyright © Pakistan Academy of Sciences
ISSN: 2518-4245 (print), 2518-4253 (online)

Pakistan Academy of Sciences

Research Article

Discrete Time Stochastic Root-finding with Forced Stopping Time

Ali Nasir[1]* and Huma Rehman Baig[2]

[1]Department of Electrical Engineering, University of Central Punjab, Lahore, Pakistan
[2]Directorate of Research, University of Central Punjab, Lahore, Pakistan

Abstract: In this paper, we present a Markov Decision Process (MDP) based formulation for solving the stochastic root-finding problem with predefined stopping time. In the problem that we pose, we need to find only one root of a given finite valued function ***f*** **(*p,u,h*)**. Here, p is a known Markov chain, u is the adjustable variable, and h is the unknown random variable with known distribution. Hence we cannot measure the true value of the function because h is unknown. We assume that we have a way of measuring whether or not ***f*** is within some bound ε from zero. We also present a formulation for the problem where ***f*** **(*p,u*)** is measurable but the adjustment of u is stochastic with known distribution. Another challenge in our problem is the introduction of finite stopping time. This means that the MDP policy has only a predefined finite number of actions available for adjusting u to find the root (or bring $|\boldsymbol{f}| < \boldsymbol{\varepsilon}$). We have included a price control example in the paper to demonstrate the behavior of the resulting MDP policy in response to the available time steps and the variable values of ***u*** and ***p***. The results show reasonable trajectories for our simulated environment.

Keywords: Markov decision processes, Stochastic root finding, Stochastic approximation.

1. INTRODUCTION

Stochastic root-finding is a well-researched problem in mathematics and engineering. The problem that we consider in this paper however is different from the standard stochastic root-finding problem in the literature. One of the main differences is the discrete and finite nature of the unknown function and consideration of known but arbitrary functional form. Another major difference is the inclusion of a finite time step limit for finding the root. Formally, the problem can be posed in two ways:

i. Given a discrete finite-value function $f\,(p,u,h)$ where p is known and changes at regular intervals according to a known Markov chain, u is adjustable and known, and h is unknown with a known distribution, find the control policy $\varphi\,(\,p\,,\,u\,)$ for u that can compensate changes in p and the unknown value of h such that the function value is driven to within a bound ε from 0 in less than or equal to n steps.

ii. Given a discrete finite-value function $f\,(p,h(u))$ where p is known and changes at regular intervals according to a known Markov chain, u is adjustable and known, and h is a random function of u with known conditional distribution given the values of u, find the control policy $\varphi(p,u)$ for u that can compensate changes in p and the value of h such that the function value is driven to within a bound ε from 0 in less than or equal to n steps.

This type of problem has abundant applications in planning under uncertainty and fault detection. For example, consider a problem of elevator assignment in a building where the number of passengers is bounded by the floor capacity but their appearance at each floor at any given time follows a known distribution. In this problem, we would like to bring the average waiting time for each passenger close enough to a pre-specified optimal time. Similar problems arise in the assignment of vehicles in the problems involving transportation, assignment of labor in the problems involving demand and production, selection of price, selection of thresholds in fault detection, selection of vehicle

Received: March 2019; Accpeted: June 2020
*Corresponding Author: Ali Nasir <a.nasir@ucp.edu.pk>

speed on the road (speed bounded by road speed limits), etc. The traditional stochastic root-finding problem has been studied extensively in the past. Some of the early contributions include the work of Robbins and Monro [1] where a non-parametric approach similar to the Newton-Raphson method is proposed. Wu has shown that the knowledge of the distribution of measurement noise can greatly improve the convergence rate [2]. Similarly, Yazidi [3] has proposed a novel approach for stochastic root-finding which is based on adaptive *d-ary* search. The solution shrinks the search by a factor of *2d/3*, where $d \geq 2$ is user-defined. Glimsdal [5] has proposed a Thompson sampling-based stochastic searching solution for deceptive environments. The proposed solution is also applicable to the root-finding problem. Similarly, Zhang [6] has presented a symmetrical, hierarchical stochastic searching on a 1-D line by operating a controlled random walk and obtaining information from the stochastic environment. Also, Vahidipour [4] has presented the shortest path in stochastic graphs using Learning Automata and Adaptive Stochastic Petri Nets (APSN-LA). The proposed solution has been validated on six different stochastic graphs and it has been reported as a relatively shortest path solution compared to other algorithms. Similarly, Pfeffer [7] a stochastic root-finding solution based on homotopy analysis which is applied to Schwinger equations. The author has reported a mathematical formulation that shows superior convergence properties compared to the bold diagrammatic Monte Carlo approach. A good review of the available solutions for the stochastic root-finding problem is found in [11].

In this work, we assume the distribution of the unknown part of the function is known. Our formulation is based on the Markov Decision Process (MDP) modeling. We present two modeling approaches. In our first approach, the function has some unknown element. In our second approach, the actions to change the function have uncertainty involved. The first approach corresponds to the problems with noisy sensors and perfect actuators. The second approach corresponds to the problems with perfect sensors and noisy actuators.

We have discussed the possibilities of deadlocks and live locks while using the formulation presented in this paper. These situations basically can arise due to a lack of history dependence in Markov process-based modeling. To avoid deadlocks and live locks, appropriate history has to be included in the state information. There is a class of problems though, which naturally exhibits Markov property and for those problems, our formulation works fine without any additional information in the state space. We have discussed the properties of conditional distribution that can ensure that the system obeys Markov property i.e. no history is required to find the root.

For an illustration of our proposed framework, we have also included a simulation-based example of the price control problem. The resulting optimal policy yields reasonable results and has some important characteristics that are briefly discussed. The paper is organized as follows. In Section 3 we present an appropriate background for MDPs. Our formulations are discussed in Section 4. Section 5 presents the simulation example and Section 6 includes conclusions and future work.

2. BACKGROUND ON MDP

An MDP is a controlled Markov chain that is solved using a discrete stochastic dynamic programming algorithm e.g. value iteration or policy iteration [8-10]. Value iterations are applied to the optimal control problem that maximizes an expected discounted reward function of the form.

$$V^{Pol}(s) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s^t, \mu^t) \mid Pol, s^0 = s\right] \qquad (1)$$

Here, s^t represents state after t actions, and μt is the action applied in state st according to a policy Pol (s^t is a random variable). V is the expected discounted reward function of states of the Markov chain (also called the value function of the state). The discount factor γ ($\gamma \in (0, 1)$), indicates that future rewards have a lower value. We assume that R is bounded from above and below. The policy that selects the optimal action may be found as:

$$Pol^*(s_i) \in \arg\max_k \left(R(\mu_k, s_i) + \gamma \sum_{j \in S} T(s_j \mid \mu_k, s_i) V(s_j) \right) \qquad (2)$$

There is a direct relationship between the value of a state and the values of all the states that can be reached from that state in a single optimal action. This relationship can be expressed using the

Bellman equation:

$$V_{t+1}(s_i) = R(s_i) + \max_k \left(\sum_{j \in S} \gamma T(s_j \mid \mu_k, s_i) V_t(s_j) \right) \quad (3)$$

where Vt+1(si) is the value of state si at iteration t+1. R(si) is the immediate reward of state si. T (sj |μk, si) is the probability of transitioning from state si to sj by executing action μk. Value iterations converge and one can bind the number of iterations (Itr) to reach an error bound of ε as:

$$Itr = \left\lceil \log\left(\frac{2R_{\max}}{(1-\gamma)\varepsilon} \right) / \log\left(\frac{1}{\gamma} \right) \right\rceil. \quad (4)$$

Here ε is the required tolerance of the solution satisfying,

$$\left\| V_{t+1}(\varsigma_i) - V(\varsigma_i) \right\| < \varepsilon, \forall i. \quad (5)$$

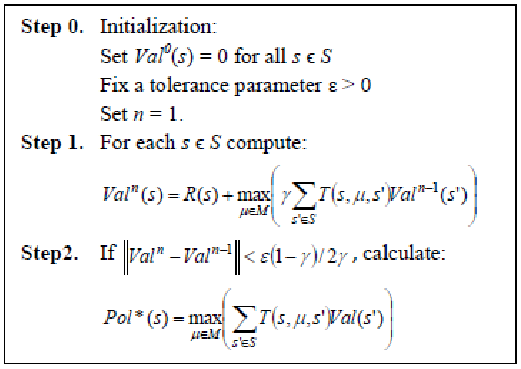
The inequality (5) is ensured by

$$\left\| V_{t+1}(\varsigma_i) - V_t(\varsigma_i) \right\| < \varepsilon \left(\frac{1-\gamma}{\gamma} \right). \quad (6)$$

The computational complexity of value iteration is of the order O(N2k), where, N is the number of states and k is the number of actions in the MDP. As described in the book by Kumar and Varaiya [8], Equation (3) converges to a unique solution. The solution of Equation (3) achieves its maximum value of the right-hand side in Equation (1). If the policy is calculated using (2) with a solution of (3), it will be optimal concerning (1). One of the algorithms to solve (3) and find an optimal policy from (2) is called value iteration; it is shown in Fig. 1.

3. STOCHASTIC ROOT FINDING FORMULATION

In this section, we formulate the MDP models for the two problems discussed in Section 1.



```
Step 0. Initialization:
        Set $Val^0(s) = 0$ for all $s \in S$
        Fix a tolerance parameter $\varepsilon > 0$
        Set $n = 1$.
Step 1. For each $s \in S$ compute:
        $Val^n(s) = R(s) + \max_{\mu \in M} \left( \gamma \sum_{s' \in S} T(s, \mu, s') Val^{n-1}(s') \right)$
Step2.  If $\left\| Val^n - Val^{n-1} \right\| < \varepsilon(1-\gamma)/2\gamma$, calculate:
        $Pol^*(s) = \max_{\mu \in M} \left( \sum_{s' \in S} T(s, \mu, s') Val(s') \right)$
```

Fig. 1. Value Iteration Algorithm

3.1 Known Action and Unknown Function

The objective of root finding in this subsection is to achieve $|f| < \varepsilon$ where f is a function of three variables, i.e., known but uncontrollable variable p, known and controllable variable u, and unknown and uncontrollable variable h. For stochastic root-finding formulation where the function f is not measurable (due to h) and actions of changing u have a deterministic effect on u, the states of the MDP are defined as:

$$\begin{aligned} &S = \{s_1, s_2, \ldots, s_N\} \\ &\textit{where,} \\ &s_i = \{p_i, u_i, B_i, q_i\} \\ &p_i \in P, u_i \in U, B_i \in \{0,1\}, q_i \in \{1, 2, \ldots, n\} \end{aligned} \quad (7)$$

where, B_i is the binary flag in state si indicating whether or not $|f| < \varepsilon$ ($B = 0$ when $|f| < \varepsilon$). Also qi indicates how many steps have passed since $|f| < \varepsilon$. All the states with $q_i = n$ are considered as failed states and are absorbing, i.e., no action is available from these states. P and U are finite sets of discrete values that variables p and u can take. Note that, we do not assume any particular property about the form of the function f; we just assume that we know that form.

The set of actions is defined as

$$M = \{\mu_{+1}, \mu_{-1}, \ldots, \mu_{+m}, \mu_{-m}, NOOP\} \quad (8)$$

where μ_{+i} increases u by the amount defined by increment step i. Similarly, μ_{-i} decreases u by the amount defined by decrement step i. There is also a *NOOP* action for the situations where we do not wish to change u.

The reward function for the problem is defined as:

$$R(s_i) = \begin{cases} e^{-B_i q_i} & if: q_i < n \\ 0 & otherwise \end{cases} \quad (9)$$

here, the reward is a negative exponential of a penalty function that increases with a decreasing number of allowable time steps (n - q).

The transition probabilities are computed from the distribution of h.

$$T(s_i, \mu_k, s_j) = \begin{cases} P\left(\delta_1(p_j, u_j) < h < \delta_2(p_j, u_j)\right) & for: B_j = 0, q_j = 1 \\ 1 - P\left(\delta_1(p_j, u_j) < h < \delta_2(p_j, u_j)\right) & for: B_j = 1, q_j = q_i + 1 \end{cases} \quad (10)$$

$T(s_i, \mu_k, s_j)$ represents the transition function from state s_i to state s_j when action μ_k is applied. Here, δ_1 and δ_2 are computed from the functional form of f(p,u,h) to find the range of h such that $|f| < \varepsilon$. Also, the above equation is for the states s_i other than the failure states, i.e., $q_i < n$ in Equation (10). Finally, the values of p and u in the next state s_j in Equation (10) are such that $p_j = p_i$ and $u_j = u_i + u_k$.

With the above formulation, we can use values iteration to find the optimal policy for adjusting u. next we discuss a slightly different formulation for a similar problem with a different point of view.

3.2 Known Function and Unknown Action

The objective of root finding in this subsection is to achieve $|f| < \varepsilon$ where f is a function of two variables, i.e., known but uncontrollable variable p and controllable but unknown variable u, here u is assumed to be a function of h where h is unknown except for its probability distribution. For the case where the function is computable but the effects of changing the control variable are stochastic with a known distribution, we assume f is of the form $f(p,u)$. The states can now be defined as:

$$\begin{aligned} &S = \{s_1, s_2, ..., s_N\} \\ &where, \\ &s_i = \{f_i, u_i, q_i\} \\ &f_i \in F, u_i \in U, q_i \in \{1, 2, ..., n\} \end{aligned} \tag{11}$$

here, F is a finite set for values of f and all other symbols have the same meaning as before. The set of actions remain the same. The reward function is technically the same but has a slightly different form:

$$R(s_i) = \begin{cases} e^{-I(f_i \geq \varepsilon) q_i} & if: q_i < n \\ 0 & otherwise \end{cases} \tag{12}$$

here, we have used the indicator function $I(x)$ which is 1 when x is true and 0 otherwise. The transition function will now depend upon the distribution of the effect of actions on variable u.

$$\begin{aligned} &T(s_i, \mu_k, s_j) = P(u_+ = u_j \mid u = u_i, \mu = \mu_k) \\ &q_j = \begin{cases} 1 & if: |f_j| < \varepsilon \\ q_i + 1 & otherwise \end{cases} \end{aligned} \tag{13}$$

here, the distribution of u is assumed to be Markov and known. Similar to Equation (10), Equation (13) assumes $q_i < n$. since the variable p does not change during the execution of the action, f_j is computed from f_i, u_i, and u_j.

3.3 Deadlocks and Live Locks

So far in the above formulations, we have not assumed any specific relation between f and u. All we know is that f depends upon u. This can lead to deadlocks and live locks. By deadlocks we mean the situations where the policy is stuck with one action and that action is not working (i.e. not ensuring $|f| < \varepsilon$). By live lock, we mean the situation where the policy is oscillating between two actions (or more actions) that are not working. The example of a deadlock is as follows. Suppose that $f = \sin(p) + \cos(u) + h$, $\varepsilon = 0.01$, $p = 0$, $h = 1$, and $u = \pi/2$. Also, suppose that h is uniformly distributed between -1 and 1. According to the distribution of h, the best action computed by the MDP could be *NOOP* (since h is unknown for the MDP). Hence, the MDP policy will keep on suggesting *NOOP* and the root will never be found. If the *NOOP* is defined in such a way that it cannot change f, even, in that case, the policy may only fluctuate between two best values of u closest to $\pi/2$. Similarly, for the second formulation, if $f = \sin(p) + \cos(u)$, $\varepsilon = 0.01$, $p = 0$, $u = \pi/2 - 0.1$, and distribution of change in u conditioned upon applied actions is such that a particular action μ has the highest probability of increasing μ by 0.1, the MDP policy, in this case, could keep on applying the action μ and in reality, it might never change μ at all. Livelock situations could also arise in both formulations. To avoid these unwanted situations, MDP needs to have a record of past actions so that the actions that do not work may not be tried again. This will increase the state space depending upon how many values μ can assume. For example, if μ can take on 10 possible values and we incorporate binary flags for indicating which of the 10 values have been used in the past, then the size of the state space will grow by a factor of 2^{10}. We can also incorporate copies of μ instead of binary flags for each of the finite number of past actions. For example, if we want to incorporate the history of the last 3 actions, then we need 3 copies of variable u in the state space which will increase the state space by a factor of 10^3. In either case, the reward function and the transition function would have to be adjusted accordingly to make use of the additional available information in calculating the policy.

There is a class of problems for which the deadlocks and live locks do not occur. The problems of this class exhibit a strictly monotonic relation in distribution between *f* and *u*. For example, if the conditional distribution of the change in *f* given change in *u* is such that f changes only directly or only inversely with *u* and *f* does not change if *u* does not change, then we have a monotonic relation in distribution between *f* and *u*. Additional information required here is the sign of *f*. So the flag *B* in the earlier formulation should have three possible values, indicating $f > \varepsilon$, $f < -\varepsilon$, and $|f| < \varepsilon$. The second formulation where we already know *f*, the strictly monotonic relation in conditional distribution should exist between *u* and the action to change *u* which is the same as the strictly monotonic conditional distribution between change in *f* and deterministic action to change *u* in the earlier formulation. This is because in both cases, we effectively try to change f, but the effect of our action follows a known distribution. In this sense, both formulations represent the same problem is conditional independence. Fortunately, we have a way to determine conditional independence and that is the Bayesian network. The Bayesian network for a single target-neighbor pair is shown in Fig. 2. There are a few things to note here. First, a target status is random only when it is submitted to a neighboring spacecraft for exploration. Otherwise, an idle target remains idle unless submitted for and an explored target remains explored indefinitely. Second, the target status depends only on the status of the neighboring spacecraft to which it is submitted and it is independent of the status of all other neighboring spacecraft. Third, the status of each neighbor is independent of the status of other neighbors. Under these considerations, there is one Bayesian network for each target-neighbor pair.

4. SIMULATION EXAMPLE

In this section, we present a simulation example for a price control problem where $f(p,h(u))$ is the difference between production p and demand *h*. Our control variable is price *u* that is assumed to have indirect but strictly monotonic stochastic relation with the demand, i.e., the demand increases when the price is reduced and vice versa but the increase or decrease in demand follows a known distribution. Also, if the price does not change, the demand does not change with probability 1. The state space is represented as:

$$\begin{aligned} &S = \{s_1, s_2, \ldots, s_N\} \\ &where, \\ &s_i = \{f_i, u_i, q_i\} \\ &f_i \in [-10:10], u_i \in [1:0.1:5], q_i \in [1:6] \end{aligned} \tag{14}$$

Note that the value of f ranges between -10 and 10 in the above equation. The value of the controllable variable *u* ranges from 1 to 5 with the smallest possible increment of 0.1, i.e., *u* can assume values such as 1, 1.1, 1.2, and up to so on 5. Finally, the number of steps (q) ranges from 1 to 6. The action space is defined as:

$$M = \left\{ \begin{matrix} \mu_{+0.1}, \mu_{-0.1}, \mu_{+0.5}, \mu_{-0.5}, \mu_{+1}, \mu_{-1}, \mu_{+2}, \mu_{-2}, \mu_{+3}, \mu_{-3}, \\ \mu_{+4}, \mu_{-4}, NOOP \end{matrix} \right\} \tag{15}$$

Equation (15) presents 13 possible actions where the smallest possible nonzero change in *u* is 0.1 and the largest possible change in *u* is 4. Note that these actions are context-dependent, i.e., if *u* is already equal to 5 in the current state, then any action requiring a positive change in *u* shall result in *NOOP*. Similarly, reduction in u is not allowed when *u* is at its minimum value. The reward function is represented by Equation (12) with *n = 6*. To avoid big transitions in *u*, we also used a cost function where the cost of changing the price was proportional to the magnitude of the change.

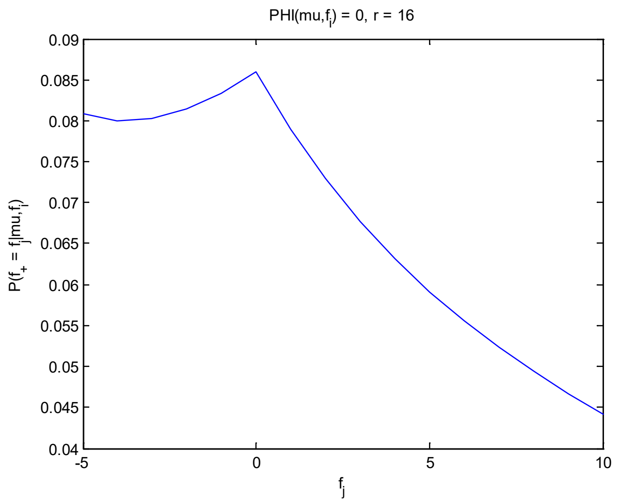


Fig 2. Conditional Distribution of Change in *f* given Initial value of *f* and Change in *u*.

The transition function is given as:

$$
\begin{aligned}
&T\left(s_i,\mu_k,s_j\right)=P(f_+=f_j \mid f=f_i,\mu_k=u_j-u_i)\\
&q_j=\begin{cases}1 & if:\left|f_j\right|<\varepsilon\\ q_i+1 & otherwise\end{cases}\\
&P(f_+=f_j \mid f=f_i,\mu_k=u_j-u_i)=\frac{r-\left|f_j-\Phi\left(f_i,\mu_k\right)\right|}{NC}\\
&r=\begin{cases}f_i-f_{\min}+1 & if:\mu_k<0\\ f_{\max}-f_i+1 & if:\mu_k>0\end{cases}\\
&\Phi\left(f_i,\mu\right)=\begin{cases}f_i+2\mu_k & if:\left|\mu_k\right|=0.5, f_i+2\mu_k\in\left[f_{\min},f_{\max}\right]\\ f_i+\mu_k+sign\left(\mu_k\right) & if:\left|\mu_k\right|\in\{1,2,3,4\}, f_i+\mu_k+sign\left(\mu_k\right)\in\left[f_{\min},f_{\max}\right]\\ f_i & otherwise\end{cases}\\
&NC=\Phi\left(f_i,\mu_k\right)\left(1+r-\Phi\left(f_i,\mu_k\right)\right)+\frac{r(r-1)}{2}
\end{aligned}
\tag{16}
$$

In Equation (16), r is the range of values f_j can take given f_i and μ_k. Φ is the function that determines the most likely value for the result of applying μ_k from f_i. NC is the normalization constant. For elaboration purposes, the distribution curve for $r = 16$, $f_i = -5$, $\mu_k = 4$, and $\Phi = 0$ is shown as a function of possible values of f_j in Fig 2. Note that, since $\mu_k > 0$, f can only increase. The results in Fig 2 indicate that the value of μ is large whenever the value of f is large. Note that μ is negative for positive values of f and positive for the negative values of f.

We calculated the optimal policy for the above example and the results are shown in Fig. 3 and Fig 4. In Fig 3, the optimal policy is to increase the price whenever demand exceeds production (i.e. $f < 0$) and decrease the price in the opposite situation. The change in price depends upon the magnitude of f and available price change such that the price remains in the range of 1 to 5. In Fig 4, the policy is plotted for $u = 1.5$ which means the decrease in price is not available. Note that in Fig. 4, the change in price not only depends upon f but is also dependent upon q. As can be seen in Fig. 4, for larger q the change in price is more aggressive for the same value of f.

Finally, we include an example of a trajectory of f generated using our optimal policy in a random environment where p jumps randomly after every 7-time steps. Fig 5 shows the results. Note that, our policy was able to bring f within the allowable range i.e., [-1 1] in maximum 5-time steps (without failing).

We have also presented the statistics of the results of Fig 5 in Table 1. Table 1 indicates that the number of steps to bring the function f within the threshold range from 1 to 5 with an average value of 2.58. The standard deviation of 1.43 indicates that the function f is brought within the threshold in four steps for most cases.

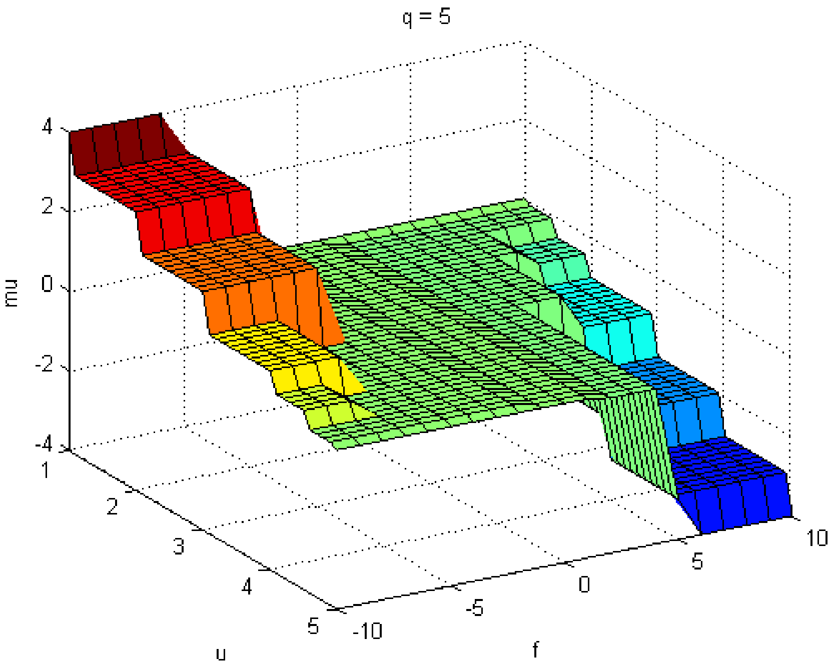


Fig 3. Optimal Policy for $q = 5$

Table 1. Statistics Regarding Number of Steps

Statistical Measure	**Number of Steps**
Mean	2.58
Min	1
Max	5
Mode	2
Standard Deviation	1.43

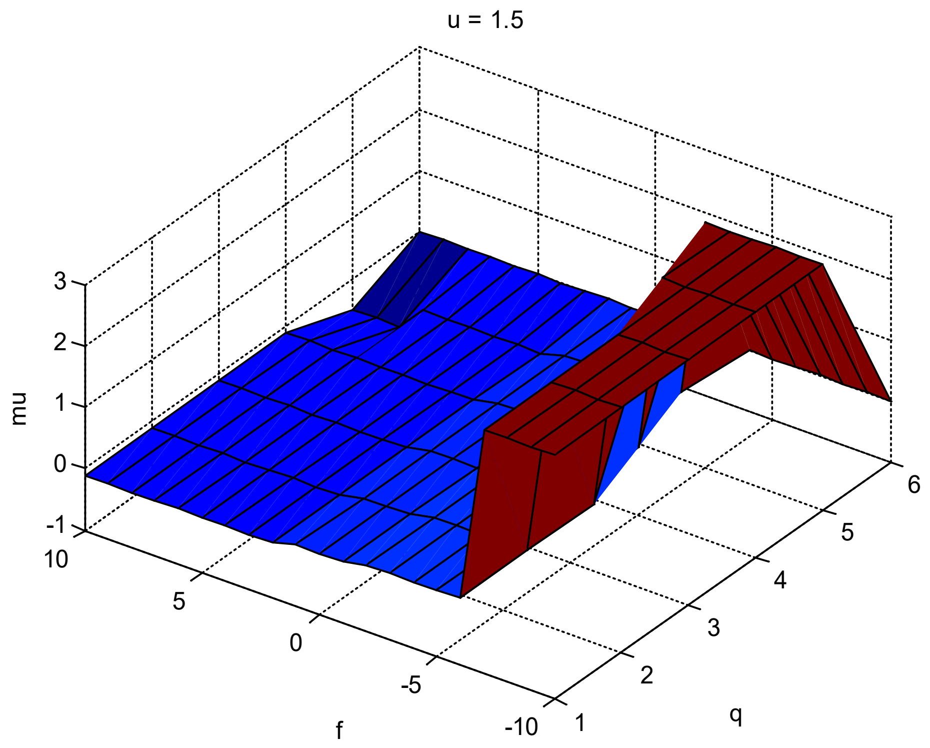


Fig 4. Optimal Policy for u = 1.5

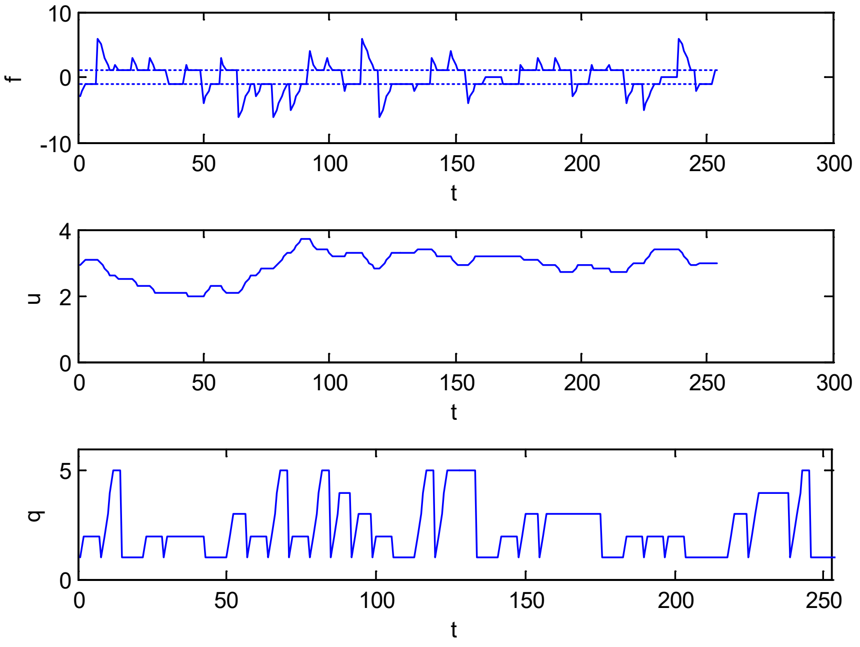


Fig 5. The trajectory of f in a Random Environment

5. CONCLUSIONS AND FUTURE WORK

In this paper, we have shown that by using MDP formulations, it is possible to calculate optimal policies for the stochastic root-finding problem with a finite number of allowable steps. The root-finding problem discussed in this paper is discrete where the function can take on only a finite number of values. The problem of deadlocks and live locks for a general class of problems has been discussed and possible ways of avoiding such situations along with the consequences have been mentioned. Also, the specific class of problems for which the deadlocks and live locks do not occur has been identified. The behavior of optimal policy for a simulation example has been included to show how the policy responds to the available time steps as well as the available change in the controlled variable. We have also included the trajectory of f using our policy. The results show good performance in terms of keeping the function within its allowable range for our simulated environment.

5. REFERENCES

1. H. Robbins, and S. Monro. A stochastic approximation method. *Annals of Mathematics and Statistics* 29, 373-405 (1951).
2. C.F.J. Wu. Efficient sequential designs with binary data. *Journal of American Statistical Association* 80 974-984 (1985).
3. A. Yazidi, and B.J. Oommen. A novel technique for stochastic root-finding: Enhancing the search with the adaptive d-ary search. *Information Sciences,* 393: 108-129 (2017).
4. S.M.Vahidipour., M.R. Meybodi, and M. Esnaashari. Finding the shortest path in stochastic graphs using learning automata and adaptive stochastic petri nets. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 25(03) 427-455 (2017).
5. S. Glimsdal, and O.C. Granmo, Thompson Sampling Guided Stochastic Searching on the Line for Deceptive Environments with Applications to Root-Finding Problems. *arXiv preprint arXiv*:1708.01791 (2017).
6. J. Zhang., Y. Wang., C. Wang, and M. Zhou. Symmetrical hierarchical stochastic searching on the line in informative and deceptive environments. *IEEE transactions on cybernetics*, 47(3) 626-635 (2017).
7. T. Pfeffer, and L. Pollet. A stochastic root finding approach: the homotopy analysis method applied to Dyson–Schwinger equations. *New Journal of Physics*. 19(4): (2017).
8. P.R. Kumar, and P. Varaiya, *Stochastic Systems: Estimation, Identification, and Adaptive Control,* Prentice Hall Inc., Englewood Cliffs, New Jersey 07632, 1986.
9. M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming,* © John Wiley and Sons Inc. (1994).
10. S. Russell, and P. Norvig. *Artificial Intelligence: A Modern Approach,* 2nd Edition, Prentice-Hall, Upper Saddle River, New Jersey 07458, (2005).
11. P. Raghu, and S. Kim. The stochastic root-finding problem: Overview, solutions, and open questions. *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 21(3) 1-23 (2011).