Optimal Control Problem

Yerdaulet Kappar
Supervisor: Kerem Ugurlu

April 2024

Abstract

This research paper delves into optimizing the Average Value at Risk (AVaR) using Approximate Dynamic Programming (ADP) in the context of optimal control problems. The study focuses on comparing different numerical optimization methods to achieve that. The methods include Bisection, Gradient Descent, Simulated Annealing, and Conjugate Gradient. The purpose is to assess their accuracy and computational effectiveness in optimizing AVaR function within discrete time, finite horizon settings.

Keywords: Keywords: approximate dynamic programming, reinforcement learning, average value-at-risk, optimal control, Markov decision processes, stochastic modelling, convex optimization, optimization techniques

1 Introduction

In finance, uncertainty and complexity are common, making the evaluation and management of risk essential for strategic planning and investments. One of the key tools for measuring risk is Average Value at Risk (AVaR), which provides a detailed view of potential losses in financial portfolios and decisions by focusing on expected losses in worst-case scenarios. However, the computation of AVaR can be complex, particularly in scenarios involving optimal control problems with discrete time frames and finite horizons.

To address this complexity, researchers have explored the use of Approximate Dynamic Programming (ADP) as a computational method. ADP offers an alternative to exact computation techniques, providing precise approximations of AVaR with reduced computational burden. This study focuses on enhancing AVaR calculation through outer minimization approaches within the ADP structure. The minimization process is crucial in determining the scalar value that significantly influences AVaR computations.

The study compares methods like Bisection, Gradient Descent, Simulated Annealing, and Conjugate Gradient to assess their effectiveness, computational efficiency, and accuracy in optimizing AVaR. The investigation aims to determine the advantages and disadvantages of using advanced outer minimization strategies to tackle robust optimal control problems involving uncertainty. By evaluating ADP's ability to approximate AVaR accurately, the study offers valuable insights into its relevance in financial risk management and optimal control. Ultimately, the research highlights the significance of choosing outer minimization approaches and underscores the broader implications of utilizing ADP for complex decision-making challenges in finance and other fields.

2 Definitions

2.1 Average Value-at-Risk (AVaR)

Value-at-risk (VaR) is a metric that estimates the possible loss of an investment over some time at a given confidence level. For example, a 95% VaR for a portfolio might say there's a probability of 5% that the portfolio could lose more than a certain amount in a day. Chun, S. Y., Shapiro, A., & Uryasev, S. (2012) define it as follows:

$$\mathrm{V@R}_\alpha(X) = \inf\{t : F_X(t) \geq \alpha\} \tag{1}$$

Here, $\alpha \in (0,1)$ is a confidence level and F_X is a cumulative distribution function of a random variable X, i.e. $F_X(t) = P(X \leq t)$.

While VaR tells one the potential worst-case scenario, AVaR calculates the average loss one might expect if the situation falls within that worst-case tail end of the distribution. AVaR is also referred to as expected shortfall, conditional value-at-risk, expected tail loss, etc. (Chun, S. Y., Shapiro, A., & Uryasev, S., 2012). It is defined by the following function:

$$\mathrm{AV@R}_\alpha(X) = \inf_t \left\{ t + \frac{1}{1-\alpha}\mathbb{E}[(X-t)_+] \right\} \quad \text{Where } (x)_+ = \max\{0, x\} \tag{2}$$

or similarly:

$$\text{AV@R}_\alpha(X) = \frac{1}{1-\alpha}\int_\alpha^1 \text{VaR}_\tau(X)d\tau. \tag{3}$$

In order not to calculate the V@R function, this paper will use (2) as a definition of AV@R.

2.2 Markov Decision Processes (MDPs)

Markov Decision Processes (MDPs) are used to model decision-making in sequential and stochastic environments. In this model, a decision maker (also called an agent), exists in an environment that changes state randomly in response to the agent's actions. The action taken by the agent impacts the immediate reward received (or cost undertaken) by the agent and the future states and their rewards. The agent's aim is to choose actions that optimize the total reward in a long run. There are several algorithms designed for solving MDPs efficiently, for example, linear programming, dynamic programming, compact representations, etc. These algorithms make modelling and solving some artificial intelligence, economics, robotics problems of planning possible (Littman, M. L., 2001). The MDP framework has the following key components:

S: states ($x_t \in S$)

A: Actions ($a_t \in A$)

$P(x_{t+1}|x_t, a_t)$: Transition probabilities

$R(x_t, a_t)$: Reward

The graphical representation of the MDP model is as follows:

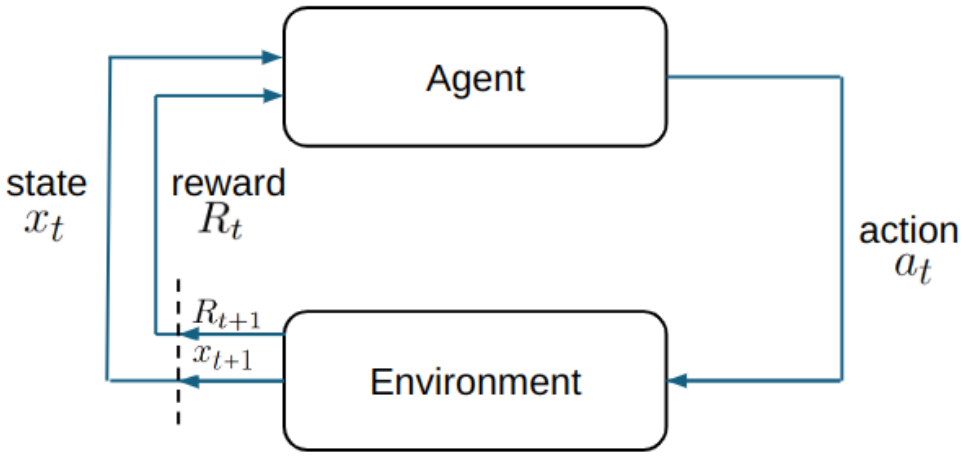


Figure 1: MDP model

The MDP model utilizes the Markov Property, informally formulated as "future state depends only on the current state but not on past states", since the current state contains all the information from the past. Formally, the Markov Property is defined as:

$$P(x_{t+1}|x_1, x_2, x_3, \ldots, x_t) = P(x_{t+1}|x_t) \tag{4}$$

2.3 Optimal Control Problem

This work consider discrete-time finite-horizon stochastic control problem. Shreve and Bertsekas (1978) define such problems as a problem of minimization of the cost function

$$J = \sum_{t=0}^{T} c(x_t, a_t, \xi_t) \tag{5}$$

subject to the system of equations:

$$x_{t+1} = f(x_t, a_t, \xi_t) \tag{6}$$

Here, $x_t \in S$ is a state at time step t with fixed x_0, $a_t \in A$ is an action taken at time t, ξ_t is some random noise that occurs at each time. c is a cost of an action a_t at a state x_t and f is a transition function.

So, the objective of the problem is to find the optimal total cost:

$$J^*(x) = \min_{\pi \in \Pi} E^\pi \left[\sum_{t=0}^{T} c(x_t, a_t, \xi_t)\right] \tag{7}$$

Note that the minimization here is with respect to a policy π, which is, as mentioned, a sequence of taken actions and Π is a set of all possible policies.

2.4 Reinforcement Learning (RL)

Reinforcement learning, often referred to as RL, is a machine learning concept that focuses on training an agent to act optimally from the actions taken upon the given environment. The process of training involves rewarding good actions and punishing bad ones to encourage the development of certain behaviors. This method is frequently used by developers to enable machine learning algorithms to explore environments, for example, video game characters exploring the game's world or robots in factories exploring the labirints. (Sutton, R. S., Bach, F., & Barto, A. G. , 2018).

In reinforcement learning, favorable actions receive positive values that motivate the agent to engage in these behaviors. On the other hand, unfavorable actions are assigned negative values to avoid their occurrence. Following this approach, the agent prioritizes long-term rewards and optimal outcomes over immediate goals.

By providing feedback through rewards and penalties, the decision maker gradually learns to prioritize positive behaviors over negative ones. This method of learning has been widely adopted in a field of artificial intelligence (AI) as a way of directing unsupervised learning through positive and negative reinforcement. This learnign method is also called as semi-supervised learning.

2.5 Approximate Dynamic Programming (ADP)

Dynamic Programming (DP) is a problem solving technique based on dividing the problem into subproblems, solving the smaller problems, and again merge them in one big solution. We can use Dynamic Programming to solve the deterministic optimal control problem by using a solution described in Bertsekas, D.P. (2023):

For all $x_T \in S$ define:

$$J_T^*(x_T) = \min_{a_T \in A} c(x_T, a_T)$$

Then, for $t = T-1, T-2, \ldots, 0$ calculate:

$$J_t^*(x_t) = \min_{a_t \in A} c(x_t, a_t)$$

Note that in the solution above there is no randomness ξ_t since the problem is deterministic. Also, note that we need to take in consideration all the possible values of x_t and t, which might be inefficient since in some cases there are very large amount of possible values of x_t and t. This phenomena is called the *curse of dimensionality* (Bertsekas, D.P., 2023).

To make the idea relevant to the stochastic optimal control problems alongside tackling the curse of dimensionality we use Approximate Dynamic Programming (ADP). The idea is to replace the optimal functions J_t^* with sub-optimal approximations $\tilde{J}_t$. To tackle the curse of dimensionality various techniques are used, this paper will use the Monte-Carlo simulation.

Defining the Problem

This research is focused on the problem of optimizing the Average Value-at-Risk (AVaR) by finding an optimal action sequence $\{a_t\} \in A$. The mathematical formulation of our AVaR minimization problem is expressed as follows:

$$\begin{aligned} \text{AV@R}_\alpha\left(\sum_{t=0}^{T} c(x_t, a_t)\right) &= \min_{s\in\mathbb{R}} \min_{\pi\in\Pi} \left\{ s + \frac{1}{1-\alpha}\mathbb{E}\left[\left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right]\right\} \\ \text{subject to} &\begin{cases} x_{t+1} = x_t + a_t + \xi_t, \\ x_0 \text{ is given.} \end{cases} \end{aligned} \tag{8}$$

The cost function c is chosen not to depend on the occurred randomness and be defined as $c(x_t, a_t) = x_t^2 + a_t^2$. The randomness ξ_t was chosen to be described as $\xi_t = \begin{cases} 1 & \text{, with probability } \frac{1}{2} \\ -1 & \text{, with probability } \frac{1}{2} \end{cases}$

3 Approach to Minimization Over Policy

We approach the inner minimization (minimization over policy) problem by introducing a function:

$$f(s, \alpha, x_0) = s + \frac{1}{1-\alpha} \min_{\pi\in\Pi} \mathbb{E}\left[\left(\sum_{t=0}^{T} c(x_t, a_t) - s\right)_+\right], \tag{9}$$

aiming to find a policy π that minimizes the expected total cost beyond a pre-defined threshold s.

Direct Approach

For a constant values of s, x_0, and α, $f(s, \alpha, x_0)$ can be computed by considering all possible action sequences π and outcomes $\xi \in \{-1, 1\}^T$, with a computational complexity of $O\left(|A|^{T+1} \cdot 2^T \cdot T\right)$.

Algorithm 1 Exhaustive Search Algorithm

Prepare action set $\Pi = A^{T+1}$ and outcomes $\Xi = \{-1, 1\}^T$.
Initialize *min* to some large number.
for each policy π in Π **do**
    Initialize cumulative cost V to zero.
    for each outcome sequence ξ in Ξ **do**
        Reset current cost *current_cost* to zero.
        for each time step $t = 0$ to $T - 1$ **do**
            Accumulate *current_cost* with $c(x_t, a_t)$.
            Update state x_{t+1}.
        end for
        Finalize *current_cost* with cost at time T.
        Adjust *current_cost* for threshold s.
        Update V with adjusted *current_cost* averaged over $|\Xi|$.
    end for
    if V is less than *min* **then**
        Update *min* with V.
    end if
end for
Compute f as $s + \frac{min}{1-\alpha}$.

ADP-Based Approximation

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

This method approximates the function $f(s, \alpha, x_0)$ by dividing the cumulative cost into portions, allowing for a piecewise optimization approach. An additional state $s_{\tilde{t}}$, representing the adjusted cumulative cost up to timestep $\tilde{t}$, is introduced for this purpose.

ADP is applied within a Markov Decision Process framework, where each state includes the current time t, the system state x_t, and the adjusted cumulative cost s_{t-1} so that the state at time t is not only $x_t \in \mathbb{R}$, but $(x_t, s_{t-1}) \in \mathbb{R}^2$. The transition probability for a chosen action a_t is given so that $x_{t+1} = \begin{cases} x_t + a_t + 1 & \text{, with probability } \frac{1}{2} \\ x_t + a_t - 1 & \text{, with probability } \frac{1}{2} \end{cases}$ and $s_t = s_{t-1} + c(x_t, a_t)$ independently from the randomness.

The DP steps are used to calculate the value function at each time step $t = T-1, T-2, \ldots, 0$. As mentioned before, Monte-Carlo simulation was used to approximate the states, and, thus, the value functions at each time step.

4 Approach to Minimization Over Threshold

The core objective of this research is to evaluate and compare different numerical optimization techniques to solve the Average Value-at-Risk (AVaR) optimization problem within an Approximate Dynamic Programming (ADP). The main goal of this section is finding the scalar s that minimizes the AVaR function. To find the optimal value of s, four numerical optimization techniques will be tested: Bisection Method, Gradient Descent, Simulated Annealing, and Conjugate Gradient.

Derivative of AVaR

Some of the tested methods will imply the derivative of the subjective function. For that purpose, let us define a derivative of the function $f(s, \alpha, x_0)$ as

$$\frac{\partial f}{\partial s} = \frac{f(s+\delta, \alpha, x_0) - f(s, \alpha, x_0)}{\delta} \tag{10}$$

for some small real δ.

Bisection Method

The Bisection method is a technique used to find a root of a function given an initial searching interval where the only one root is existing. The method divides the searching interval into two equal-sized sub-intervals and selects on of them where the root is located. Since our objective function is convex, meaning that its derivative has only one root, we can use this method to find the root of the function's derivative.

Algorithm:

Algorithm 3 Bisection Method for Minimizing AVaR with Derivative Approximation

Require: $s_{min}, s_{max}, \epsilon, \delta$
while $|s_{max} - s_{min}| > \epsilon$ **do**
  $s_{mid} \leftarrow (s_{min} + s_{max})/2$
  $f'(s_{mid}) \leftarrow \frac{f(s_{mid}+\delta) - f(s_{mid})}{\delta}$ ▷ Approximate the derivative
  if $f'(s_{mid}) = 0$ **then**
    return s_{mid}
  else if $f'(s_{mid}) > 0$ **then**
    $s_{max} \leftarrow s_{mid}$
  else
    $s_{min} \leftarrow s_{mid}$
  end if
end while
return $(s_{min} + s_{max})/2$

Gradient Descent

Gradient Descent is an algorithm used for finding the minimum of a function that uses the iterative process. Here, we apply it to find the optimal s.

Update Rule:

$$s^{(new)} = s^{(old)} - \lambda f'(s^{(old)}),$$

where λ is the learning rate.

Algorithm 4 Gradient Descent for Minimizing AVaR

Require: Function f, its gradient ∇f (in our case, the derivative of f), initial guess s_0, learning rate λ, tolerance ϵ
Ensure: Optimal s minimizing f
Initialize $s \leftarrow s_0$
repeat
  $s_{\text{new}} \leftarrow s - \lambda \cdot f'(s)$
  $s \leftarrow s_{\text{new}}$
until $f'(s) > \epsilon$
return s

Simulated Annealing

Simulated Annealing is a stochastic method for approximating the global minimum of a given convex function. It is particularly useful for avoiding local minima.

Update Rule: Given a current state s, the algorithm considers a neighboring state s' and moves to it based on the acceptance probability $P(e, e', T)$, where e and e' are the energies (function values) of s and s', respectively, and T is the system temperature.

Algorithm 5 Simulated Annealing Minimization

Require: Function f, initial guess s_0, initial temperature T_0, final temperature threshold $T\_final, cooling\_rate = 0.95, max\_iter = 100$

```
s_current ← s_0
fs_current ← f(s_0)
control_param ← T_0
for i ← 1 to max_iter do
    s_new ← s_current+ uniform random number between −1 and 1
    fs_new ← func(s_new)
    if fs_new < fs_current or random number < exp((fs_current − fs_new)/control_param) then
        s_current ← s_new
        fs_current ← fs_new
    end if
    control_param ← control_param × cooling_rate
    if control_param < T_final then
        break
    end if
end for
return s_current, fs_current
```

Conjugate Gradient

The Conjugate Gradient method is used for finding a solution for systems of linear equations with symmetric positive-definite matrices, but might be adopted to a general case. It is adapted here for non-linear minimization by using it in a line search to find the minimum along the conjugate directions.

Update Rule:

$$s_{k+1} = s_k + \alpha_k d_k,$$

where d_k is the conjugate gradient direction and α_k is the step size.

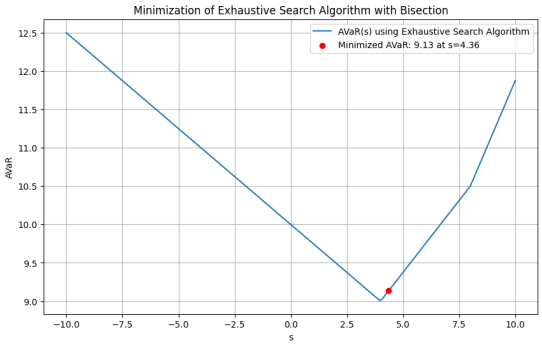
Algorithm 6 Conjugate Gradient Method for Minimizing AVaR

Require: Function f, its gradient ∇f (in our case, the derivative of f), initial guess s_0
Ensure: Optimal s minimizing f

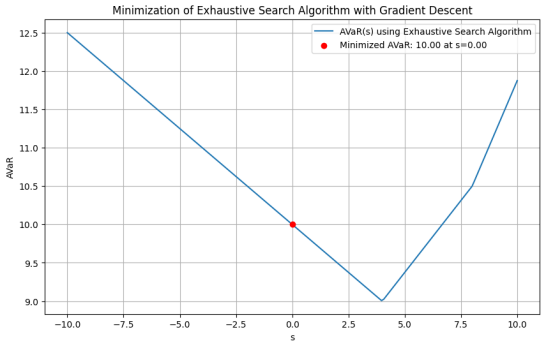
1: Initialize $s \leftarrow s_0$, $g_0 \leftarrow -f'(s_0)$, $d_0 \leftarrow g_0$
2: **for** $k = 0, 1, 2, \ldots$ until convergence **do**
3: $\quad \alpha_k \leftarrow \text{line\_search}(f, s_k, d_k)$
4: $\quad s_{k+1} \leftarrow s_k + \alpha_k d_k$
5: $\quad g_{k+1} \leftarrow -f'(s_{k+1})$
6: $\quad \beta_{k+1} \leftarrow \frac{g_{k+1}^\top g_{k+1}}{g_k^\top g_k}$
7: $\quad d_{k+1} \leftarrow g_{k+1} + \beta_{k+1} d_k$
8: **end for**
9: **return** s_{k+1}

Experimental Results

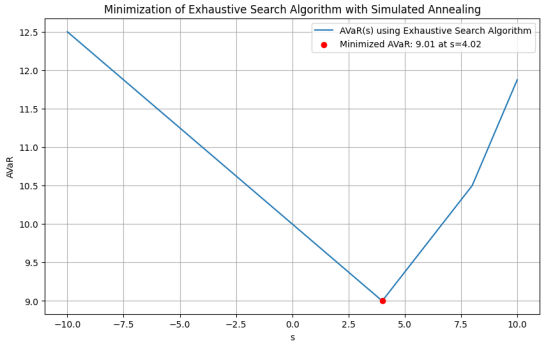
This study conducted a series of experiments to assess the efficiency of each numerical minimization techniques mentioned above. Figures 2 and 3 illustrate the convergence of AVaR evaluations using different methods for various α values and initial states x_0. Table **??** summarizes the optimal s values and the minimized AVaR for each method.



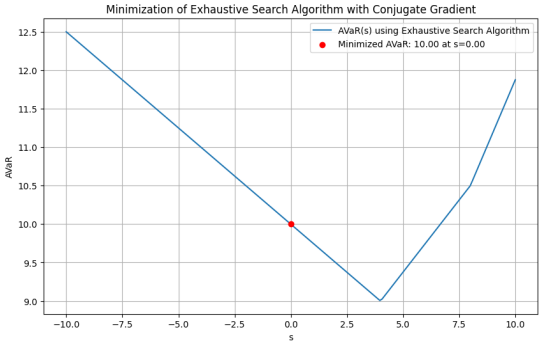
(a) Bisection method with Brute Force



(b) Gradient descent with Brute Force

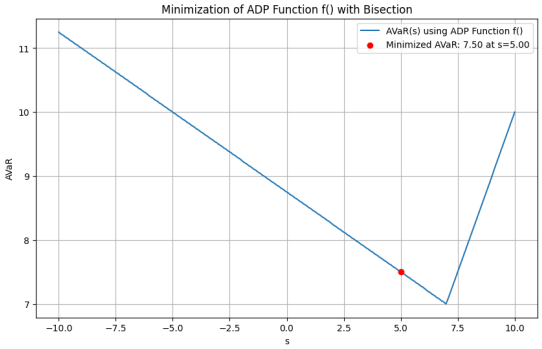


(c) Simulated annealing with Brute Force

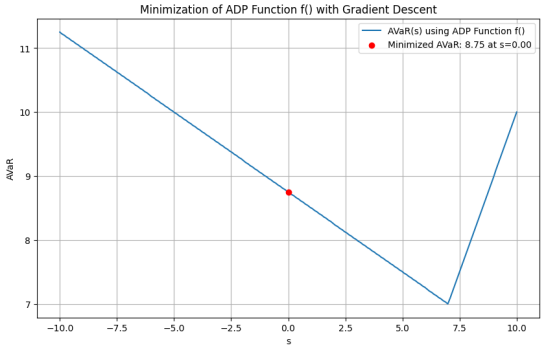


(d) Conjugate gradient with Brute Forcesis

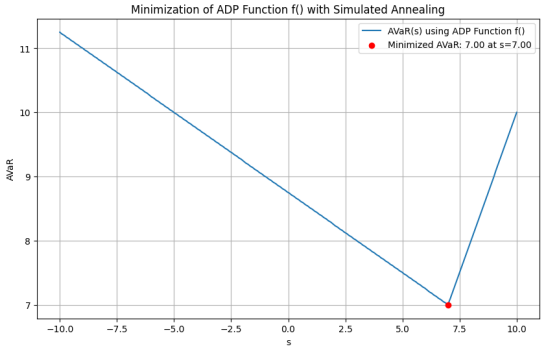
Figure 2: Convergence comparison of different methods for Brute Force.



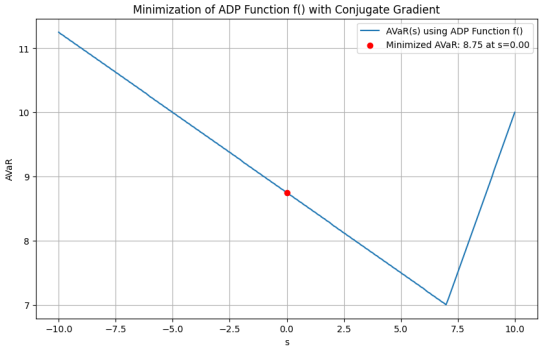
(a) Bisection method with ADP



(b) Gradient descent with ADP



(c) Simulated annealing with ADP



(d) Conjugate gradient with ADP

Figure 3: Convergence comparison of different methods with the ADP.

4.1 Analysis of Outer Minimization Results

The table below presents the results of applying four distinct outer minimization methods: Bisection, Gradient Descent, Simulated Annealing, and Conjugate Gradient. Each of the methods was tested on two separate algorithms, Brute Force and the Approximate Dynamic Programming (ADP). These methods were employed to optimize the AVaR calculation under various configurations.

Table 1: Comparison of Outer Minimization Methods for AVaR Calculation

Method	Brute Force		ADP	
	s^*	AVaR	s^*	AVaR
Bisection	4.359126	9.13467	1.999993	4.46×10^{-11}
Gradient Descent	0.000000	10.00000	1.984317	2.46×10^{-4}
Simulated Annealing	3.881393	9.02965	2.036306	1.32×10^{-3}
Conjugate Gradient	0.000000	10.00000	1.999990	9.79×10^{-11}

From the table, several key observations can be made regarding the performance and applicability of these optimization methods to AVaR calculations:

- **Bisection Method** demonstrates satisfactory performance in both functions, finding the lowest AVaR for the Brute Force and precise minimization for the ADP. This method proves to be particularly effective for problems where the derivative information is not fully available or reliable.

- **Gradient Descent**, while showing the potential for reaching global minima, seems to struggle with precision in the Brute Force, defaulting to an s^* value of 0. Its performance improves with the ADP, indicating its sensitivity to the function's characteristics and initial conditions.

- **Simulated Annealing** presents the best results, achieving the lowest AVaR for both functions. Its stochastic nature allows it to more effectively find global minima, suggesting its suitability for complex, non-linear optimization problems.

- **Conjugate Gradient Method**, despite its powerful convergence properties for certain classes of problems, does not perform optimally for the Brute Force but achieves high precision for the ADP. This discrepancy highlights the method's dependency on the function's gradient behavior and suggests that it may not always be the most reliable choice for non-linear or complex AVaR calculations.

In summary, the effectiveness of each method varies significantly between the two functions, emphasizing the importance of considering the specific characteristics of the AVaR calculation problem when selecting an optimization method. The results also underscore the potential benefits of combining or adapting these methods to suit particular problem settings or incorporating additional problem-specific insights to enhance performance.

5 Conclusion

In summary, this study focused on the optimization of Average Value-at-Risk (AVaR). To evaluate the AVaR on some fixed threshold s, Approximate Dynamic Programming (ADP) was used. We conducted a comparative analysis of different numerical minimization methods, such as Bisection, Gradient Descent, Simulated Annealing, and Conjugate Gradient methods. Our findings show that Simulated Annealing is particularly effective in finding minima in a particular case, making it a suitable choice for such problems. On the other hand, the Bisection method performs well when the information about the derivative is unknown or unreliable. This research highlights the importance of selecting appropriate outer minimization techniques to improve the accuracy of minimizing the AVaR under the stochastic decision making procedure. By improving our understanding of these optimization methods within the ADP framework, this work contributes to developing more efficient tools for financial engineering and other fields where decision-making under uncertainty is critical.

References

Bertsekas, D.P. (2023). A course in reinforcement learning. Athena Scientific.

Chun, S. Y., Shapiro, A., & Uryasev, S. (2012). Conditional value-at-risk and average value-at-risk: Estimation and asymptotics. Operations Research, 60(4), 739–756. Retrieved from https://doi.org/10.1287/opre.1120.1072?

Littman, M. L. (2001). Markov decision processes. international Encyclopedia of the Social & Behavioral Sciences, 9240–9242. Retrieved from https://doi.org/10.1016/b0-08-043076-7/00614-8

Shreve, S. E., & Bertsekas, D. P. (1978, November). Alternative theoretical frameworks for finite horizon discrete-time stochastic optimal control. In (Vol. 16, pp. 953–978). DOI: https://doi.org/10.1137/0316065

Sutton, R. S., Bach, F., & Barto, A. G. . (2018). Reinforcement learning: An introduction. MIT Press Ltd.