CHALMERS UNIVERSITY OF TECHNOLOGY | UNIVERSITY OF GOTHENBURG

Risk-Averse Multi-Armed Bandit Problem with Multiple Plays

A Markovian Bandit Solution

Master's thesis in Computer Science and Engineering

SIRI DAHLGREN
NICHOLAS MARRIOTT

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Risk-Averse Multi-Armed Bandit Problem with Multiple Plays

A Markovian Bandit Solution

SIRI DAHLGREN
NICHOLAS MARRIOTT

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Risk-Averse Multi-Armed Bandit Problem with Multiple Plays
A Markovian Bandit Solution
SIRI DAHLGREN
NICHOLAS MARRIOTT

Supervisor: Milad Malekipirbazari, Data Science and AI Division, Department of Computer Science and Engineering
Examiner: Morteza Haghir Chehreghani, Data Science and AI Division, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Risk-Averse Multi-Armed Bandit Problem with Multiple Plays
A Markovian Bandit Solution
SIRI DAHLGREN
NICHOLAS MARRIOTT
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This study aims to construct an efficient heuristic, referred to as RA, for a risk-averse Markovian multi-armed bandit problem (MAB) with multiple plays. The RA incorporates risk-aversion and multiple plays by modifying the Gittins index strategy. The performance of RA is compared to a risk-neutral version of the Gittins index strategy (RN) on a risk-averse MAB, using Markov Decision Process (MDP) policies as references for optimality. The results demonstrate that RA outperforms RN in the majority of the tested cases, showcasing its effectiveness in a risk-averse setting for multiple plays. Notably, RA exhibits a substantial performance improvement, with up to a 120.86% better performance than RN for MAB instances with rewards generated from a normal distribution, and a remarkable 270.55% better performance for MAB using exponential distributions.

Furthermore, the runtime of RA exhibits a linear growth pattern as the problem size increases, which is in contrast to the exponential growth observed in MDP approaches. The study's findings provide strong evidence of the RA heuristics efficacy in solving risk-averse MAB problems with multiple plays.

Keywords: MAB, Gittins, Markovian bandit, risk-aversion, policy iteration, multiple plays.

Acknowledgements

The thesis was conducted at the Department of Computer Science & Engineering at the joint department of Chalmers University of Technology and University of Gothenburg.

We would like to take this opportunity to express our immense gratitude to our supervisor, Milad Malekipirbazari, who was always friendly and available to support us throughout our thesis. His contribution to our weekly meetings kept us focused and on track to completing this project.

Siri Dahlgren, Gothenburg, 2023-06-12 Nicholas Marriott, Gothenburg, 2023-06-12

Contents

List of Figures

List of Tables

1

Introduction

The aim of this study is to construct an efficient heuristic for a specific variant of the multi-armed bandit problem (MAB). In short, the MAB is a decision-making problem where an agent must choose from a finite set of actions that yields varying rewards. The goal is typically to maximise the cumulative rewards over time. There are several real-life applications that can be modelled as MABs, such as recommender systems, portfolio selections and dialogue systems [1]. Another well mentioned application in the literature for MABs is clinical trials, even if the MAB has yet to be implemented in this setting in real life [2]. In a clinical trial the possible actions could be to try one of several drugs, and the solution to the MAB would show the most efficient drug to try.

In this study, a new heuristic is developed to solve a specific variant of the problem, namely a risk-averse Markovian MAB with multiple plays. This type of MAB has to the best of the author's knowledge not yet been studied. The newly developed heuristic will be referred to as RA for its risk-averse characteristics. To create the RA, some modifications are made to a previously known heuristic for the MAB problem, called the Gittins index strategy. The modifications incorporate the component of risk-aversion as well as multiple plays into the heuristic. Incorporating risk-aversion into the MAB means that the objective of the problem changes from being one of maximising the total expected reward, to also including the aim of minimising the risk of getting any relatively low rewards. The incorporation of multiple plays means that instead of making only one action in every time instance, several actions will be made simultaneously. Taking the example of a clinical trial, this would mean that the new heuristic, RA, could propose two or more drugs for each patient. The drugs would be chosen to both maximise the health improvement of the patients but also to minimise the risk of severe side-effects.

The RA heuristic is evaluated by comparing its results to two other standard heuristics. One of these heuristics, referred to as RN, is a generalised risk-neutral version of the Gittins index for multiple plays. The RA and RN are applied to the same risk-averse MAB. The policies generated by RA and RN are compared to benchmark policies, which are considered optimal. The degree of deviation from the optimal policies, determines the performance of the algorithms. In addition to assessing how well RA achieves the goal of the risk-averse MAB with multiple plays, its time complexity is also evaluated.

The structure of this thesis is as follows: The Background explains the different components which this study relies on in greater depth and gives an overview of relevant prior research. The Methodology section provides an explanation of the design of the heuristics utilised in this study, as well as the experimental setup. The Results show numerical results from the evaluation of RA and the Discussion discusses both the results and the limitations of this thesis. The Conclusion summarises the most important findings.

2 Background

This section presents both MAB in general and the specific type of MAB focused on in this thesis. Various approaches for solving the Markovian MAB are explained, namely as an MDP (via policy or value iteration) or with the use of the Gittins Index. An overview is given of prior research for risk-averse MAB, MAB with multiple plays, and possible practical application for the risk-averse MAB with multiple plays.

2.1 Multi-Armed Bandit Problem

MAB is a classic problem within the reinforcement learning framework. The problem is commonly described as an individual instructed to choose to pull an arm on a slot machine at a casino. As there are multiple machines, and thereby multiple arms available at the casino to choose from, this leads to the name multi-armed bandit, with bandit referring to the machine taking money from the losing players. The chosen machine will offer a reward depending on an underlying distribution corresponding to that machine. The player will make a stepwise choice of playing the same or another arm. This individual playing on the machine wishes to outsmart the machines, leaving the casino with the highest possible reward. The MAB problem can take on a variety of restrictions, which will result in different variations of the problem. Each variation calls for a different approach, which results in different optimal strategies for maximising the potential reward from the available slot machines.

When studying the MAB, known rewards and unlimited time horizons are sometimes used to evaluate new heuristics for the exploration phase. Controlled experiments with these settings can be performed to assess the performance of various exploration algorithms in a more controlled environment. Often in practical settings the underlying reward distributions of the MAB are unknown and the allocation resources are limited. This would mean that the individual in the example could only choose an arm to pull a certain number of times. This leads to a balancing act between exploration vs exploitation. This definition refers to how much time and resources does the individual invest in discovering the extent of profitability of the available machines. Exploration is when the individual tests different actions, to try and find the highest rewards. Exploitation is when the individual uses the remaining resources on the machine they believe will result in the maximum rewards. With finite resources, an extensive exploration would result in less time for exploitation.

With a limited exploration, the individual may choose worse actions leading to lower potential rewards.

There are several variants of the MAB problem. Bubeck and Bianchi suggest that these may be divided into three core categories: Stochastic bandits, Adversarial bandits and the focus of this thesis, Markovian bandits [3]. There are several subtypes for each of these categories, however the distinguishing factor between them is their reward structure.

Stochastic bandits have a statistical nature. In this MAB each arm is associated with a fixed reward distribution. The rewards from each arm may vary between time steps, but are always randomly drawn from the same distribution.

Adversarial bandits on the other hand do not rely on statistical assumption nor potential probabilities. The rewards are instead decided by an omniscient adversary, who fully understands the players tactic and has control over the payouts. As such, the adversary would select a certain reward structure with the goal of minimising the players rewards.

Markovian bandits also rely on statistics to determine the rewards, but in a different way than Stochastic bandits. In Markovian bandits, each arm can be seen as a Markov chain, where every arm has a certain number of states. Each state of each arm is associated with a fixed reward and fixed transition probabilities to the other states of that arm.

This thesis focuses on the aforementioned Markovian multi-armed bandit problem, which shall continue being referenced as MAB. The specific subtype incorporates risk-aversion and multiple plays. It is investigated in a setting with known rewards and an unlimited time horizon. The arms of the MAB will be independent from each other and its rewards and transition probabilities stay fixed over time. Each arm of the MAB will have an equal number of states, and there will be no termination states.

2.2 Markov Decision Process

Just as Markovian bandits, Markov Decision Processes (MDPs) can be seen as a framework where there are certain states, and certain actions can be taken in each state. When an action is taken, the process moves to another or the same state with a certain probability, and presents a certain reward depending on which state it reaches, and/or moves from. These probabilities are called transition probabilities and show the chance of moving from a state to another. They must adhere to the constraint that the sum of all transition probabilities from a given state to its reachable states equals one. This process is stochastic because the transition probabilities will determine the outcome of each action for each timestep. The optimal solution to an MDP is presented as a policy, which shows the best action to play for each state, to reach the maximum expected total reward.

Algorithms such as value iteration and policy iteration can be used for solving the MDP. This can be done by applying Markov Decision theory, as presented in Puterman [4]. Formulating the MAB as an MDP means that the number of states of the MDP will grow exponentially when the number of arms or the number of states per arm increases. This will result in the curse of dimensionality. Gittins index gives a less computationally expensive way of finding a high reward giving policy for the MAB. However, using Gittins indices might result in a suboptimal policy, for different subtypes of the MAB.

The solutions for the MDP can be found with the help of Bellman's equation which is incorporated in algorithms such as the value iteration and policy iteration. Bellmans equation is described as follows:

$$V(x) \leftarrow \ argmax_a \sum \ _{y,x} P(y|x,a) \left[r(y|x,a) + V(y) \right] \tag{2.1}$$

where $V(x)$ is the assigned value for each state,

a is an action,

x is the current state and y the state to which the process transits next,

$P(y|x,a)$ is the probability of moving from x to y when taking action a,

$r(y|x,a)$ is the reward that the player receives for transitioning from state x to state y when taking action a,

γ is the discount factor. It determines the weight of immediate rewards relative to future rewards and plays a crucial role in dynamic programming for infinite time horizons, enabling convergence through the discounting of future rewards.

Bellman's equation includes a recursive element since the values of a state (x) depend on the value of the neighbouring states (y). When Bellman's equation is used in value iteration or policy iteration, the values can be calculated iteratively until an optimal policy (*) for the MDP is found.

2.3 MAB as an MDP

This section shows how a small-scaled version of a MAB would be constructed as an MDP in this study.

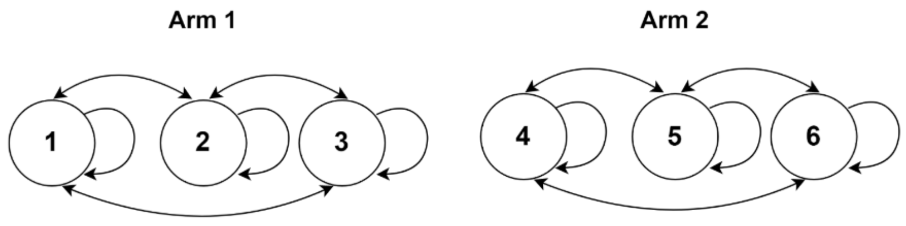


Figure 2.1: A small MAB example consisting of two arms, three states per arm

Figure 2.1 shows two arms with three states each. Each arm functions like a Markov Process, with their own fixed transitional probabilities. Each state gives a fixed reward when the process transitions from it. To construct the MAB as an MDP, each state of Arm 1 needs to be combined with each state of Arm 2, and together work as a state of the MDP. This leads to the number of states of the MDP to be the states per arm to the power of the number of arms. For this small MAB example the following states are created for the MDP, see Table 2.1.

Table 2.1: States of MDP, generated from Arm 1 and Arm 2

State in MDP	**State of [Arm 1 & Arm 2]**
1	$[1, 4]$
2	$[2, 4]$
3	$[3, 4]$
4	$[1, 5]$
5	$[2, 5]$
6	$[3, 5]$
7	$[1, 6]$
8	$[2, 6]$
9	$[3, 6]$

Table 2.2: Example of transition probabilities for Arm 1.

State	**1**	**2**	**3**
1	0.55	0.25	0.2
2	0.71	0.14	0.15
3	0.35	0.61	0.05

Table 2.3: Example of rewards for Arm 1

State	**1**	**2**	**3**
1	5.44	5.44	5.44
2	5.57	5.57	5.57
3	5.79	5.79	5.79

Table 2.2 provides the transition probabilities for Arm 1, with each row representing the current state and the columns indicating the probabilities of transitioning to the next states. For instance, Arm 1 has a transition probability of 0.25 for moving from state 1 to state 2. Similarly, the probability of Arm 1 transitioning from state 2 to state 1 is 0.71.

Table 2.3 shows the rewards associated with Arm 1, following the same structure as the transition probabilities. Thus, if arm 1 transitions from state 1 to state 2, the corresponding reward is 5.44. Likewise, if arm 1 transitions from state 2 to state 1, the reward value is 5.57.

Table 2.4: Example of transition probabilities for Arm 2.

State	**4**	**5**	**6**
4	0.50	0.40	0.10
5	0.70	0.15	0.15
6	0.30	0.60	0.10

Table 2.5: Example of rewards for Arm 2.

State	**4**	**5**	**6**
4	5.21	5.21	5.21
5	5.89	5.89	5.89
6	5.08	5.08	5.08

Table 2.4 and 2.5 show the transition probabilities and rewards of Arm 2. The matrices operate in a similar manner to the rewards and transitional probabilities of Arm 1, but they possess distinct values specific to Arm 2.

Table 2.6: Example of transition probabilities for action 1.

State	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	0.55	0.25	0.2	0	0	0	0	0	0
2	0.71	0.14	0.15	0	0	0	0	0	0
3	0.35	0.61	0.05	0	0	0	0	0	0
4	0	0	0	0.55	0.25	0.2	0	0	0
5	0	0	0	0.71	0.14	0.15	0	0	0
6	0	0	0	0.35	0.61	0.05	0	0	0
7	0	0	0	0	0	0	0.55	0.25	0.2
8	0	0	0	0	0	0	0.35	0.61	0.05
9	0	0	0	0	0	0	0.35	0.61	0.05

Table 2.7: Example of rewards for action 1.

State	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
1	5.44	5.44	5.44	0	0	0	0	0	0
2	5.57	5.57	5.57	0	0	0	0	0	0
3	5.79	5.79	5.79	0	0	0	0	0	0
4	0	0	0	5.44	5.44	5.44	0	0	0
5	0	0	0	5.57	5.57	5.57	0	0	0
6	0	0	0	5.79	5.79	5.79	0	0	0
7	0	0	0	0	0	0	5.44	5.44	5.44
8	0	0	0	0	0	0	5.57	5.57	5.57
9	0	0	0	0	0	0	5.79	5.79	5.79

Table 2.6 and Table 2.7 display the rewards and transition probability matrices specifically for action 1 (pulling Arm 1) in the example of a two armed MAB with three states per arm. In these matrices, zeros indicate the impossibility of transitioning between the corresponding states when performing action 1. For instance, if there is a zero in the entry representing a transition from state 1 to state 4 when action 1 is taken, it signifies that such a transition is not achievable. This is because when action 1 (pulling Arm 1) is performed, only the part of the MDP state corresponding to Arm 1 can change, and thus transitioning into state 4, which would require a change in the second part of the MDP state (from state 4 to 5) is not possible.

2.4 Value Iteration

Value iteration is a dynamic programming algorithm used to determine optimal policies for MDPs. It works by iterating over the states and uses Bellman's equation to update the value of each state. Its an example of backward induction. For every state and in every iteration, the algorithm evaluates what action that maximises the state's value. When no value improves much more between the iterations, the equation converges and breaks. The actions that give the highest values at this point will be extracted and saved as the optimal policy. Value iteration is guaranteed to converge to the optimal policy for an MDP with a finite time horizon. For an MDP with infinite time horizon, the difference between the obtained policy and the true optimal policy is bound by $\frac{2\epsilon}{1-\gamma}$.

Algorithm 1: Value Iteration

```
Initialize $v_{old}$, $v$ and $\pi$ arbitrary for all $x \in X$
Find optimal values while $\Delta$ is less than $\epsilon$:
while $\Delta < \epsilon$ do
    $V_{old} \leftarrow V$
    $\Delta \leftarrow 0$
    for $all\ x \in X$ do
        $V(x) \leftarrow max_a \sum_{y,x} P(y, x|x, a) [r + \gamma V_{old}(y)]$
        $\Delta \leftarrow max(\Delta, V(x) - V_{old}(x))$
    Extract the optimal policy:
    for $all\ x \in X$ do
        $\pi(x) \leftarrow argmax_a \sum_{y,x} P(y, x|x, a) [r + \gamma V_{old}(y)]$
```

Value iteration can be computationally expensive for MDPs with a large number of states and actions, and alternative algorithms such as policy iteration may be more efficient.

2.5 Policy Iteration

Policy iteration is another type of dynamic programming algorithm used to determine optimal policies for MDPs. This algorithm consists of two steps and alternates between these until the optimal policy is found, being policy evaluation and policy improvement. Before starting, an arbitrary policy is set. It then runs the policy evaluation step where it iterates over all states and uses Bellman's equation to update the value of each state. Instead of calculating the values based on the best possible action for every state, as in value iteration, it only uses the action determined by the policy. It continues iterating over the first step of the algorithm until the values for all states have converged. It then moves to the second step of the algorithm, policy improvement.

In policy improvement Bellman's equation is used once again to calculate the value for each state, now considering all possible actions. If the value of a state can be improved with any other action than the one previously suggested by the policy, the policy is updated to the action that maximises the value. The optimal policy is found when the policy stops updating.

Algorithm 2: Policy Iteration

```
Initialize $v_{old}$, $v$ and $\pi$ arbitrary for all $x \in X$
Repeat until optimal policy is found:
1. Policy evaluation:
while $\Delta < \epsilon$ do
    $\Delta \leftarrow 0$
    for all $x \in X$ do
        $V_{old}(x) \leftarrow V(x)$
        $V(x) \leftarrow max_a \sum_{y,r} P(y, r|x, \pi(x)) [r + \gamma V_{old}(y)]$
        $\pi(x) \leftarrow argmax_a \sum_{y,r} P(y, r|x, \pi(x)) [r + \gamma V_{old}(y)]$
        $\Delta \leftarrow max(\Delta, V(x) - V_{old}(x))$
2. Policy Improvement:
for all $x \in X$ do
    $a \leftarrow \pi(x)$
    $\pi(x) \leftarrow argmax_a \sum_{y,r} P(y, x|s, a) [r + \gamma V_{old}(y)]$
    if $a \neq \pi(x)$ then optimal policy is not found
```

Policy iteration, just as value iteration, is guaranteed to converge to the optimal policy for a MDP with a finite time horizon. For an MDP with infinite time horizon, the difference between the obtained policy and the true optimal policy will also be bounded in the same fashion as value iteration, by $\frac{2\epsilon}{1-\gamma}$.

2.6 Gittins Index

A more computationally efficient way of solving the Markovian MAB problem than using Markov Decision theory is via the Gittins index strategy. Gittins proposed the concept of "dynamic allocation indices" in his paper from 1979, which later became known as Gittins indices [5]. In his paper he also showed the optimality of this approach. Another proof of the optimality was shown by Whittle in 1980, with the help of dynamic programming [6]. The Gittins index works by giving every state of every arm an index using forward induction. The optimal strategy is to always play the arm whose current state has the highest index value. The calculations of the indices are based on rewards and probabilities of exclusively the considered arm. Only considering the maths of one single arm at the time makes the Gittins index more computationally efficient to use than to solve the problem as an MDP.

To explain the Gittins index mathematically, let an arm i be a Markov Process with a finite state space X^i. X_t^i is the state of arm i at time t, where $i = 1, 2, \ldots, K$. Let $u_t = \left(u_t^1, \ldots, u_t^K\right)$ denote the decision that the individual/player takes at time t. Then u_t^i is equal to:

$$u_t^i = \begin{cases} 1 \text{ if the individual plays bandit } i \\ 0 \text{ otherwise} \end{cases} \tag{2.2}$$

If u_t^i takes a value of 0, this means the individual does not choose to play that arm and the arm will remain frozen. If it takes that value of 1, the individual decides to play the arm, resulting in X_t^i progressing in a Markovian manner.

When playing an arm, the individual will receive a reward $r^i\left(X_t^i\right)$, with r_i representing the reward of arm i, for the state X of that arm i and for the time t. Future rewards are seen as less meaningful than present rewards. Due to the problem's stochastic nature the aim is to maximise the expected total discounted reward:

$$\mathbb{E}_\pi = \left[\sum_{t=0}^{\infty}\gamma^t\sum_{i=1}^{\infty}r^i\left(X_t^i\right)u_t^i\Big|X_0 = x_0\right] \tag{2.3}$$

π is the playing strategy/function that denotes which action/arm to play,

$x_0 = (x_0^1, \ldots x_0^n)$ are the initial starting state of all arms,

$\mathbb{E}$ is the expected total reward,

γ is the discount factor,

To find the function π that maximises the total expected discounted reward, Gittins uses an equation for giving every state of every arm a value,

$$v^i\left(x^i\right) = \max_{\tau>0} = \frac{\mathbb{E}\left[\sum_{t=0}^{\tau}\gamma^t r^i\left(X_t^i\right)\Big|X_0^i = x^i\right]}{\mathbb{E}\left[\sum_{t=0}^{\tau}\gamma^t\Big|\left(X_0^i\right)X_0^i = x^i\right]} \tag{2.4}$$

Here, τ is the stopping time.

There are multiple algorithms to select from within the scope of Gittins [7]. They can be classified into two groups, "offlin" and "onlin". Offline algorithms use the fact that the optimal playing time will depend on which states the arms are in, rather than a certain amount of time steps. Taking this into account the equation can be rewritten as follows:

$$v^i\left(x^i\right) = \max_{S(a)\in X} = \frac{\mathbb{E}\left[\sum_{t=0}^{\tau(S(a))}\gamma^t r^i\left(X_t^i\right)\Big|X_0^i = x^i\right]}{\mathbb{E}\left[\sum_{t=0}^{\tau(S(a))}\gamma^t\Big|\left(X_0^i\right)X_0^i = x^i\right]} \tag{2.5}$$

Offline algorithms introduce two sets $C(a)$ and $S(a)$. $C(a)$, known as the continuation set, contains all states with higher valued indices than the current state of the arm. $S(a)$, known as the stopping set, includes all the states which have the same or lower valued indices.

$$C(a) = \{b \in X : b \succ a\} \tag{2.6}$$

$$S(a) = \{b \in X : a \succeq b\} \tag{2.7}$$

a and b are states of an arm.

The chosen algorithm for this study is a modified version of a well-known method referred to as the largest-remaining-index algorithm. It identifies the values of the states for one arm at a time according to decreasing order [8]. This means the continuation set will start as empty and for every iteration of the algorithm one more state will be added to $C(a)$.

Algorithm 3: Largest-remaining-index algorithm

```
Initialize as empty; $C$ for continuation set, $G$ for Gittins indices set and $S$ for stopping set
for $all\ arm \in ARMS$ do
    $G(x_1) \leftarrow max_r$
    $C(x_1) \leftarrow argmax_r$
    $S \leftarrow x \in X \backslash x_1$
    for $k$ in range(S) do
        $P^{(k)}_{y,x} \leftarrow P_{y,x}1\left[x \in C(X_k)\right]$
        $d^k \leftarrow \left(I - \gamma P^{(k)}\right)^{-1} r$
        $b^k \leftarrow \left(I - \gamma P^{(k)}\right)^{-1} 1$
        $V^k \leftarrow d^k / b^k$
        $C(x_k) \leftarrow argmax_{v^k}$
        $G(x_k) \leftarrow max_{v^k}$
        $S \leftarrow x \in X \backslash x_1$
Extract policy
```

The optimality of the Gittins index relies on some assumptions of the MAB problem. For example, the decisions of the process need to be irrevocable [9]. This means that no decisions throughout the process should change the underlying structure of the problem. This is for example true for stationary MABs which is the focus of this study. The term "stationary" implies that the rewards and probabilities associated with each arm remain constant throughout the decision process.

2.7 Multiple Plays

In the standard form of MAB, that is the MAB with single plays, only one arm is played at each time step. In MAB with multiple plays, the player will play a certain number of arms simultaneously at each timestep. Each individual arm can at most be played one time for each time step. The objective is still generally to maximise the cumulative rewards. The solutions for a MAB with multiple plays could be applied in various fields where resources have the possibility of being located to more than one instance at a time. The paper by Song and Teneketzis shows how a search problem with multiple sensors can be modelled as a multi-armed bandit with multiple plays [10].

Pandelis and Teneketzis defined certain conditions under which the Gittins index is known to be optimal for the MAB with multiple plays [11]. They state that if a current state's value of an arm is greater than that of the next best playable state, it must also be greater or equal to the next best state when multiplied by $(1-\gamma)$. When this condition is satisfied, it is conclusive that playing these two arms together will yield the optimal result, for a problem where two arms should be played simultaneously. The result may still be optimal even when the condition does not hold, but it is no longer guaranteed.

2.8 Risk-Aversion

In the standard form of MAB, that is the risk-neutral MAB, the goal is usually to maximise the cumulative rewards. In a risk-averse MAB the goal is to both reach a high amount of cumulative rewards, but to also avoid any individual reward to be particularly low. Another way of explaining the risk-averse MAB is to say that the objective is not only to maximise the total rewards but to also minimise variability in the rewards.

The importance of the risk-averse MAB shines when evaluating scenarios where risks can be detrimental. One popular example of application is when introducing a new drug within the healthcare sector; if a drug has a very positive effect on 95% of the patients but greatly harms or kills 5%, the standard MAB might still choose this drug prior to a safer but less effective option. A risk-averse MAB however would take the risk into consideration and likely propose a less risky option. Another example of where it is important to incorporate risk-aversion into the model is within financial applications. This is especially true for investors who have a tight budget and who cant afford to gamble with its money.

There have been different proposals of how risk can be incorporated into the MAB. Denardo, Park and Rothblum proposed the use of utility functions [12]. By increasing the values of the rewards obtained from the environment according to an exponential utility-function, they made their risk-averse approach into one that favoured higher rewards and thereby penalised low ones. They showed that their algorithm can be preferable to the risk-neutral heuristic in some risk-averse settings of the MAB.

Malekipirbazari and Cavus showed how risk can be incorporated into MAB with the help of dynamic coherent risk measures [13]. They argue that a downside of using utility functions is that such functions that correspond to the player's risk-aversion can be hard to find, and the resulting solutions may be difficult to interpret. Malekipirbazari and Cavus work relies on the work of Ruszczyski who showed how risk-aversion can be included into a Markov Decision Process [14]. The risk measures they use are the first-order mean-semideviation and the mean-CVaR. Their paper shows that their algorithm gives optimal or near optimal results depending on the MABs settings. In this study, the first-order mean-semideviation will be incorporated into the MAB in a similar fashion as in Malekipirbazari and Cavus [13].

2.9 Risk-Aversion and Multiple Plays

The combination of both risk-aversion and multiple plays in a Markovian MAB, to the authors knowledge, has not yet been researched. When applying a model that takes risk-aversion and multiple plays into account, to real-world applications, this may resolve optimality issues within, but not limited to, the following scenarios: Natural disaster rescues, ambulance redeployments, stock-portfolio selections, and clinical trials.

In a natural disaster rescue operation, the aim would be to optimally allocate multiple rescue services to certain locations to reduce the risk of mortality. To model the problem with multiple plays is intuitive as there would usually be more than one service working simultaneously in such an operation. Incorporating risk-aversion to this model, could help the allocating decision maker to not make too risky decisions. A riskful allocation decision may lead to a larger variance of people being rescued, which might not be preferable if the timeframe is limited. After a natural disaster, the priority might be to find fewer people with a higher certainty, than to risk not finding anyone.

As for ambulance redeployments, the aim would be to optimally allocate ambulances in certain areas, to be as close to people in emergencies as possible. Multiple plays in this model would represent the possibility of allocating several ambulances at once. Incorporating risk-aversion in the model could give the benefit of fewer transportations being too long for the patient's to survive. A MAB with multiple plays and risk-aversion has earlier been proposed as a model for this use case. This model has been investigated by Sahin et al., [15], yet in their application the use of stochastic MAB was chosen instead of Markovian MAB. For the scenario of stock-portfolio selection, a MAB with multiple plays and risk-aversion could potentially reduce an investors risk, while having numerous stock options to select from. The model would strive for choosing both secure and high-yielding stock options.

When using MAB solutions for clinical trials, these should intuitively be risk-averse, as the selected drug directly impacts an individuals health and wellbeing and could potentially have severe side-effects. When adding multiple plays to the MAB, this opens up for new possibilities. One model could propose for one patient to take several medicines at once. This could allow a person to get the best medicines available, also when it is preferable to use more than one treatment simultaneously.

3

Methodology

This section presents the heuristics employed in this study and discusses the incorporation of risk-aversion and multiple plays. The optimal policy of the MAB was chosen from three different algorithms: value iteration and two methods of policy iteration. An overview of these algorithms along with their respective time complexities are provided.

3.1 Algorithms

The objective of this study is to assess the effectiveness of a new heuristic called RA. To evaluate RA, which aims to find optimal or near optimal policies for the risk-averse MAB with multiple plays, two more algorithms are created, specifically RN and a Benchmark. RN follows a similar approach as RA but lacks the risk-averse component, focusing on finding optimal or nearly optimal policies for the risk-neutral MAB with multiple plays. Both RA and RN are generalisations of the Gittins index. The Benchmark is created by utilising Markov Decision Theory and aims to identify optimal policies for the risk-averse MAB with multiple plays.

Table 3.1: Algorithms Employed in the Study

Algorithms	**Risk-attitude**	**Based on**
RA	Risk-averse	Gittins Index
RN	Risk-neutral	Gittins Index
Benchmark	Risk-averse	MDP

All the algorithms are implemented using the Python 3.10 programming language. For an overview of the three algorithms see Table 3.1.

3.2 Incorporating Risk-Aversion

To incorporate risk into the algorithms and thereby into the MAB, a dynamic coherent risk measure, namely first-order mean-semi deviation, was used. Incorporation of this risk measure into an MDP is described by Ruszczynski [14]. Additionally, Malekipirbazari and Cavus [13] generalises the same measure to be used in a modified algorithm for Gittins indices. The risk measure works by penalising rewards that are below average. The formula of first-order mean-semi deviation is provided below.

$$\rho(Y) = E(Y) - \kappa E(E(Y) - Y) \qquad \kappa \in [0,1] \tag{3.1}$$

Y is a random reward,

$E(Y)$ is the expected reward,

κ represents the weight of the risk-averse component of the equation.

Expected reward $E(Y)$ subtracted by random reward Y results in either a positive value or 0, as all negative values are set 0.

The equation above can be added to the benchmarks to add risk-aversion.

3.2.1 Risk-Aversion in Benchmark

For risk-aversion in the benchmark, the value iteration or policy iteration, values are now calculated as:

$$u(X) = \sum_{x \in X} P(y|x, \pi(x)) \left[r(x, y, \pi(x)) + \beta v(y)\right] \tag{3.2}$$

$$v(X) = .ax_{a \in A} u(x) - \kappa \sum_{u \in X} P(y|x, \pi(x)) \left[u(x) - r(x, y, \pi(x)) + \beta v^{\pi}(y)\right] \qquad x \in X \tag{3.3}$$

$u(x)$ is the expected reward,

$v(x)$ is the maximum value of a state x,

$x, y \in X$ are states.

In this new formula, for calculating the values, first-order mean-semi-deviation has been added to the equation. Just as before, the aim of the algorithm is to find the action that maximises this value for each state.

3.2.2 Risk-Aversion in RA

Some modifications are made to the largest-remaining-index algorithm to incorporate risk-aversion. An iterative approach is added to find the values for d and b. These updated values rely on the first-order mean-semi deviation.

Algorithm 4: Modified risk-averse largest-remaining-index algorithm

```
Initialize as empty; $C$ for continuation set, $G$ for Gittins indices set and $S$ for
  stopping set
$ones \leftarrow$ vector of ones
for $all\ arm \in ARMS$ do
    $G(x_1) \leftarrow max_r$
    $C(x_1) \leftarrow argmax_r$
    $S \leftarrow x \in X \backslash x_1$
    for $k\ in\ range(S)$ do
        $P^{(k)}_{y,x} \leftarrow P_{y,x} 1 [x \in C(X_k)]$
        while $\Delta < \epsilon$ do
            $\Delta \leftarrow 0$
            $d_{old}(x) \leftarrow d(x)$
            $b_{old}(x) \leftarrow b(x)$
            for $all\ x \in X$ do
                $dn^{(k)}_{old_{x,y}} \leftarrow d_{old_{y,x}} 1 [x \in C(X_k)]$
                $bn^{(k)}_{old_{x,y}} \leftarrow b_{old_{y,x}} 1 [x \in C(X_k)]$
                $u_d(x) \leftarrow max_a \sum_{y,x} P(y, r|x, \pi(x)) [r + \gamma d_{old}(y)]$
                $u_b(x) \leftarrow max_a \sum_{y,x} P(y, r|x, \pi(x)) [ones + \gamma b_{old}(y)]$
                $v_d(x) = u_d(x) -$
                  $\kappa \sum_{u \in X} P(y|x, \pi(x)) [u(x) - r(x, y, \pi(x)) + \beta dn^{\pi}_{old}(y)]_+, x \in X$
                $v_b(x) =$
                  $u_b(x) - \kappa \sum_{u \in X} P(y|x, \pi(x)) [u(x) - ones + \beta bn^{\pi}_{old}(y)]_+, \quad x \in X$
            $\Delta \leftarrow max(\Delta, d(x) - d_{old}(x), b(x) - b_{old}(x))$
        $V^k \leftarrow d^k / b^k$
        $C(x_k) \leftarrow argmax_{v^k}$
        $G(x_k) \leftarrow max_{v^k}$
        $S \leftarrow x \in X \backslash x_1$
Extract policy
```

In the new heuristic, the algorithm updates the values iteratively until it converges.

3.3 Incorporating Multiple Plays

In MAB with multiple plays, a defined number of actions are taken simultaneously at each timestep. How to incorporate the component of multiple plays differs between the MDP approach and the Gittins algorithms.

3.3.1 Multiple plays in MDP Benchmark

When incorporating multiple plays into the MDP, adjustments need to be made to the transition probabilities and reward structure. Each action must now consider the effects of playing multiple arms simultaneously. Each possible combination of an arbitrary number of arms should now be a possible action. To achieve this, both the reward structure and the probabilities need to be modified.

Initially, the rewards and transitions are generated as if it were a MAB problem with single plays, maintaining the underlying problem's functionality. The transition probabilities are then altered by multiplying the transition probabilities of the different possible combinations of arms, using matrix multiplication to determine the transition probability matrices for the new actions. For example, if the number of simultaneously pulled arms is 2, then pulling arm 1 and 2 together is an action, and the transition probabilities for this action will be the same as multiplying the transition probability matrix of action 1 with the matrix of action 2. Similarly, the rewards are adjusted mathematically by going through the same steps as in a matrix multiplication, but instead of multiplying the values, summing them. However, if any of the values being summed is zero, the sum is also set to zero. With the updated rewards and transition probabilities, value iteration or policy iteration can be applied just as in the MAB problem with single plays.

3.3.2 Multiple Plays in RA and RN

To incorporate multiple plays into the largest-remaining-index algorithm, the algorithm stays the same until the step of extracting the policy. Instead of only saving the action giving the highest value of each state as the policy, the n highest value-giving actions are saved, where n represents the number of multiple plays.

3.4 The Benchmark

Markov Decision Theory was employed to determine the benchmark policy. Both value iteration and policy iteration algorithms were evaluated to find these optimal policies. For policy iteration, two distinct approaches were considered; the standard iterative method and convex optimization. The time complexity of each approach was assessed, and the most efficient algorithm was selected to be used for the computational experiment.

3.4.1 Convex Optimization

For the Convex optimization approach, the following optimization problem is solved:

$$
\begin{aligned}
&Minimize: \sum_{u \in X} v(x) \\
&Subject\ to: \\
&u(x) = \sum_{y \in X} P\left(y|x, \pi^k(x)\right)\left[\beta v(x) - r(x, y, \pi^k(x))\right], \qquad x \in X \\
&v(x) \geq u(x) + \kappa \sum_{y \in X} \rho(x, y) P\left(y|x, \pi^k(x)\right), \qquad x \in X \\
&\rho(x, y) \geq \beta v(y) - r\left(x, y, \pi^k(x) - u(x)\right), \qquad x \in X \\
&\rho(x, y) \geq 0, \qquad x \in X
\end{aligned}
\tag{3.4}
$$

In general, when working with dynamic risk-measures for MDPs, the optimisation problem is convex. However, when utilising first-order mean-semideviation, the problem can be represented as a linear program.

Thus, as seen with the optimization equation above, several restrictions need to be set up before solving the linear programming problem. This requires multiple matrices, vectors and bounds. The time taken for setting up the restrictions of policy iteration convex (policy iteration with convex optimization) grows exponentially to the problem size of the MAB.

3.4.2 Time Complexity

The benchmark algorithm was selected based on the results of an experiment that evaluated the time efficiency of each considered algorithm. The experiment involved running the algorithms on the same environment for that iteration for each of the 100 repetitions. For the MAB problem with two-arm plays and varying complexity, the mean and maximum number of iterations, as well as the mean and maximum runtime, were assessed for three algorithms: value iteration, policy iteration, and policy iteration based on convex optimization. The experiment utilised a random seed of 123 to ensure consistent comparisons. The results of this experiment are presented visually in Table 3.2.

Table 3.2: Number of plays 2, 100 repetitions of value iteration, iterative and convex-optimization-based policy iteration. Discount 0.5.

Number of Plays: 2 & Repetitions 1 & Discount 0.5		Value Iteration				Policy Iteration				Policy Iteration Convex			
MAB Arms | States	Total States	Median # Iteration	Max # Iteration	Time Mean (s)	Time Max (s)	Median # Iteration	Max # Iteration	Time Mean (s)	Time Max (s)	Median # Iteration	Max # Iteration	Time Mean (s)	Time Max (s)
3 Arms 2 States per Arm	8	22	22	0.018	0.032	31	45	0.010	0.020	2	3	0.006	0.013
3 Arms 3 States per Arm	27	22	22	0.057	0.071	33	44	0.035	0.051	2	3	0.650	0.152
3 Arms 4 States per Arm	64	22	22	0.143	0.177	34	45	0.093	0.130	2	3	0.938	1.604
4 Arms 2 States per Arm	64	22	22	0.046	0.067	33	52	0.023	0.042	2	4	0.017	0.035
4 Arms 3 States per Arm	729	22	22	0.236	0.273	37	50	0.130	0.209	2	3	2.412	3.562

Table 3.2. presents the runtime results obtained from performing value iteration, policy iteration (with iterative approach), and policy iteration based on convex optimization for a total of 100 repetitions with a κ value of 1 and a discount value of 0.5. Additionally, Table 3.2 includes the max and median number of iterations required by each algorithm to discover an optimal policy.

Table 3.3: Number of plays 2, 100 repetitions of value iteration, iterative and convex-optimization-based policy iteration Discount 0.9.

Number of Plays: 2 & Repetitions 1 & Discount 0.9		Value Iteration				Policy Iteration				Policy Iteration Convex			
MAB Arms | States	Total States	Median # Iteration	Max # Iteration	Time Mean (s)	Time Max (s)	Median # Iteration	Max # Iteration	Time Mean (s)	Time Max (s)	Median # Iteration	Max # Iteration	Time Mean (s)	Time Max (s)
3 Arms 2 States per Arm	8	133	135	0.107	0.182	176	316	0.048	0.084	2	3	0.007	0.015
3 Arms 3 States per Arm	27	132	136	0.345	0.430	164	298	0.155	0.300	2	3	0.062	0.110
3 Arms 4 States per Arm	64	131	135	0.827	0.935	172	248	0.378	0.565	2	3	1.010	1.589
4 Arms 2 States per Arm	64	134	137	0.273	0.319	184	272	0.104	0.162	2	3	0.018	0.036
4 Arms 3 States per Arm	729	133	136	1.390	1.548	180	323	0.550	0.936	2	4	2.482	4.576

Table 3.3 shows how much time and number of iterations is needed to complete value iteration, iterative policy iteration and convex-optimization-based policy iteration with a number of 100 repetitions with a κ value of 1 and a discount factor of 0.9.

Seen in both Table 3.2 and 3.3, increasing the complexity of the MAB also increases the total runtime. The varying algorithms have different numbers of iterations needed to find optimal policies. A higher discount (see Table 3.3) results in a larger number of iterations needed before finding optimal policies (see Table 3.2). However, the total number of states does not seem to affect the number of iterations.

For the smallest MAB example, with three arms and two states per arm, convex-optimization-based policy iteration is the fastest algorithm. This changes drastically when increasing the complexity of the MAB, making convex-optimization-based policy iteration considerably slower than its counterparts. For larger examples, policy iteration is the fastest, with value iteration being around 40% slower.

The reason behind convex-optimization-based policy iteration taking the longest for larger problems is the way the algorithm calculates the states values. As mentioned previously, several matrices need to be created to solve the linear programming problem. The size of these matrices grows exponentially as the problem size of the MAB increases. The focus of this thesis is to study a MAB with three arms and four states per arm while applying the RA heuristic and comparing it to one of the benchmarks presented in tables 5 and 6. The lowest runtime of this specific MAB was achieved by policy iteration, which is hereafter chosen as the benchmark algorithm for this thesis.

3.5 Value Function from Gittins

After performing the RA or RN, a value has been calculated for each state in each arm. According to the definition of the Gittins index, the optimal action is to pull the arm whose current state possesses the highest value. To compare these values with the value function obtained through policy iteration, two steps are required. First, a policy of the RA/RN needs to be arranged in the same structure as of the states in the MDP. This means that if a state in the MDP is a combination of for example state 1 of arm 1, and state 4 of arm 2, choosing the arm which corresponds to the highest valued state of these two options, should be seen as the RA/RN-policy for this combined MDP-state.

Secondly, the correctly arranged policy is given as an input to the policy evaluation step of the policy iteration. The policy evaluation outputs a value function based on the RA or RN policy which can be compared to the value function obtained from the policy iteration algorithm.

3.6 Computational Experiments

To evaluate the RA, both its efficacy in finding optimal policies and its time complexity were assessed. The structure of the experiments is a generalisation of the work of Malekipirbazari and Cavus [13]. The experiments were, just as the other algorithms, coded in Python 3.10.

To evaluate the optimality of the RA, 1,000 test instances were generated for each setup of the problem formulation. These experiments considered a problem of a three-armed bandit with four states per arm and two arms played simultaneously. The transition probabilities of each arm were randomly drawn from a uniform distribution between 0 and 1. The resulting transition matrix was normalised such that the sum of the probabilities of each row adds up to 1. The rewards for the first experiment are drawn from a truncated normal distribution with a random mean value generated from a uniform distribution between 5 and 6. The truncation ensures the rewards were non-negative. The rewards for the second experiment were drawn from an exponential distribution with a random β value generated from a uniform distribution between 5 and 6.

For both experimental set ups, three policies were assessed for each test instance. These consist of an optimal policy, one risk-averse index policy (RA) and one risk-neutral index policy (RN). The experiments are conducted with standard deviations $\sigma \in \{0.01, 0.5, 1, 2\}$, discount factors $\gamma \in \{0.5, 0.75, 0.9\}$. As risk measures first-order mean-semideviation, with $\kappa \in \{0.0, 0.25, 0.50, 0.75, 1\}$ were used. A visual presentation of the varying variables can be seen in Table 3.3.

Table 3.4: Variable names and descriptions

Variable name	**Symbol**	**Values**	**Description**
Number of arms		4	Number of arms within the MAB structure
Number of states per arm		3	Number of states for each arm within the MAB structure
Distribution setup		Normal and exponential	Type of distribution for initialising the rewards
Standard deviation	σ	0.01, 0.5, 1, 2	The standard deviation for the distributions from which rewards are drawn
Discount factor	γ	0.5, 0.75, 0.9	The discount factors which determine the importance of immediate/future rewards
First-Order Mean-Semideviation	κ	0, 0.25, 0.50, 0.75, 1	A weight that determines the level of risk-aversion incorporated in the heuristic

For each test instance RA and RN policies are compared based on the performance measure: suboptimality percentage. The suboptimality percentage of policy $\pi \in \{RA, RN\}$ for each state $x \in X$ is computed as:

$$100 \times \frac{R(x) - R^{\pi}(x)}{R(x)} = \% \text{ Optimality Gap} \tag{3.5}$$

where $R^{\pi}(x)$ denotes the value of a state under policy π and $R(x)$ denotes the value of a state under the optimal policy. The equation above computes the difference between the value of RA or RN and the optimal value obtained through the benchmark algorithm. A zero optimality gap implies that the two policies produce the same optimal value, whereas a non-zero optimality gap represents the deviation between the two values obtained from the two policies. For each test instance, the maximum and the median suboptimality percentage are calculated. The maximum and the mean of the maximum suboptimality percentage over all the 1000 test instances was calculated as well as the maximum and the mean of the median suboptimality percentages.

Table 3.5: Evaluation measures and descriptions

Evaluation Measure	**Description**
Max max	Maximum of maximums, which takes the maximum value from all maximum values when compared to the benchmark
Mean max	Average of maximum, which takes the mean of all maximum values when compared to the benchmark
Max median	Maximum of median, which takes the max value from all the median values when compared to the benchmark
Mean median	Average of median, which takes the mean of all median values when compared to the benchmark

To evaluate the time complexity, the runtime of the RA heuristic was compared to the risk-averse value iteration algorithm. Both algorithms were executed for 10 repetitions.

4

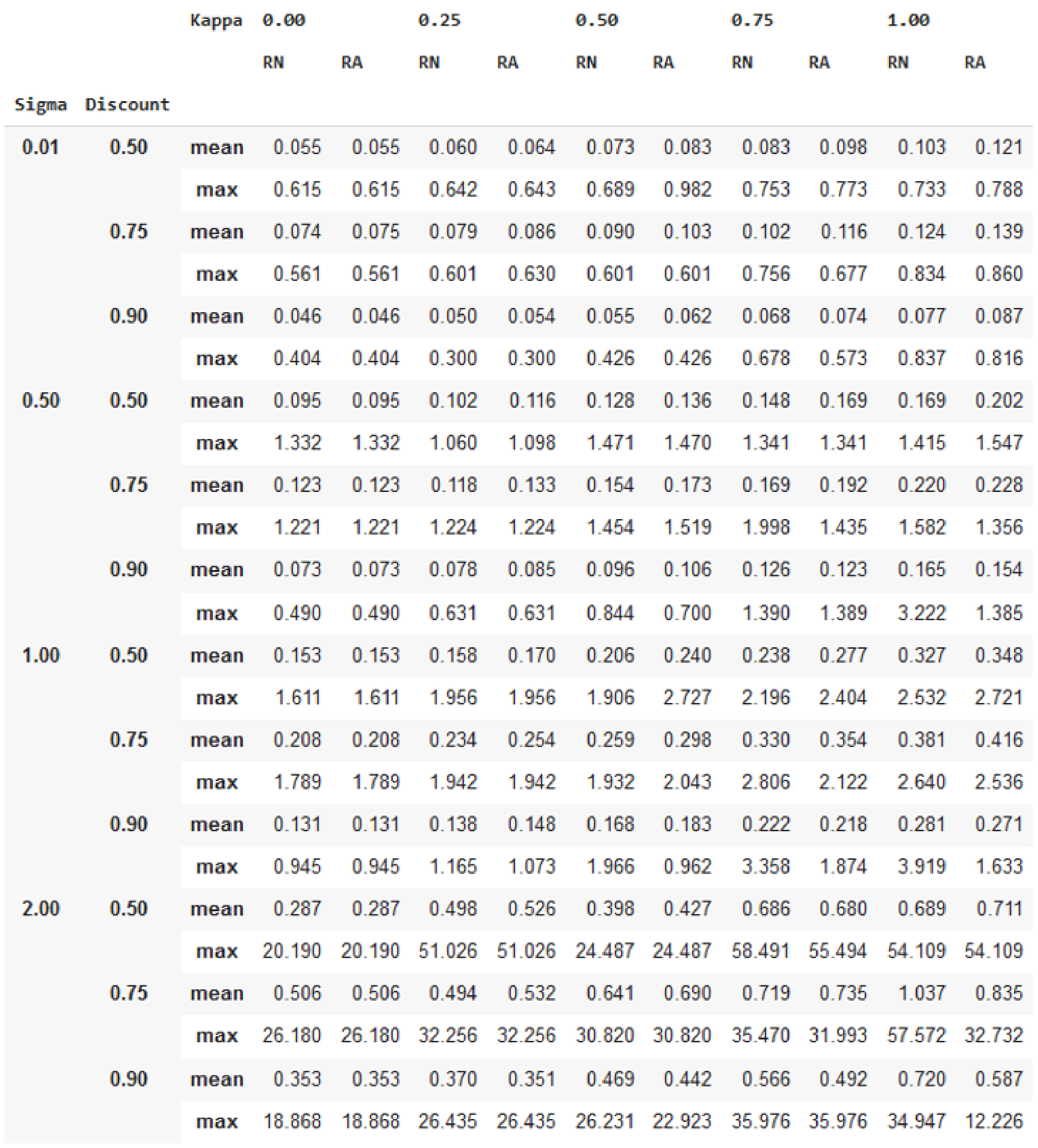
Results

This section discusses the results obtained for the RA and RN in comparison to the benchmark (the optimal policy). There have been two big experiments conducted to evaluate the quality of the RAs performance, on MABs with three arms and four states per arm. One of the experiments uses normal reward distributions and the other uses exponential distributions. Both experiments are analysed separately and in comparison. Lastly, results of a time complexity experiment are presented.

4.1 Efficacy Experiment

The efficacy of RA and RN is measured by the optimality gap between its values and the values of the benchmark. All the experiments in this section are conducted with $\sigma = \{0.01, 0.5, 1, 2\}$, $\gamma = \{0.5, 0.75, 0.90\}$, and $\kappa = \{0, 0.25, 0.5, 0.75, 1\}$.

Table 4.1: Statistics of maximum suboptimality percentages, normal reward distribution, varying κ, σ and γ



		Kappa	0.00		0.25		0.50		0.75		1.00	
			RN	RA	RN	RA	RN	RA	RN	RA	RN	RA
Sigma	Discount											
0.01	0.50	mean	0.055	0.055	0.060	0.064	0.073	0.083	0.083	0.098	0.103	0.121
		max	0.615	0.615	0.642	0.643	0.689	0.982	0.753	0.773	0.733	0.788
	0.75	mean	0.074	0.075	0.079	0.086	0.090	0.103	0.102	0.116	0.124	0.139
		max	0.561	0.561	0.601	0.630	0.601	0.601	0.756	0.677	0.834	0.860
	0.90	mean	0.046	0.046	0.050	0.054	0.055	0.062	0.068	0.074	0.077	0.087
		max	0.404	0.404	0.300	0.300	0.426	0.426	0.678	0.573	0.837	0.816
0.50	0.50	mean	0.095	0.095	0.102	0.116	0.128	0.136	0.148	0.169	0.169	0.202
		max	1.332	1.332	1.060	1.098	1.471	1.470	1.341	1.341	1.415	1.547
	0.75	mean	0.123	0.123	0.118	0.133	0.154	0.173	0.169	0.192	0.220	0.228
		max	1.221	1.221	1.224	1.224	1.454	1.519	1.998	1.435	1.582	1.356
	0.90	mean	0.073	0.073	0.078	0.085	0.096	0.106	0.126	0.123	0.165	0.154
		max	0.490	0.490	0.631	0.631	0.844	0.700	1.390	1.389	3.222	1.385
1.00	0.50	mean	0.153	0.153	0.158	0.170	0.206	0.240	0.238	0.277	0.327	0.348
		max	1.611	1.611	1.956	1.956	1.906	2.727	2.196	2.404	2.532	2.721
	0.75	mean	0.208	0.208	0.234	0.254	0.259	0.298	0.330	0.354	0.381	0.416
		max	1.789	1.789	1.942	1.942	1.932	2.043	2.806	2.122	2.640	2.536
	0.90	mean	0.131	0.131	0.138	0.148	0.168	0.183	0.222	0.218	0.281	0.271
		max	0.945	0.945	1.165	1.073	1.966	0.962	3.358	1.874	3.919	1.633
2.00	0.50	mean	0.287	0.287	0.498	0.526	0.398	0.427	0.686	0.680	0.689	0.711
		max	20.190	20.190	51.026	51.026	24.487	24.487	58.491	55.494	54.109	54.109
	0.75	mean	0.506	0.506	0.494	0.532	0.641	0.690	0.719	0.735	1.037	0.835
		max	26.180	26.180	32.256	32.256	30.820	30.820	35.470	31.993	57.572	32.732
	0.90	mean	0.353	0.353	0.370	0.351	0.469	0.442	0.566	0.492	0.720	0.587
		max	18.868	18.868	26.435	26.435	26.231	22.923	35.976	35.976	34.947	12.226

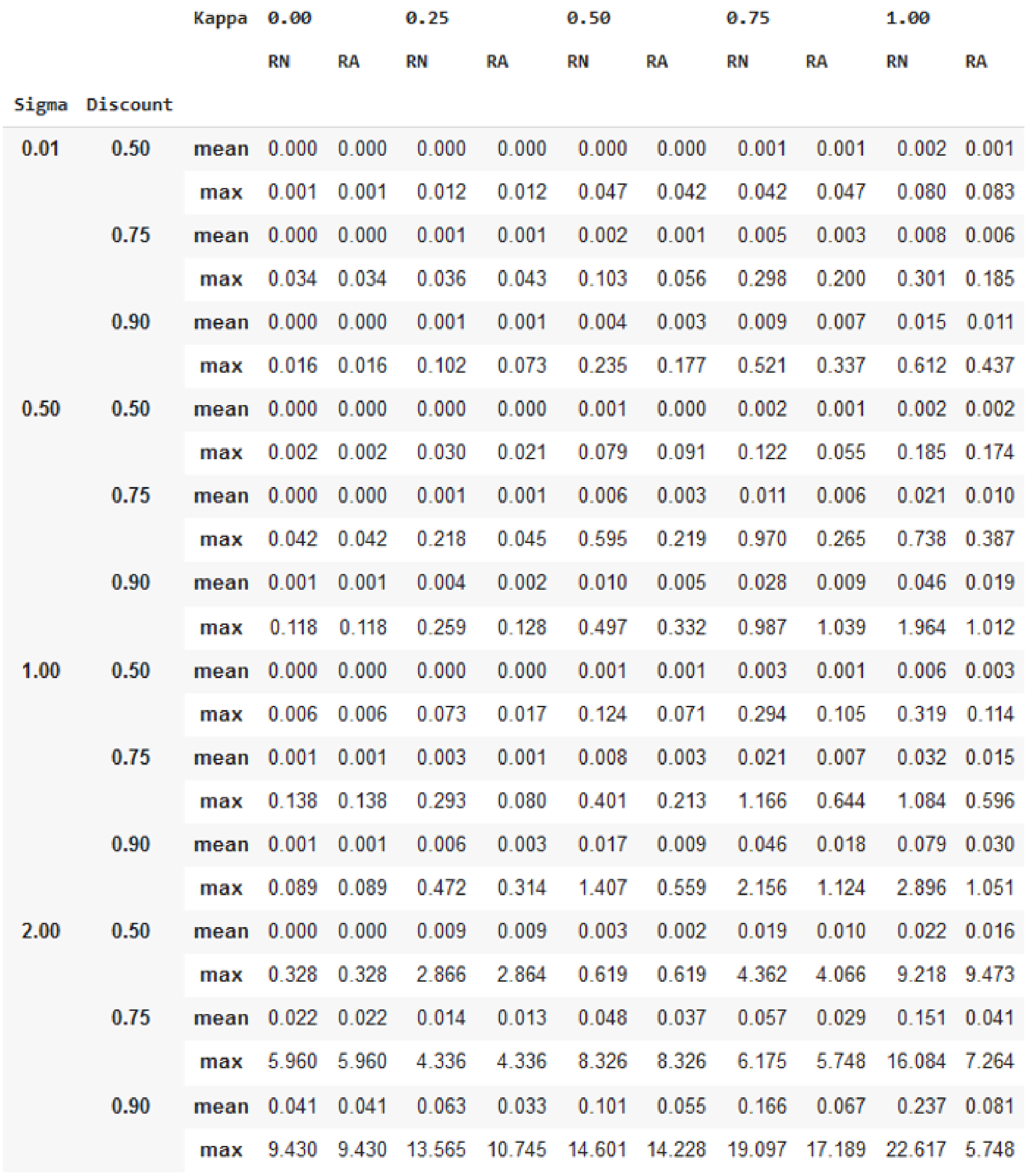
Table 4.1 summarises the statistics of maximum suboptimality percentages when generating the rewards from a normal distribution. It shows both mean and maximum of maximum suboptimality percentages for both RA and RN. The mean values for both the RN and RA, achieved near optimal policies. The highest mean value for RA is 0.835%, which is paired with RNs 1.037% at a σ of 2, κ of 1 and γ of 0.75.

When considering the maximum, there were instances of higher error. The highest optimality gaps were primarily observed for the highest values of σ. This indicates that σ plays a noteworthy role in increasing the variability of the policies' optimality and potential deviations from the benchmark. The highest values for maximum suboptimality percentage for RN and RA are 58.49% and 55.49%, respectively, seen at discount 0.5, σ 2, and a κ of 0.75. Nevertheless, albeit having these high values of max RN and RA, the mean values for the same parameters are 0.686 and 0.680 respectively. With 1,000 instances in our experiment, this indicates that there are a few high outliers where RN and RA result poorly against the benchmark.

In addition, Table 4.1 reveals a trend where increasing σ in general leads to higher suboptimality percentages for both the RN and RA policies. Similarly, as κ increases, more risk-aversion is incorporated, increasing the maximum suboptimality percentages for both policies. The values of RA and RN in relationship to γ on the other hand, demonstrates a concave pattern. Its concave in the sense that a low or high discount value has a lower optimality gap compared to the middle case. It is worth noting that when κ equals 0, values for RN and RA are identical, as expected in a risk-neutral scenario. Overall, Table 4.1 shows that the RA performed slightly better than RN.

When comparing the maximum optimality gaps observed in Table 4.1, with each respective mean optimality gaps of the same set of parameters, the importance of investigating a median optimality gap arises. In some outlying cases, the max optimality gap was up to 10 times higher than the average. This indicates potential inflation caused by a few high max optimality gaps affecting the overall average of the values.

Table 4.2: Statistics of median suboptimality percentages, normal reward distribution, varying κ, σ and γ



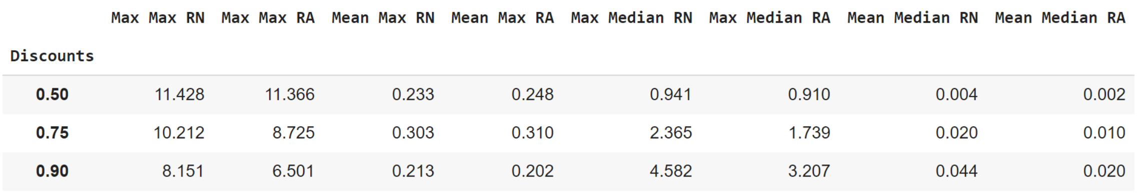
		Kappa	0.00		0.25		0.50		0.75		1.00	
			RN	RA	RN	RA	RN	RA	RN	RA	RN	RA
Sigma	Discount											
0.01	0.50	mean	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.002	0.001
		max	0.001	0.001	0.012	0.012	0.047	0.042	0.042	0.047	0.080	0.083
	0.75	mean	0.000	0.000	0.001	0.001	0.002	0.001	0.005	0.003	0.008	0.006
		max	0.034	0.034	0.036	0.043	0.103	0.056	0.298	0.200	0.301	0.185
	0.90	mean	0.000	0.000	0.001	0.001	0.004	0.003	0.009	0.007	0.015	0.011
		max	0.016	0.016	0.102	0.073	0.235	0.177	0.521	0.337	0.612	0.437
0.50	0.50	mean	0.000	0.000	0.000	0.000	0.001	0.000	0.002	0.001	0.002	0.002
		max	0.002	0.002	0.030	0.021	0.079	0.091	0.122	0.055	0.185	0.174
	0.75	mean	0.000	0.000	0.001	0.001	0.006	0.003	0.011	0.006	0.021	0.010
		max	0.042	0.042	0.218	0.045	0.595	0.219	0.970	0.265	0.738	0.387
	0.90	mean	0.001	0.001	0.004	0.002	0.010	0.005	0.028	0.009	0.046	0.019
		max	0.118	0.118	0.259	0.128	0.497	0.332	0.987	1.039	1.964	1.012
1.00	0.50	mean	0.000	0.000	0.000	0.000	0.001	0.001	0.003	0.001	0.006	0.003
		max	0.006	0.006	0.073	0.017	0.124	0.071	0.294	0.105	0.319	0.114
	0.75	mean	0.001	0.001	0.003	0.001	0.008	0.003	0.021	0.007	0.032	0.015
		max	0.138	0.138	0.293	0.080	0.401	0.213	1.166	0.644	1.084	0.596
	0.90	mean	0.001	0.001	0.006	0.003	0.017	0.009	0.046	0.018	0.079	0.030
		max	0.089	0.089	0.472	0.314	1.407	0.559	2.156	1.124	2.896	1.051
2.00	0.50	mean	0.000	0.000	0.009	0.009	0.003	0.002	0.019	0.010	0.022	0.016
		max	0.328	0.328	2.866	2.864	0.619	0.619	4.362	4.066	9.218	9.473
	0.75	mean	0.022	0.022	0.014	0.013	0.048	0.037	0.057	0.029	0.151	0.041
		max	5.960	5.960	4.336	4.336	8.326	8.326	6.175	5.748	16.084	7.264
	0.90	mean	0.041	0.041	0.063	0.033	0.101	0.055	0.166	0.067	0.237	0.081
		max	9.430	9.430	13.565	10.745	14.601	14.228	19.097	17.189	22.617	5.748

Table 4.2 displays the mean of medians and max of medians from the same underlying values displayed on Table 4.1. Similarly to Table 4.2, an increase in σ and κ respectively leads to an increase in the optimality gap. The behaviour observed for varying γ values does not follow the same concave pattern as seen in Table 4.1.

The advantage of the RA algorithm becomes more pronounced in Table 4.2 compared to Table 4.1, where statistics of median suboptimality percentages are presented. The maximum median value for suboptimality percentage for RN is 22.62%, which occurs at a κ of 1, σ of 2 and γ of 0.9. The RA for these same parameters has a maximum median value of 5.748%. As for the maximum median of RA, this occurs at a κ of 0.75, σ of 2, and γ of 0.9 resulting in a suboptimality percentage of 17.19% compared to an RN of 19.10% for the same settings. Overall, it is evident that RN rises more than RA, with an increase in κ. However, there are a few exceptions, for example at κ equal 1, γ of 0.5 and a σ of 2 or 0.01, RA has a value of 9.47% and RN a value of 9.22%, and RA a value of 0.08% and RN 0.08% respectively.

To assess the effect of the γ over all the values presented in Table 4.1 and 4.2, the means for all the presented evaluation measures, grouped by the γ, is presented. This shows how well the RA and RN performs in general, disregarding the varying σ and κ.

Table 4.3: Evaluation measures with regard to γ for normal reward distribution.



	Max Max RN	Max Max RA	Mean Max RN	Mean Max RA	Max Median RN	Max Median RA	Mean Median RN	Mean Median RA
Discounts								
0.50	11.428	11.366	0.233	0.248	0.941	0.910	0.004	0.002
0.75	10.212	8.725	0.303	0.310	2.365	1.739	0.020	0.010
0.90	8.151	6.501	0.213	0.202	4.582	3.207	0.044	0.020

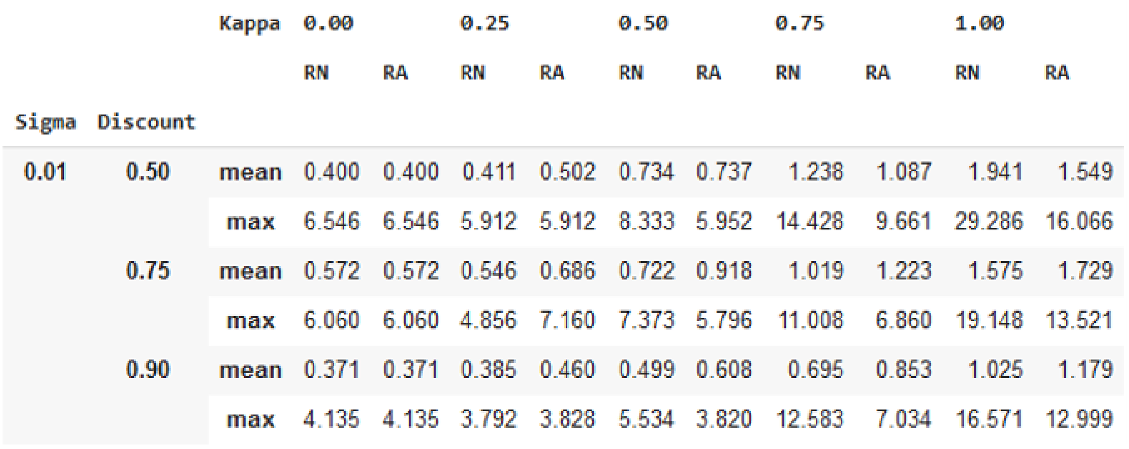
Table 4.3 categorises the average of different values of RA and RN by the γ . For γ values of 0.5 and 0.75, the RA algorithm exhibits a lower suboptimality gap compared to RN for the maximum of maximums, the maximum of medians, and the mean of medians. However, the RA algorithm shows a higher mean maximum value compared to RN for the mean of maximums. This indicates that for specific γ values, the RA algorithm tends to achieve higher maximum values more frequently than RN whilst still not reaching the most extreme maximums. Interestingly, this is not the case for the median values, for which RA outperforms RN, and the margins grow as the discount increases.

For γ values of 0.5 and 0.75 the mean max RN performed 6.53%, 2.15% better than the mean median RA, respectively, whilst for γ of 0.9, the RA performed 5.27% better than the RN. For γ values of 0.5, 0.75 and 0.9 the mean median RA performed 48.35%, 103.75% and 120.86% better than the mean median RN, respectively. As for the max median, the suboptimality gap in RA compared to RN is not as large as seen in the mean median, but has a large increase as γ increases. When γ has values of 0.5, 0.75 and 0.9 the max median RA performs 3.39%, 35.96% and 42.86% better than the mean median RN, respectively. Lastly, comparing the max max RN and RA, when γ has values of 0.5, 0.75 and 0.9 the RA performs 0.55%, 17.04% and 25.38% better than the RN, respectively.

The γ has a different effect on each of the evaluation measures. The RA performs increasingly better with respect to the performance of the RN, when γ rises. Both RN and RA have a concave relationship to γ. The max max has a different pattern than the other evaluation measures, where an increase in γ leads to a decrease in suboptimality. For the remaining evaluations measures, an increase in γ increases the suboptimality of both RA and RN from the benchmark, yet also improves the performance of RA compared to the RN.

Overall, these findings indicate that the RA heuristic is an improvement on the RN for risk-averse settings. RA is overall better than RN at avoiding the most extreme deviations from the benchmark and at the same time outperforms the RN for the median values.

Table 4.4: Statistics of maximum suboptimality percentages, exponential reward distribution, varying κ and γ

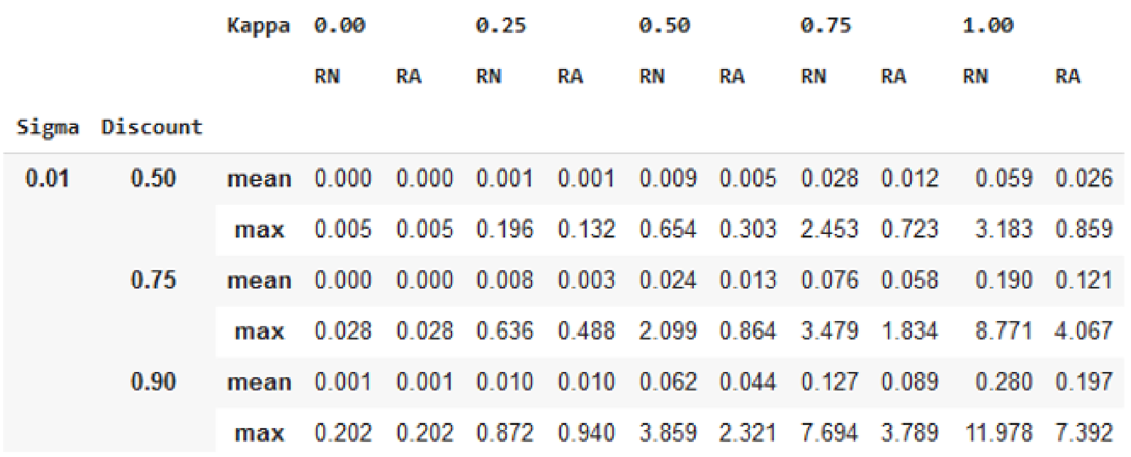


		Kappa	0.00		0.25		0.50		0.75		1.00	
			RN	RA	RN	RA	RN	RA	RN	RA	RN	RA
Sigma	Discount											
0.01	0.50	mean	0.400	0.400	0.411	0.502	0.734	0.737	1.238	1.087	1.941	1.549
		max	6.546	6.546	5.912	5.912	8.333	5.952	14.428	9.661	29.286	16.066
	0.75	mean	0.572	0.572	0.546	0.686	0.722	0.918	1.019	1.223	1.575	1.729
		max	6.060	6.060	4.856	7.160	7.373	5.796	11.008	6.860	19.148	13.521
	0.90	mean	0.371	0.371	0.385	0.460	0.499	0.608	0.695	0.853	1.025	1.179
		max	4.135	4.135	3.792	3.828	5.534	3.820	12.583	7.034	16.571	12.999

Table 4.4 shows what effect changing reward distributions from normal to exponential has on the results. In this table, when κ is less than 0.5, the RA performs equal to or worse than RN throughout all parameters of γ. The highest maximum value of RN and RA is 29.286 and 16.066 respectively. This happens for a γ value of 0.5 and κ of 1. An increasing κ has the affect of increasing the suboptimality percentages, while an increasing γ has the opposite effect. However, there are some slight inconsistencies to these patterns. The inconsistencies from the affect of γ occur exactly at κ 0.25 and 0.75. When it comes to the affect of κ, the pattern for the means is clear while for the maximums it is more subtle.

For the mean values, the RA take a concave pattern in relationship to the γ throughout all values of κ. This is not the case for max values of RA, which decrease as discount increases, with an exception for κ of 0.25.

Table 4.5: Statistics of median suboptimality percentages, exponential reward distribution, varying κ and γ



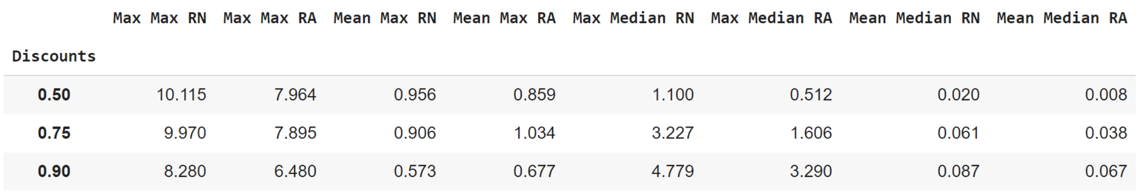
		Kappa	0.00		0.25		0.50		0.75		1.00	
			RN	RA	RN	RA	RN	RA	RN	RA	RN	RA
Sigma	Discount											
0.01	0.50	mean	0.000	0.000	0.001	0.001	0.009	0.005	0.028	0.012	0.059	0.026
		max	0.005	0.005	0.196	0.132	0.654	0.303	2.453	0.723	3.183	0.859
	0.75	mean	0.000	0.000	0.008	0.003	0.024	0.013	0.076	0.058	0.190	0.121
		max	0.028	0.028	0.636	0.488	2.099	0.864	3.479	1.834	8.771	4.067
	0.90	mean	0.001	0.001	0.010	0.010	0.062	0.044	0.127	0.089	0.280	0.197
		max	0.202	0.202	0.872	0.940	3.859	2.321	7.694	3.789	11.978	7.392

Table 4.5 shows the median values of the exponential reward distributions. Comparing the suboptimality percentage of the RA to the RN in this table, the RA performed equal to or better throughout the different values of κ and γ.

The highest value of RN and RA is located at the highest κ and γ, with the RN being 11.978 and the RA being 7.392. The best performing case of RA in comparison to RN occurred at κ of 1 and γ of 0.5, where the RA performs 270.55% better than the RN. RA performs better in all cases compared to the RN with the exception of κ equal to 0.25 and γ equals to 0.9.

There is a clear pattern throughout the table, where the values of RA and RN rise as the value of κ or γ increase.

Table 4.6: Evaluation measures with regard to γ for exponential reward distribution.



	Max Max RN	Max Max RA	Mean Max RN	Mean Max RA	Max Median RN	Max Median RA	Mean Median RN	Mean Median RA
Discounts								
0.50	10.115	7.964	0.956	0.859	1.100	0.512	0.020	0.008
0.75	9.970	7.895	0.906	1.034	3.227	1.606	0.061	0.038
0.90	8.280	6.480	0.573	0.677	4.779	3.290	0.087	0.067

Table 4.6 is a replication of Table 4.3, but for exponential reward distributions. Similarly to Table 4.3, the RA performs better than the RN on all evaluation measures with the exception of mean max. For γ values of 0.75 and 0.9 the mean max RN performed 14.09% and 18.16% better than the mean max RA, respectively. For γ of 0.5, the RA performed 11.21% better than the RN.

As for the mean median, when γ has values of 0.5, 0.75 and 0.9 the RA performs 137.83%, 62.96% and 31.62% better than the RN, respectively. The max median RA performs 114.84%, 100.89% and 45.29% better than the RN with respective γ values of 0.5, 0.75 and 0.9. Lastly, for the max max when γ has values of 0.5, 0.75 and 0.9 the RA performs 27.01%, 26.29% and 27.82% better than the RN, respectively.

An increase in γ for max median and mean median increases the suboptimality gap. While an increase in γ for the mean max, results in RN decreasing in value while RA takes a convex pattern. Once more, all the median values of RA are outperforming RN.

4.2 Conclusions of Efficacy Experiment

Table 4.3 and Table 4.6 summaries that in most cases RA outperforms RN. The exception for this arises in the mean of maximums for both the normal and exponential distribution.

For normal distributions, the maximum suboptimality percentages shown by Table 4.1 reveals a few cases for when the RN only slightly outperforms RA, with these divergences occurring at different values of κ, σ and γ. As for the median suboptimality, the improved performance of RA over the RN is substantial. Seen in Table 4.2 at variables κ of 1, σ of 2 and 0.9, the highest value of RN was 22.62%, while RA for the same parameters was 5.75%, a 293.39% improved performance by the RA over the RN. Table 4.3 presents the average of all evaluation measures when categorised by the γ. It shows that the RA does not perform as well as the RN when looking at the mean of maximums, yet it performs better on the other evaluation measures.

The pattern seen for the normal distribution is dependent on γ. An increase in γ leads to a concave pattern on the mean max suboptimality values for both RN and RA, whilst an increasing γ decreases the suboptimality gap for mean max with this decrease occurring faster within the RA than the RN. For the max median and mean median, an increase in γ leads to an increased optimality gap. Importantly, an increase in γ improves the performance of RA against RN within all evaluation measures.

As for the exponential distribution, 4.5 illustrates a consistent pattern where the median suboptimality gaps increases for both RA and RN as the values of κ or γ rise. The same pattern of the affect of κ is visible in 4.4 over maximums while γ has the opposite effect for the values of maximums than for medians.

Table 4.6 and Table 4.3, which both summarises the results categorised by γ, shows the same patterns, for maximum of maximums as well as mean and maximum of medians, the RA outperforms RN for each discount value. For the mean of maximum the pattern is inconsistent, with RN performing better for some γ value but not others.

When comparing the RAs performance over the RN, the mean median RA decreases as γ increases. Specifically, when γ has values of 0.5, 0.75 and 0.9 the mean median RA performs 137.83%, 62.96% and 31.62% better than the RN, respectively. A similar pattern appears for the max median RA, where the RA performs 114.84%, 100.89% and 45.29% better than the RN for the γ values of 0.5, 0.75 and 0.9. However, this pattern does not stay consistent for the maximum of maximums of RA As γ has values of 0.5, 0.75 and 0.9 the RA performs 27.01%, 26.29% and 27.82%

better than the RN, showing no noticeable improvement from the RN as γ increases. Furthermore, in the maximum of maximums, the increase in γ leads to a smaller suboptimality gap, meaning that max max RA and RN both improve.

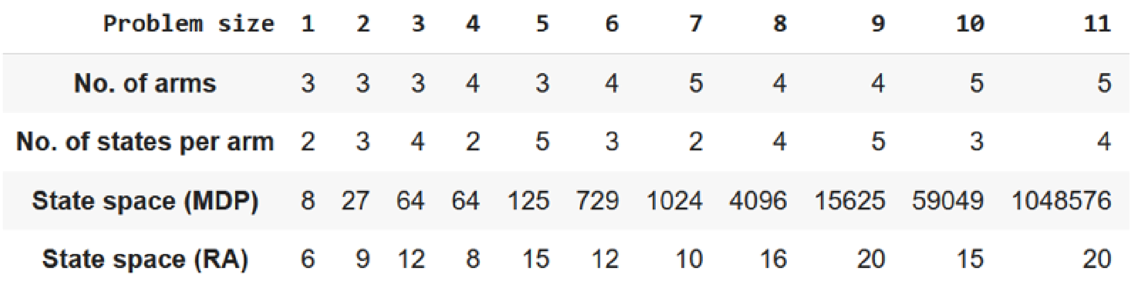
One major difference between the exponential distribution and the normal distribution is that for higher γ values, the RA for the normal distributions performs relatively better against the RN than for lower γ. For exponential rewards, when the RAs effectiveness against the RN in percentage is calculated, a differentiating pattern for different evaluation measures emerges than as seen in the normal distribution. This is evident in the resulting percentage differences of RA against RN, where the maximum of maximums remains constant and mean median decreases in regards to γ respectively. A similarity between both distributions is that RA performs better in almost all cases with the exception of mean maximums, where certain γ values affect RAs performance more negatively than the RN.

In conclusion, the RA performs better than the RN in most scenarios. There are a few exceptions, especially when assessing mean of maximums. For the normal reward distribution, an increase in the γ value will raise the suboptimality gaps overall, whilst also improving the relative performance of RA when compared to RN. An exponential reward distribution on the other hand, will make the RA perform decreasingly well in comparison to the RN as γ increases, for all the evaluation measures except for the maximum of maximums.

4.3 Time Complexity Experiment

This experiment aimed to visualise the time efficiency of the RA algorithm in comparison to solving the MDP optimally. The experiment involved measuring the CPU time of the algorithms for 11 different problem sizes. The problem sizes are used as an index where the number of arms and the number of states per arm, leads to different total state spaces in the MDP and different total state spaces in the RA policy.

Table 4.7: Problem size table values. Time complexity of RA and policy iteration.



Problem size	1	2	3	4	5	6	7	8	9	10	11
No. of arms	3	3	3	4	3	4	5	4	4	5	5
No. of states per arm	2	3	4	2	5	3	2	4	5	3	4
State space (MDP)	8	27	64	64	125	729	1024	4096	15625	59049	1048576
State space (RA)	6	9	12	8	15	12	10	16	20	15	20

Table 4.7 displays the problem size with its corresponding number of arms, states per arm, the total number of states in the MDP for that corresponding set up, and lastly the state space for the RA.

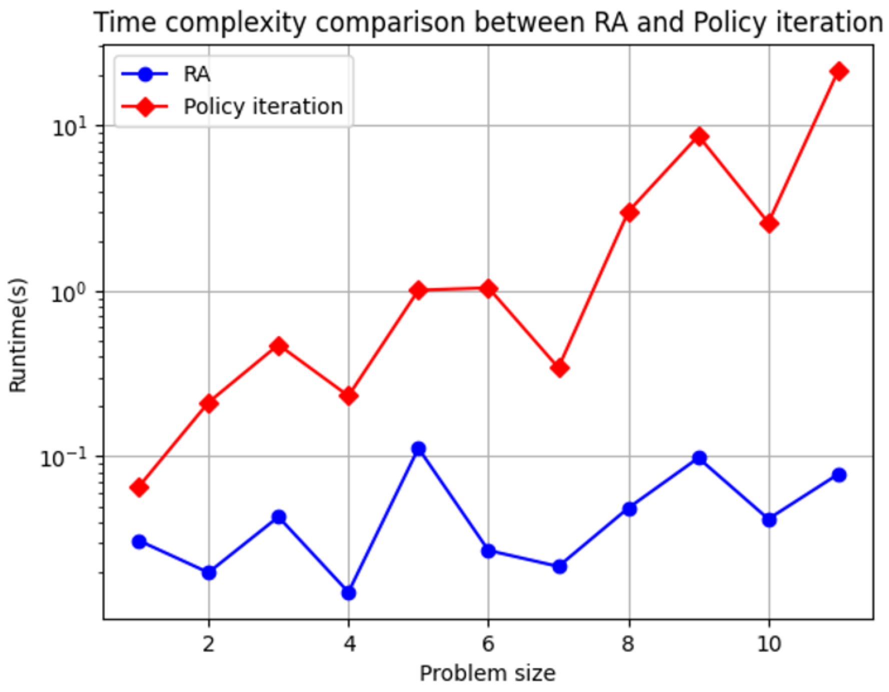


Figure 4.1: Time complexity comparison between RA and Policy interation.

Figure 4.1 presents the results of the time experiment, with the CPU time displayed on a logarithmic scale to facilitate better comparison.

The results show that the computation time of the RA is substantially lower than that of policy iteration. As the problem size increases, the computation time for the RA appears to grow linearly, while the computation time for solving the risk-averse MDP optimally exhibits exponential growth. This demonstrates the superior time efficiency of the RA algorithm compared to the optimal solution.

The runtime of the RA is consistent with the state space seen in Table 4.7. An increase in the total number of states leads to an increase of log time. The time complexity of the policy iteration does not only increase with the state space (MDP). It seems to increase with the number of arms and states, with a higher emphasis on the number of states. Taking problem sizes 6 and 7 as examples as seen in Table 4.7, problem size 6 has 4 arms, 3 states per arm leading to a total of 729 state spaces while problem size 7 has 5 arms, 2 states per arm and 1024 state spaces. Figure 4.1 shows that problem size 6 takes a longer time for the policy iteration than the size 7. This pattern is consistent throughout the graph.

5 Discussion

In this thesis, a risk-averse MAB with multiple plays is analysed. The model aims to help decision makers with different extents of risk-aversion to make optimal risk-adjusted actions. The proposed and evaluated algorithm, RA, demonstrates superior performance compared to RN in the majority of tested cases, particularly when considering median deviations from the optimal value functions or the absolute maximum deviations. Furthermore, the time complexity of computing the RA is less than the benchmark, especially so in larger MAB settings.

This thesis can be seen as an extension of the work by Malekipirbazari and Cavus [13]. They showed how their RA for single plays outperformed their RN for single plays in a risk-averse setting. The findings of this thesis confirm the superiority of RA over RN also for multiple plays, although less apparent. The reason for the RA of this thesis performing worse is probably due to the incorporation of multiple plays which adds complexity to the problem. When multiple plays are incorporated, the RN is no longer guaranteed to be optimal even for the risk-neutral case.

5.1 Limitations

A drawback of this thesis is the lack of a mathematical proof to demonstrate the effectiveness of the RA. Instead, the thesis relies on experimental comparisons to showcase the efficiency of the algorithms.While the current approach effectively demonstrates the heuristic's effectiveness within the specific tested environment, it may not yield consistent results in different contexts, such as with varying sizes of the MAB or different reward distributions, especially when the arms have distinct reward distributions. With a more generous time frame allocated to the thesis project, it would have been possible to conduct a greater number of experiments, to test various settings and gain a better understanding of the algorithm's performance under different conditions.

Furthermore, in real-life situations, time constraints are often present, which can impact the algorithm's performance. Assuming an unlimited time horizon as in this thesis may not hold in practical applications, and the algorithm's effectiveness may be somewhat compromised when operating within finite time frames.

5.2 Future Work

There are several potential extensions that can be explored based on this thesis. One possibility is to incorporate different risk measures, such as the mean-CVaR, into the problem formulation. This would provide a more comprehensive assessment of risk in decision-making.

Furthermore, there are numerous plausible extensions to the risk-averse MAB problem with multiple plays. For instance, the MAB could be extended to include restlessness, where the underlying probabilities and rewards change over time. Additionally, linkages between the arms could be incorporated into the model.

An interesting extension would involve applying the model to practical scenarios such as clinical trials, where it is desirable to select a variable number of treatments. This would require the inclusion of dummy arms that always yield a reward of 0. The number of dummy arms would be one less than the total number of possible treatments. This setup would enable the achievement of optimality regardless of the number of treatments selected. This means the number of multiple plays would not have to be a fixed number for each action as in this thesis.

Another practical setting in which the RA would be interesting to study is the problem of ambulance redeployment. A research study could explore RAs potential in reducing fatal injuries during ambulance transportation. This would provide insights into the real-world effectiveness of the algorithm.

6 Conclusion

This thesis presents an analysis of a risk-averse multi-armed bandit problem with multiple plays. The proposed algorithm, RA, demonstrates superior performance compared to RN in the majority of tested cases, particularly when considering median deviations from the optimal value functions or absolute maximum deviations. In the most extreme cases of the normal distribution experiment, RA performed 120.86% better than the RN. In the most extreme case for the exponential distribution, RA performed 270.55% better than the RN. The overall obtained results extend on the findings of relevant literature that RA outperforms RN in a risk-averse setting for single plays.

Moving forward, future research can explore extensions such as incorporating different risk measures, restlessness, and linkages between arms into the risk-averse MAB with multiple plays. Practical applications, such as clinical trials with varying number of treatment selection and the problem of ambulance redeployment, offer opportunities to assess the algorithm's real-world effectiveness. These extensions would provide valuable insights into the algorithm's applicability and performance in diverse scenarios.

Bibliography

[1] D. Bouneffouf and I. Rish, "A survey on practical applications of multi-armed and contextual bandits," *CoRR*, vol. abs/1904.10040, 2019. arXiv: 1904.10040. [Online]. Available: http://arxiv.org/abs/1904.10040.

[2] S. S. Villar, J. Bowden, and J. Wason, "Multi-armed Bandit Models for the Optimal Design of Clinical Trials: Benefits and Challenges," *Statistical Science*, vol. 30, no. 2, pp. 199–215, 2015. DOI: 10.1214/14-STS504. [Online]. Available: https://doi.org/10.1214/14-STS504.

[3] S. Bubeck and N. Cesa-Bianchi, "Regret analysis of stochastic and nonstochastic multi-armed bandit problems," *CoRR*, vol. abs/1204.5721, 2012. arXiv: 1204.5721. [Online]. Available: http://arxiv.org/abs/1204.5721.

[4] M. Puterman, *Stochastic models / edited by D.P. Heyman, M.J. Sobel.* (Handbooks in operations research and management science ; v. 2), eng. Amsterdam: North-Holland, 1990, ch. 8, pp. 331–434, ISBN: 0444874739.

[5] J. C. Gittins, "Bandit processes and dynamic allocation indices," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 41, no. 2, pp. 148–177, 1979, ISSN: 00359246. [Online]. Available: http://www.jstor.org/stable/2985029 (visited on 06/06/2023).

[6] P. Whittle, "Multi-armed bandits and the gittins index," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 143–149, 1980. DOI: https://doi.org/10.1111/j.2517-6161.1980.tb01111.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1980.tb01111.x. [Online]. Available: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1980.tb01111.x.

[7] J. Chakravorty and A. Mahajan, "Multi-armed bandits, gittins index, and its calculation," in *Methods and Applications of Statistics in Clinical Trials.* John Wiley and Sons, Ltd, 2014, ch. 24, pp. 416–435, ISBN: 9781118596333. DOI: https://doi.org/10.1002/9781118596333.ch24. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781118596333.ch24. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118596333.ch24.

[8] P. Varaiya, J. Walrand, and C. Buyukkoc, “Extensions of the multiarmed bandit problem: The discounted case,” *IEEE Transactions on Automatic Control*, vol. 30, no. 5, pp. 426–439, 1985. DOI: 10.1109/TAC.1985.1103989.

[9] D. Teneketzis and A. Mahajan, “Stochastic networked control systems: Stabilization and optimization under information constraints,” English (US), in *Systems and Control* (Systems and Control: Foundations and Applications 9781461470847), 1st ed., Systems and Control: Foundations and Applications 9781461470847. Switzerland: Birkhäuser, 2013, ch. 6, pp. 121–309.

[10] N.-O. Song and D. Teneketzis, “Discrete search with multiple sensors,” *Mathematical Methods of Operational Research*, vol. 60, Sep. 2004. DOI: 10.1007/s001860400360.

[11] D. G. Pandelis and D. Teneketzis, “On the optimality of the Gittins index rule for multi-armed bandits with multiple plays,” *Mathematical Methods of Operations Research*, vol. 50, no. 3, pp. 449–461, Dec. 1999. DOI: 10.1007/s001860050080. [Online]. Available: https://ideas.repec.org/a/spr/mathme/v50y1999i3p449-461.html.

[12] E. V. Denardo, H. Park, and U. G. Rothblum, “Risk-sensitive and risk-neutral multiarmed bandits,” *Mathematics of Operations Research*, vol. 32, no. 2, pp. 374–394, 2007, ISSN: 0364765X, 15265471. [Online]. Available: http://www.jstor.org/stable/25151795 (visited on 06/06/2023).

[13] M. Malekipirbazari and Ö. Çavu, “Risk-averse allocation indices for multiarmed bandit problem,” *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5522–5529, 2021. DOI: 10.1109/TAC.2021.3053539.

[14] A. Ruszczyski, “Risk-averse dynamic programming for markov decision processes,” English (US), *Mathematical Programming*, vol. 125, no. 2, pp. 235–261, Oct. 2010, ISSN: 0025-5610. DOI: 10.1007/s10107-010-0393-3.

[15] Ü. Sahin, V. Yücesoy, A. Koç, and C. Tekin, “Risk-averse ambulance redeployment via multi-armed bandits,” in *2018 26th Signal Processing and Communications Applications Conference (SIU)*, 2018, pp. 1–4. DOI: 10.1109/SIU.2018.8404439.