Machine Learning Techniques Applied to Robust Optimal Control Problems

Dilzhan Zhangunissov
Supervisor: Kerem Ugurlu

April 24, 2024

Abstract

This project aims to solve the discrete time stochastic optimal control problem of evaluation of Average Value-at-Risk (AVaR) function. AVaR is an important tool in market risk management used to measure the risk. In the paper it was designed as a sequential decision model and solved by formulating an optimal control problem of minimizing the value. Brute force and Approximate Dynamic Programming (ADP) techniques were used for exact and approximate solutions respectively. Golden section search was used to solve the problem completely. The numerical experiments conducted at the end showed the effectiveness of the algorithm in evaluating the AVaR.

Keywords: approximate dynamic programming; average value-at-risk; optimal control; Markov decision processes

1 Introduction

Average Value-at-Risk (AVaR) is a significant measure in market risk management that is used to assess potential losses in the worst-case scenarios. This project aims to evaluate the AVaR by formulating it as a discrete time stochastic optimal control problem. The main goal of the work is to minimize the total cost of decisions made over a finite time horizon. Optimal control theory deals with sequential decision models, where decisions made at each time step incur costs and may affect future decisions. The project investigates both exact and approximate solutions by using brute force and Approximate Dynamic Programming (ADP) techniques, respectively. By analyzing the stability of the ADP solution alongside its accuracy with respect to the brute force solution the paper verifies the reliability of the ADP solution. Furthermore, the golden section search technique was used to solve the outer minimization in order to evaluate the AVaR function completely.

2 Theory

Average Value-at-Risk

Average Value-at-Risk(AVaR) (also known as Expected Shortfall, Conditional Value-at-Risk, etc.) is a risk measurement tool widely used in market risk management. Acerbi and Tasche (2002) define AVaR at a level α as "the literal mathematical transcription of the concept "average loss in the worst $100\times\alpha\%$ cases"".

Definition 2.1. Let $X \in L^1(\Omega, \mathcal{F}, \mathbb{P})$ be a real valued random variable defining the future profit of an arbitrary portfolio and let $\alpha \in (0,1)$ (Bäuerle & Ott, 2011)
Then:

a) The Value-at-Risk of the portfolio X at level α is defined by:

$$\mathrm{VaR}_\alpha(X) = \inf\{x \in \mathbb{R} : \mathbb{P}(X \le x) \ge \alpha\}$$

b) The Average Value-at-Risk of the portfolio X at level α is defined by:

$$\mathrm{AVaR}_\alpha(X) = \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_t(X)\,dt$$

Rockafellar and Uryasev (2002) formulate AVaR as:

$$\mathrm{AVaR}_\alpha(X) = \min_{s\in\mathbb{R}}\left\{s + \frac{1}{1-\alpha}\mathbb{E}\left[(X-s)_+\right]\right\} \tag{1}$$

and claim that $s^* \triangleq \arg\min_{s\in\mathbb{R}} \left\{ s + \frac{1}{1-\alpha}\mathbb{E}\left[(X-s)_+\right] \right\} = \mathrm{VaR}_\alpha(X)$. Here, $(\cdot)_+$ denotes the ramp function, namely, $(x)_+ = \begin{cases} x, & \text{if } x > 0 \\ 0, & \text{otherwise} \end{cases}$.

Optimal Control

Optimal control deals with sequential decision models, where the decision made incurs some cost and may affect the future decisions. The problem is to optimize the total cost of the decisions, it can be either maximize the profit or minimize the risk (Bertsekas & Shreve, 1996).

Bertsekas and Shreve (1996) in their work give a classical example of the optimal control problem: the portfolio management, where the objective of the investor is to maximize the profit.

Optimal control problems may be classified into deterministic and stochastic. In deterministic optimal control, the state obtained under a decision is deterministic, i.e. there is no randomness. In stochastic optimal control, the next state is also affected by some random disturbance (Bertsekas & Shreve, 1996).

The other important classification of the optimal control problems is the finiteness of the horizon. In finite horizon problems the sequential decision model is supposed to stop at the specified point in time. Whereas infinite horizon problems deal with perpetual decision making. In such problems either contraction or the monotonicity of the value function over time are assumed (Bertsekas & Shreve, 1996).

This work focuses on finite horizon discrete time stochastic optimal control problem. The objective is to minimize the total cost of the made decisions over time T given the set of the decisions could be made $\mathcal{A}$ with the fixed initial state x_0, namely, evaluate the:

$$V(0, x_0) = \min_{\pi} \mathbb{E}\left[\sum_{t=0}^{T} c(x_t, a_t)\right] \tag{2}$$

(Bertsekas, 2017).

Markov Decision Processes

Markov Chain is a random model that describes a sequence of events (states) with some random transitions between them (Gagniuc, 2017; Yang, 2020).

Definition 2.2. A sequence of random states is called Markovian if it satisfies the Markov Property of independence of future states on past states, namely:

$$\mathbb{P}\left[X_n = x_n | X_0 = x_0, X_1 = x_1, \dots X_{n-1} = x_{n-1}\right] = \mathbb{P}\left[X_n = x_n | X_{n-1} = x_{n-1}\right]$$

Markov Decision Process (MDP) is an extension of Markov Chains with the addition of control (Bäuerle & Rieder, 2010). The next state in the MDP is both controlled and random, under the assumption of Markov Property. This work specializes on Discrete Time Finite Horizon Markov Decision Process. At each time step the decision maker chooses an action from action set $\mathcal{A}$. After the action is chosen, the random transition occurs to define the next state (the state at the next time step). For each taken action the decision maker receives some reward (or pays some cost), which they will try to optimize (Bäuerle & Rieder, 2010).

According to Bäuerle and Rieder (2010), the Discrete Time Markov Decision Process $\mathcal{M}$ with finite horizon $N \in \mathbb{N}$ consists of data $(S, \mathcal{A}, D_n, P_n, r_n, g_N)$, where for $n = 0, 1, 2, \dots, N-1$:

- S denotes the state space.
- $\mathcal{A}$ denotes the action space.
- $D_n \subset S \times \mathcal{A}$ denotes the subset of admissible state-action pair meaning all the combinations of states and actions allowed at time n.
- P_n is a Markov kernel from D_n to S meaning the transition probabilities from some state under some action.
- $r_n : D_n \to \mathbb{R}$ denotes the reward taken by a decision maker given a state and action at a time step n.
- $g_N : S \to \mathbb{R}$ is a terminal reward of the system depending on the final state of the system.

This paper supposes that $D_n = S \times \mathcal{A}$ for all $n = 0, 1, 2, \dots, N-1$ and $g_N = \sum_{n=0}^{N-1} r_n$. From now, the MDP will be formulated as a tuple of four parameters $(S, \mathcal{A}, P_n, r_n)$.

Problem Statement

The project aims to solve the problem of evaluating AVaR function by choosing an optimal policy $\pi \in \Pi = \mathcal{A}^{T+1}$, namely:

$$\begin{aligned} \text{AVaR}_\alpha\left(\sum_{t=0}^{T} c(x_t, a_t)\right) = \min_{s\in\mathbb{R}} \min_{\pi} \quad & \left\{ s + \tfrac{1}{1-\alpha}\mathbb{E}\left[\left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right]\right\} \\ \text{subject to:} \quad & x_{t+1} = x_t + a_t + \xi_t \\ & x_0 \text{ is fixed} \end{aligned} \tag{3}$$

The cost function $c(\cdot,\cdot)$ was chosen to be defined as $c(x_t, a_t) = x_t^2 + a_t^2$ and ξ_t here is randomness that was defined as $\xi_t = \begin{cases} -1 & \text{,with probability } \frac{1}{2} \\ 1 & \text{,with probability } \frac{1}{2} \end{cases}$.

Note that the problem can not be considered as a classical Markov Decision Problem since AVaR is a convex measure (Bäuerle & Ott, 2011).

3 Inner Minimization

Firstly, let us focus on solving the minimization over a policy. To do so, we will define a function

$$\begin{aligned} f(s, \alpha, x_0) \quad &= \min_{\pi}\left\{ s + \frac{1}{1-\alpha}\mathbb{E}\left[\left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right]\right\} \\ &= s + \tfrac{1}{1-\alpha}\min_{\pi}\left\{\mathbb{E}\left[\left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right]\right\} \end{aligned} \tag{4}$$

Exact Solution

For fixed values of s, α, x_0 the value $f(s, \alpha, x_0)$ can be evaluated exactly using a brute force technique by going through all the possible policies $\pi \in \Pi$ and randomness sequences $\xi \in \{-1, 1\}^T$.

For each picked policy $\pi = \{a_0, a_1, \ldots, a_T\}$ and randomness sequence $\xi = \{\xi_0, \xi_1, \ldots, \xi_{T-1}\}$ we calculate the total cost of the applied actions, subtract s from the obtained sum and apply the ramp function. Since ξ_i's are independently and identically distributed, we claim that the probability of randomness ξ being occurred is 2^T. To count the expectation we use the definition of the expectation:

$$\mathbb{E}\left[\left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right] = \sum_{\pi\in\Pi}\left[\frac{1}{2^T} \times \left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right]$$

The time complexity of the following algorithm is $O\left(|\mathcal{A}|^{T+1} \times 2^T \times T\right)$.

Algorithm 1 Brute Force Algorithm

```
Generate Π = A^{T+1}.
Generate Ξ = {−1, 1}^T.
min ← ∞
for π ∈ Π do
    V ← 0
    for ξ ∈ Ξ do
        current_cost ← 0
        for t = 0 to T − 1 do
            current_cost = current_cost + c(x_t, a_t)
            x_{t+1} ← x_t + a_t + ξ_t
        end for
        current_cost = current_cost + c(x_T, a_T)
        current_cost = max(current_cost − s, 0)
        V = V + current_cost/2^T
    end for
    if V < min then
        min ← V
    end if
end for
return f ← s + 1/(1−α) min
```

Approximate Solution

The problem can be solved approximately with the help of Approximate Dynamic Programming(ADP). For that purpose the sum at (4) can be split as

$$f(s,\alpha,x_0) = s + \frac{1}{1-\alpha}\min_{\pi}\left\{\mathbb{E}\left[\left(\sum_{t=\tilde{t}+1}^{T} c(x_t,a_t) + \sum_{t=0}^{\tilde{t}} c(x_t,a_t) - s\right)_+\right]\right\} \tag{5}$$

so that the additional deterministic state $s_{\tilde{t}} = \sum_{t=0}^{\tilde{t}} c(x_t,a_t) - s$ can be defined.

Now we can formulate a Markov Decision Process $\mathcal{M} = (S,\mathcal{A},P,c)$ to run the ADP on, here:

- S is the state space. Each state is in form $(x_t, s_{t-1}) \in \mathbb{R}^2$.
- $\mathcal{A}$ is the action space.
- P is the time-independent Markov kernel. For an action $a \in \mathcal{A}$ and state $(x_t, s_{t-1}) \in S$,

$$(t+1, x_{t+1}, s_t) = \begin{cases} (t+1, x_t + a + 1, s_{t-1} + c(x_t,a)) & \text{, with probability } \frac{1}{2} \\ (t+1, x_t + a - 1, s_{t-1} + c(x_t,a)) & \text{, with probability } \frac{1}{2} \end{cases}$$

- c is a cost function defined earlier, $c(x_t,a) = x_t^2 + a^2$ for an action $a \in \mathcal{A}$.

Note that the kernel and the reward (in our case cost) functions now given time-independent.

Now, let us consider the Dynamic Programming steps. At time $t = T$, there is no randomness occurring, so the decision maker needs to pick the optimal action without thinking about the expectation of the value function at next time step. Starting from time $t = T-1$ and to $t = 0$, the decision maker needs to consider the expectation of the value function at next time step with respect to the random variable $X_{t+1} = x_t + a_t + \xi_t$.

So, the Dynamic Programming steps are as follows:

$$\begin{aligned}
&t = T: \\
&\qquad V(T, x_T, s_{T-1}) &&= \min_{a_T}\left\{(c(x_T,a_T) + s_{T-1})_+\right\} \\
&t = T-1: \\
&\qquad V(T-1, x_{T-1}, s_{T-2}) &&= \min_{a_{T-1}}\left\{\mathbb{E}\left[(V(T, X_T, s_{T-1}) + c(x_{T-1},a_{T-1}))_+\right]\right\} \\
&&&= \min_{a_{T-1}}\left\{\mathbb{E}\left[(V(T, x_{T-1} + a_{T-1} + \xi_{T-1}, s_{T-1}) + c(x_{T-1},a_{T-1}))_+\right]\right\} \\
&t < T-1: \\
&\qquad V(t, x_t, s_{t-1}) &&= \min_{a_t}\left\{\mathbb{E}\left[(V(t+1, X_{t+1}, s_t) + c(x_t,a_t))_+\right]\right\} \\
&t = 0: \\
&\qquad V(0, x_0, s_{-1}) &&= \min_{a_0}\mathbb{E}\left\{\left[(V(1, X_1, s_0) + c(x_0,a_0))_+\right]\right\}
\end{aligned}$$

To approximate the value function, we will generate a random path starting from $t = 0$ to $t = T$ and evaluate the total cost for this particular path.

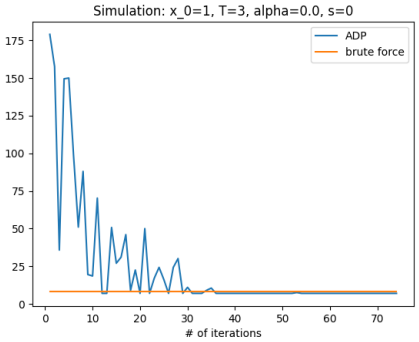
Algorithm 2 Approximate Dynamic Programming

function EXPECTATION(V, x_t, s_{t-1}, a, t, T)
  $val \leftarrow 0$
  for $\xi \in \{-1, 1\}$ **do**
    $x_{t+1} \leftarrow x_t + a + \xi$
    $s_t \leftarrow s_{t-1} + c(x_t, a)$
    if $V(t+1, x_{t+1}, s_t) = \infty$ **then**
      $V(t+1, x_{t+1}, s_t) \leftarrow I$ ▷ Some value that is close to the true value that would be explored
    end if
    $val \leftarrow val + V(t+1, x_{t+1}, s_t)/2$
  end for
  return val
end function
function SIMULATE(x_0, s_{-1}, T, N)
  $counter \leftarrow 0$
  $V \leftarrow \varnothing$ ▷ the value function
  while $counter < N$ **do**
    for $t = 0$ to $T - 1$ **do**
      Pick random $a \in \mathcal{A}$
      Pick random $\xi \in \{-1, 1\}$
      $path[t] = (t, x_t, s_{t-1}, a)$
      $x_{t+1} \leftarrow x_t + a + \xi$
      $s_t \leftarrow s_{t-1} + c(x_t, a)$
    end for
    $a_T \leftarrow \arg\min_{a \in \mathcal{A}} \{(c(x_T, a_T) + s_{T-1})_+\}$
    $V(T, x_T, s_{T-1}) \leftarrow (c(x_T, a_T) + s_{T-1})_+$
    for $t = T - 1$ to 0 **do**
      $a_t = \arg\min_{a_t \in \mathcal{A}} \{\text{EXPECTATION}(V, x_t, s_{t-1}, a_t, t, T)\}$
      $V(t, x_t, s_{t-1}) = (c(x_t, a_t) + s_{t-2})_+$
    end for
    $counter \leftarrow counter + 1$
  end while
  return V
end function

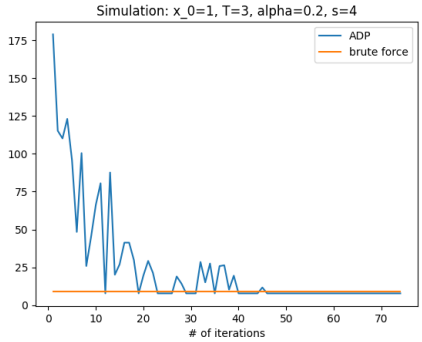
The time complexity of the described algorithm is $O(T \times N \times |\mathcal{A}|)$.

4 Convergence

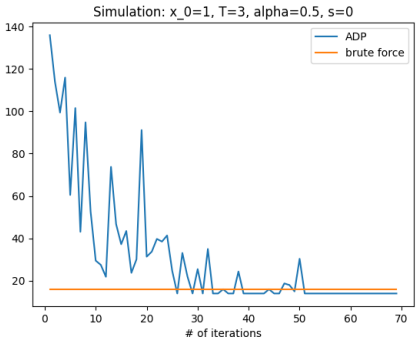
In this section, we show how the suggested ADP solution progresses over the number of iterations used in Monte-Carlo Simulation using graphs. The convergence of the solution tells us that the algorithm is stable. The accuracy of the algorithm may be checked by comparing the answers obtained using ADP with the answers obtained by brute force solution. Starting from now, we fix the action set $\mathcal{A}$ as $\{-1, 0, 1\}$. Firstly, let us check the accuracy of the ADP algorithm on smaller value of T.



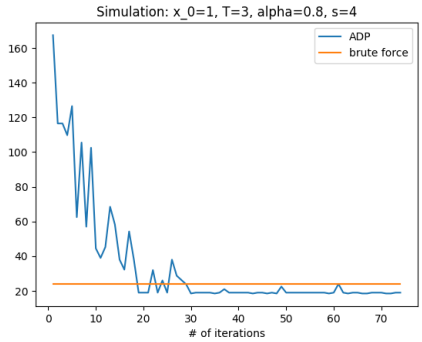
(a) $\alpha = 0, s = 0$



(b) $\alpha = 0.2, s = 4$



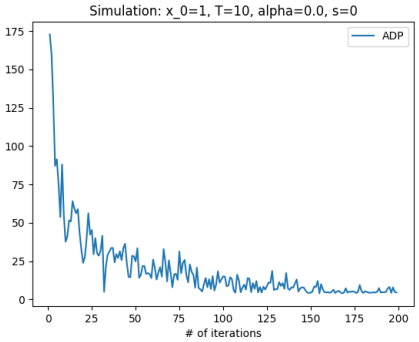
(c) $\alpha = 0.5, s = 0$



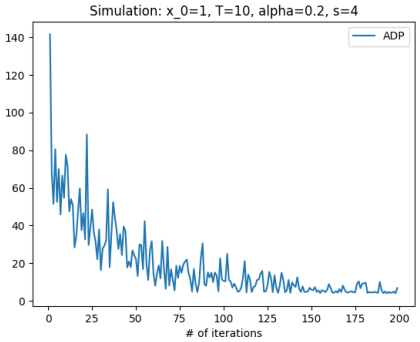
(d) $\alpha = 0.8, s = 4$

Figure 1: Convergence of the ADP answer for $T = 3, x_0 = 1$

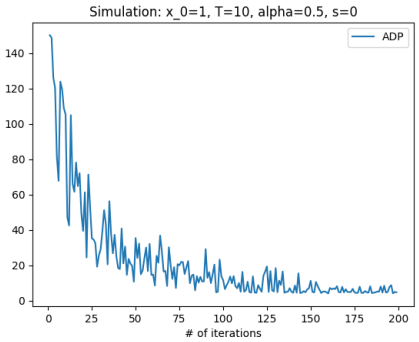
As could be seen, the ADP solution converges almost to a brute force solution through the number of iterations. This shows that the ADP solves the problem with a high accuracy. Now, the stability of the ADP solution can be verified by testing with greater values of T. For this time, we will not compare the answer with the answer obtained by using the brute force solution since it is very costly in terms of time to run it.



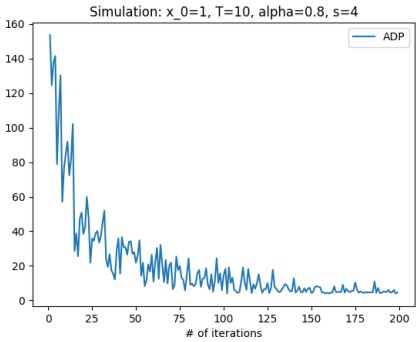
(a) $\alpha = 0, s = 0$



(b) $\alpha = 0.2, s = 4$



(c) $\alpha = 0.5, s = 0$



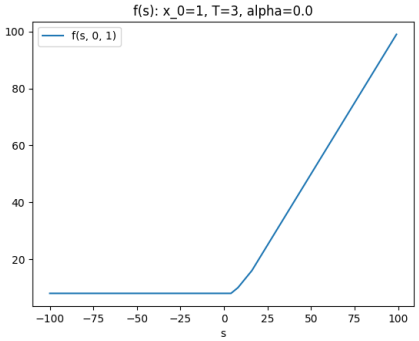
(d) $\alpha = 0.8, s = 4$

Figure 2: Convergence of the ADP answer for $T = 10, x_0 = 1$

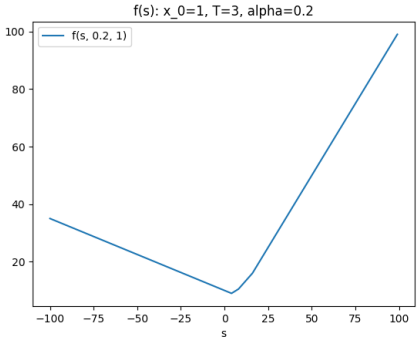
The convergence of the solution tell us that the ADP algorithm is stable, concluding that the solution is both accurate and stable, thus, reliable.

5 Outer Minimization

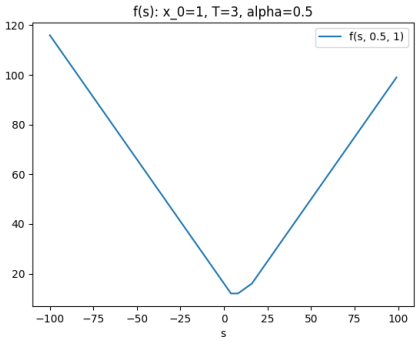
Now, when we can compute $f(s, \alpha, x_0)$ accurately, we can evaluate the AVaR function completely by solving the outer minimization. For this purpose, let us discover how the function $f(s, \alpha, x_0)$ behaves over s.



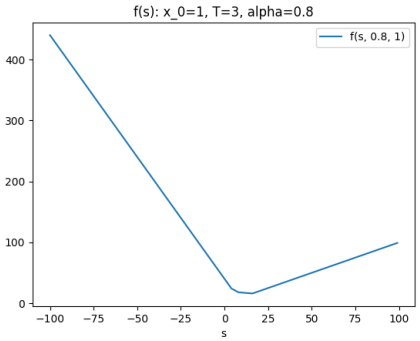
(a) $\alpha = 0$



(b) $\alpha = 0.2$



(c) $\alpha = 0.5$



(d) $\alpha = 0.8$

Figure 3: Behaviour of the $f(s, \alpha, 1)$ over s

Based on the above graphs, it may be seen that $f(s, \alpha, x_0)$ is a convex function. That means that almost any of the numerical optimization techniques for finding the minimum of the convex function can be used. We decided to use Golden Section Search described by Kurdhi et al. (2015).

Golden Section Search

Golden Section Search is a convex optimization technique that finds a minimum value of a convex function (Kurdhi et al., 2015). Given the initial search interval $[a, b]$, the method generates two points x_1 and x_2 such that:

$$b - x_1 = x_2 - a = \gamma(b - a) \tag{6}$$

for some $\gamma \in \left(\frac{1}{2}, 1\right)$. Let us consider the case where the value of the function at x_1 is smaller than x_2 so that the search interval becomes $[a, x_2]$. Then, we generate the third point x_3 such that

$$x_2 - x_3 = x_1 - a \tag{7}$$

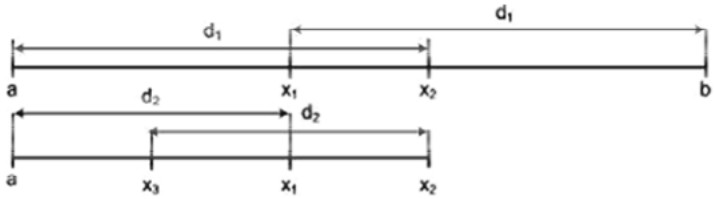


Figure 4: Visualization of Golden Section Search (Kurdhi et al., 2015)

Since the ratio γ must remain the same for each iteration of the algorithm, we get:

$$x_1 - a = x_2 - x_3 = \gamma(x_2 - a) \tag{8}$$

If we substitute values of x_1 and x_2 from (6) to (8) we get:

$$b - \gamma(b - a) - a = \gamma(a + \gamma(b - a) - a)$$

$$1 - \gamma - \gamma^2 = 0$$

$$\gamma = \frac{-1 \pm \sqrt{5}}{2}$$

Since the only value of γ corresponding to the condition $\gamma \in \left(\frac{1}{2}, 1\right)$ is $\frac{-1+\sqrt{5}}{2}$, we use this value.

Numerical Experiments

Here are some obtained AVaR values alongside the optimal s values for different T, α and x_0 values:

T	α	x_0	AVaR	s^*
2	0	1	5	0
5	0.5	1	11	11
10	0.2	5	36.67	-1

Table 1: AVaR obtained using Golden Section Search.

6 Conclusion

In conclusion, this work offers an approach for calculating the Average Value-at-Risk (AVaR) function by formulating it as a discrete time finite horizon stochastic optimal control problem.

To solve the problem the brute force and Approximate Dynamic Programming(ADP) techniques were used. The brute force solution calculates the AVaR exactly, but it is very inefficient in terms of time($O\left(|\mathcal{A}|^{T+1} \times 2^T \times T\right)$). The ADP solution evaluates the function approximately, but accurately and uses much less time to compute the answer $O(T \times N \times |\mathcal{A}|)$. By analyzing the convergence over the number of samples of the ADP solution and the difference of the obtained answer with the answer obtained using brute force the project verifies the stability and accuracy of the solution. Additionally, the golden section search was used to compute the AVaR function entirely. Overall, this study provides an algorithm for the evaluation of AVaR that could be used in market risk management.

References

Acerbi, C., & Tasche, D. (2002, 07). On the coherence of expected shortfall. In (Vol. 26, p. 1487-1503). DOI: 10.1016/s0378-4266(02)00283-2

Bertsekas, D. P. (2017, 6). A series of lectures on approximate dynamic programming. In (p. 7). Retrieved from http://ocps17.imtlucca.it/slides/bertsekas-1.pdf

Bertsekas, D. P., & Shreve, S. E. (1996). Stochastic optimal control : the discrete-time case. Belmont, Massachusetts: Athena Scientific Publ. C.

Bäuerle, N., & Ott, J. (2011, 12). Markov decision processes with average-value-at-risk criteria. In (Vol. 74, p. 361-379). DOI: 10.1007/s00186-011-0367-0

Bäuerle, N., & Rieder, U. (2010, 12). Markov decision processes. In (Vol. 112, p. 217-243). DOI: 10.1365/s13291-010-0007-2

Gagniuc, P. (2017, 05). Markov chains: From theory to implementation and experimentation.. DOI: 10.1002/9781119387596

Kurdhi, N., Sulandari, W., Martini, T., Hartatik Uns, H., & Yudhanto, Y. (2015, 12). Golden section search optimization technique for stochastic inventory problem. In (Vol. 99, p. 205-220). DOI: 10.17654/MS099020205

Rockafellar, R., & Uryasev, S. (2002, 07). Conditional value-at-risk for general loss distributions. In (Vol. 26, p. 1443-1471). DOI: 10.1016/s0378-4266(02)00271-6

Yang, X. (2020, 03). Markov chain and its applications.. DOI: 10.13140/RG.2.2.12289.61287