Universidade Federal do ABC

Universidade Federal do ABC
CECS - Centro de Engenharia, Modelagem e Ciências Sociais Aplicadas

Igor Bandim de Oliveira

Algoritmos de Multi-Armed Bandits aplicados a Testes AB em Marketing

Trabalho de Graduação

Volume I

Santo André
2019

Igor Bandim de Oliveira

Algoritmos de Multi-Armed Bandits aplicados a Testes AB em Marketing

Trabalho de graduação apresentado ao curso de Engenharia de Informação, como parte dos requisitos necessários à obtenção do título de bacharel em Engenharia de Informação.

Orientador: Ricardo Suyama

Volume I

Santo André
2019

Resumo

Neste trabalho é testada uma solução de otimização para Teste AB de *e-mail* utilizando uma modelagem através do problema de *Multi-Armed Bandits*. Partindo de uma abordagem estocástica para a solução deste problema, são adotados 4 algoritmos como base de comparação: *Greedy Algorithms, Softmax, UCB-1* e *Thompson Samping.* O desempenho desses algoritmos é avaliado através de simulações computacionais e o algoritmo de melhor desempenho é utilizado para um experimento de *Testes AB* em um ambiente controlado.

Palavras-chave: *Multi-Armed Bandit*, Teste AB, *Thompson Sampling.*

Abstract

This works presents a solution for AB Testing optimization , applying a model based on Multi-Armed Bandits problematic. Starting from a stochastic approach for this problem solution, 4 different algorithms are choosen for comparison basis: Greedy Algorithms, Softmax, UCB-1 and Thompson Sampling. The algorithms's performance is evaluated throughout computer simulations and the best performing algorithm is choosen for a AB Test in a controlled environment.

Key Words: Multi-Armed Bandits, AB Tests, Thompson Sampling.

Lista de ilustrações

Sumário

1 Introdução

A utilização de modelos de aprendizado de máquina baseados em estatística Bayesiana vem crescendo amplamente dentro do contexto de modelos de aprendizado de máquina aplicados à *marketing digital*. Esta abordagem se destaca em relação a modelos clássicos de predição ou regressão como as CART (Classification and Regression Trees) [1]. Esta vantagem se dá tanto pelo seu bom desempenho em problemas de separação não-linear , quanto pela sua ausência de pressupostos em relação ao conjunto de dados.

Diversas estratégias de testes em *marketing* voltados à experimentação podem ser abordadas a partir desta perspectiva. Um dos casos de aplicação que podem ser citados é o de Teste AB: testes em que se comparam dois materiais distintos através de um mesmo canal de comunicação. O objetivo destes testes é avaliar qual dos dois apresenta um melhor resultado sobre determinado indicador de desempenho [4].

Alguns exemplos que podem ser citados são a taxa de *cliques* em *e-mails*, tempo de navegação em páginas ou conversão sobre a oferta de determinado produto. Cada material testado, em geral, possui uma probabilidade de adesão ao público, desconhecida no momento inicial de execução dos testes e que é estimada ao final do processo com base nos resultados dos indicadores de desempenho.

De forma geral, Testes AB costumam ser implementados segmentando o público total que participará do experimento conforme o número de materiais testados em cada período de execução, chegando a uma estimativa de probabilidade uma vez que o número de testes é estatisticamente relevante. Este modelo se mostra eficiente quando não se há custos envolvidos com a alocação de tempo para testes ou número de execução dos mesmos, o que não costuma ocorrer na maior parte das aplicações em casos reais. A deficiência desta abordagem se dá pelo seu caráter puramente experimental – a informação obtida ao longo do tempo simplesmente não é utilizada para execução de etapas exploratórias.

Neste trabalho é testada uma solução para otimização de Testes AB utilizando uma modelagem através do problema dos *Multi-Armed Bandits.* Os algoritmos para solução desse problema, em geral, difenciam-se pela forma como as etapas exploratórias e experimentais são ponderadas. Alguns algoritmos definem períodos específicos para exploração e experimentação - os chamados E*pslon-greedy algorithms* [6]. Outros algoritmos, como o *UCB* [5] penalizam recursos que já foram utilizados com maior frequência. Além disso, existe um algoritmo especial , conhecimento como *Thompson Sampling* [7], que utiliza princípios da estatística Bayesiana para ponderar as escolhas dos recursos em cada etapa. Este último será um dos pontos centrais de estudo deste projeto.

1.1 Organização do Texto

No Capítulo 2 é apresentada uma breve descrição sobre testes AB, apresentando algumas justificativas para otimização dos mesmos.

O Capítulo 3 apresenta o problema dos Multi-Armed Bandits, discute alguns tópicos de aprendizado por reforço relacionados ao mesmo e enuncia alguns dos principais algoritmos voltados para a sua solução. É dada ênfase aos *greedy algorithms, Softmax e ao Upper Confidence Bound (UCB-1)*.

O Capítulo 4 traz uma introdução à estatística Bayesiana para apresentar o algoritmo do *Thompson Sampling*, algoritmo estruturado sobre as bases deste modelo estatístico.

No Capítulo 4 são exibidos os resultados de simulações entre ttes AB regulares e os 4 algoritmos de multi-armed bandits discutidos , inicialmentese sob um ambiente controlado. Posteriormente, o algoritmo de Thompson Sampling é utilizando uma base real de clientes como amostragem, realizando testes de disparos de diferentes e-mails e comparando os resultados obtidos com os disparos utilizando a abordagem clássica de testes AB.

No Capítulo 5 estrutura-se a conclusão sobre o desempenho de cada um dos algoritmos e são feitas algumas sugestões para melhorar o modelo de testes.

2 Introdução à metodologia de Testes AB

Os Testes AB surgem dentro do contexto de *marketing digital* como ferramentas práticas para crescimento de negócio baseados em testes de hipótese. Essa ferramenta pode ser aplicada em e-mails marketing, segmentação de experiência de navegação em sites, disponibilização de produtos em aplicativos de celular, dentre outras inúmeras aplicações digitais disponíveis no mercado.

O processo de como se dá a implementação de um Teste AB é extremamente simples, e é nessa simplicidade que reside a sua vantagem de utilização. Em geral, ele parte da identificação de um problema de negócio e das ações conjuntas entre áreas de produto e tecnologia das empresas para solução desse problema. Alguns exemplos práticos que podem ser citados são:

- Diminuição de vendas de algum produto em algum canal digital (e-mail, aplicativos para celular ou portal);
- Aumento no número de clientes que entram em contato com os canais de atendimento com reclamações;
- Diminuição do volume de acessos semanal em uma determinada página de serviço.

Uma vez identificado o problema, as áreas partem para um levantamento de hipóteses sobre qual poderiam ser as raízes desse problema e que medidas digitais poderiam ser usadas para solucioná-lo.

No caso da diminuição de vendas de algum produto em um canal digital, esta queda poderia estar relacionada ao carregamento do *e-mail*, do aplicativo e das páginas do site. O problema também poderia residir na forma como o produto está sendo divulgado. No caso do aumento do número de reclamações, isto poderia estar relacionado a disposição das páginas de dúvidas nos portais de atendimento, ou à falta de informações mais profundas que atendessem às necessidades dos clientes.

Após o levantamento de hipóteses, deseja-se testar possíveis soluções para as mesmas utilizando um número de clientes específicos, que comumente costuma ser definido como “base”. A proposta do teste AB é que cada solução apresentada seja testada em uma parcela dessa base e que esses testes se baseie em alguma métrica bem definida. Para página de produto, a métrica testada poderia ser o número de clientes que entram na página e concluem a compra. No caso de *e-mails*, uma alternativa é testar o número de clientes que clicam no chamado *Call-to-Action (CTA),* botão característico de *e-mails* que direciona para a página da empresa.

Normalmente esses testes são realizados dividindo a base de forma igual para cada versão testada ou enviando os mesmos de forma aleatória por um determinado período. O processo se dá até que se tenha um resultado estatisticamente relevante

entre número de envios e variação nas métricas entre as versões testadas. Apesar de ser uma solução prática, ela apresenta alguns problemas:

- Ao se utilizar segmentos de clientes como teste, corre-se um risco muito grande dos mesmos ficarem insatisfeitos com o produto/serviço ofertado e encerrarem o seu relacionamento com a empresa. O número de clientes utilizados para teste é um recurso limitado;
- Em geral, quando se fala em testes para personalização (variação de escala de cores, disposição de elementos, etc), diferentes clientes podem aderir de formas distintas a mesma versão de teste;
- Quando se executa um número elevado de testes simultaneamente, nem sempre é factível realizar um acompanhamento constante dos mesmos;
- Cada vez que um teste é realizado e um resultado é colhido, se obtém uma determinada informação de como é a adesão da base a cada alternativa de solução. Ao se utilizar uma segmentação aleatória, ou ainda uma uniformemente distribuída, ignora-se essa informação por completo.

A cada dia as empresas se deparam com um número maior de competidores nos segmentos digitais, com uma gama maior de produtos, serviços e experiências mais fluídas. Em geral, deseja-se que a identificação, levantamento de hipóteses e testes sejam realizados de modo mais otimizado possível, utilizando o mínimo possível de clientes dentro das bases.

Para este fim, é proposto neste trabalho a aplicação de um modelo estatístico, modelando os testes AB através do problema de *Multi-Armed Bandits.*

3 Introdução aos algoritmos de *Multi-Armed Bandits*

Nesta capítulo será apresentada uma introdução formal do problema de *Multi-Armed Bandits*, exibindo algumas métricas para comparar o desempenho dos mesmos dentro de um mesmo escopo. Além disso, será discutido o problema de exploração/experimentação característico de modelos de aprendizado por reforço e alguns algoritmos que podem ser aplicados como soluções para o problema de *Multi-Armed Bandits*, mostrando como os mesmos ponderam etapas de exploração/experimentação dentro da sua implementação.

Em geral, alguns algoritmos apresentados nesta seção podem ser encontrados em outras aplicações com objetivos variados e não correlacionados com a solução do problema de Multi-Armed Bandits. Apenas para fins de simplificação, estes algoritmos serão definidos neste trabalho como Algoritmos de *Multi-Armed Bandits (MAB).*

3.1 Definição do problema dos Multi-Armed Bandits

O cerne da discussão sobre *Multi-Armed Bandits (MAB)* pode ser associado à tomada de decisões de forma sequencial , estudada por Hebert Robbins em seu trabalho sobre *design* sequencial de Experimentos [8]. Robbins cita a problemática de se amostrar populações distintas no caso em que ambas se comportam como variáveis aleatórias com diferentes distribuições de probabilidade. Quando o número de amostras que podem ser obtidos de ambas populações é limitado, a escolha de quantas amostras devem ser obtidas de cada população para obter seus parâmetros de média e variância de forma otimizada é restringida.

O termo *Multi-Armed Bandit* [1] deriva das chamadas Bandits - máquinas de azar equipadas com alavancas, populares em cassinos norte-americanos. Na formulação clássica do problema, um indivíduo desloca-se a um cassino com uma quantidade limitada de recursos para apostar em n máquinas*,* devendo alocar estes recursos sequencialmente de modo a maximar a recompensa final obtida. A recompensa é entregue ao jogador logo após a escolha de uma das máquinas, mas o mesmo não possui nenhum conhecimento prévio sobre as máquinas ou a ordem em que as mesmas devem ser escolhidas.

Ao longo deste trabalho iremos nos referir constantemente a essas máquinas utilizando a terminologia *arm,* entrando em conformidade com os termos utilizados nas principais referências da bibliografia sobre o tema.

Esta modelagem é extremamente útil no contexto de Testes AB, uma vez que ela contempla aspectos importantes do método. Em geral, ao se iniciar algum Teste AB no escopo de Marketing há um limite em número de clientes que pode ser utilizado para testes, em orçamento alocado, em tempo de entrega e definição de resultado. Não obstante, é praticamente impossível inferir de forma assertiva qual será à adesão

do público a cada versão testada antes de o experimento ocorrer. Os resultados de adesão são obtidos conforme os clientes interagem com os materiais testados e os indicadores escolhidos como métricas de desempenho aumentam ou diminuem.

A abordagem do problema dos *Multi-Armed Bandits* varia dependendo de como o processo associado à entrega da recompensa é definido [2]: o mesmo costuma ser classificado como estocástico, markoviano ou adversarial. No caso estocástico, cada *arm* $i \in [n]$ se associa uma distribuição de probabilidade específica p_i em $[0, 1]$ e as recompensas para cada *arm* são extraídas dessas distribuições de forma independente e identicamente distribuída (iid). Em recompensas modeladas em processos Markovianos [10], para cada *arm* é possível associar um processo markoviano subjacente a um espaço de estado distinto, o que torna as ferramentas de análise mais similares as ferramentas utilizadas em aprendizado por reforço, modelo utilizado em problemas de aprendizado de máquina. Já no caso adversarial, não existe nenhum pressuposto probabilístico sobre as recompensas para cada *arm*. Nesse caso, as recompensas são puramente determinísticas.

A escolha da modelagem estatística do problema também define a escolha de algoritmos a serem estudados. O enfoque deste projeto será em algoritmos voltados a processo de recompensa estocástico, assim como será discutido posteriormente na seção de resultados.

Inicialmente, para comparar o desempenho de algoritmos de MAB com recompensa estocástica utiliza-se o conceito de regret (arrependimento em português). O regret é definido a partir da comparação do algoritmo com a estratégia mais otimizada possível, na qual o jogador escolhe o *arm* com a maior probabilidade de retorno ao longo de n rodadas. Nesse caso, analisa-se o quanto se perde por não ter escolhido o *arm* mais otimizado de modo constante. Dados $K \geq 2$ *arms*, e sequências $X_{i,1}, X_{i,2}, ...$de recompensas não conhecidas associadas com cada *arm* $i = 1, 2, ..K$, em cada janela de tempo $t = 1, 2, ...$ o algoritmo seleciona um *arm* I_t e recebe a sua recompensa associada $X_{I_t,t}$.

Desta forma, a métrica de regret (2.1) é definida matematicamente como:

$$R_n = \max_{i=1,..,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t} \tag{3.1}$$

Em geral, tanto a recompensa $X_{i,t}$ quanto a escolha de arm I_t podem ser estocásticos. Isto nos leva a duas novas definições: o regret médio(2.2) e o pseudo-regret(2.3):

$$\mathbb{E}[R_n] = \mathbb{E}\left[\max_{i=1,..,K} \sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t} \right] \tag{3.2}$$

$$\overline{R_n} = \max_{i=1,..,K} \mathbb{E}\left[\sum_{t=1}^{n} X_{i,t} - \sum_{t=1}^{n} X_{I_t,t} \right] \tag{3.3}$$

Pode-se observar que o pseudo-regret fornece uma noção mais fraca de regret, uma vez que ela considera apenas a ação que máxima o valor médio da recompensa, e não a recompensa propriamente dita. O valor médio do regret, por outro lado, considera o valor médio das ações que maximizam a recompensa. Em termos práticos, é possível afirmar que $\mathbb{E}[R_n] \geq \overline{R_n}$.

Para cada *arm* $i = 1, 2, ..., K$ é possível associar uma distribuição de probabilidade v_i no intervalo $[0, 1]$. As recompensas $X_{i,t}$ podem ser extraídas das distribuições v_i de forma independente. Tomando μ_i como o valor médio da recompensa do *arm* i, nos definimos:

$$\mu_i^* = \max_{i=1,..,K} \mu_i \text{ e } i^* \in \max_{i=1,..,K} \mu_i$$

Desta forma, podemos extrair uma nova métrica de pseudo regret (2.4):

$$\overline{R_n} = n\mu^* - \mathbb{E} \sum_{t=1}^{n} \mu_{I_t} \tag{3.4}$$

O pseudo regret geralmente é adotado uma vez que ele tentar escolher a ação que irá maximar ao valor esperado, ao invés de selecionar a ação mais otimizada em uma sequência de ações realizadas. Ele está naturalmente mais alinhado com a noção estocástica de entrega da recompensa definida inicialmente.

3.2 Aprendizado por Reforço e o *Trade-Off* entre exploração e experimentação.

Partindo do pressuposto de uma recompensa estocástica, a solução do problema de *Multi-Armed Bandits* se torna estimar a probabilidade v_i associada a cada *arm* $i = 1, 2, ..., K$ ao longo de cada rodada até que as n rodadas sejam concluídas. Suponha que se delegue a um agente a escolha dos *arms* ao longo de cada rodada, até o final da última rodada. Conforme ele recebe as recompensas, ele pode inferir

sobre as distribuições de probabilidade dos *arms* e escolher o que maximize o seu retorno.

Esta formulação se aproxima muito do objeto de estudo de aprendizado por reforço, um dos diversos campos de estudo do aprendizado de máquina. O aprendizado por reforço busca programar agentes para executar tarefas específicas criando modelos de aprendizado com recompensas e punições, mas sem especificar como a tarefa deve ser executada, utilizando modelos estatísticos e programação dinâmica.

O modelo clássico de aprendizado por reforço inclui 2 objetos principais: o agente e o ambiente. O agente é responsável pela escolha das ações a serem tomadas. Conforme as ações são tomadas, o ambiente varia o seu estado, e esta mudança é comunicada ao agente através de um estímulo, sendo o mesmo uma recompensa ou uma punição . A entrega desses estímulos pode ser O*n-Demand* - logo após a ação - ou pode ser realizada ao final de um conjunto de n ações. No caso da formulação inicial dos MAB, a entrega da recompensa segue o primeiro caso. Além disso, assume-se que as recompensas sejam estacionárias, isto é, que as distribuições de probabilidade de cada uma dos *arms* não são alteradas ao longo dos estados.

A definição dos algoritmos inicia-se na definição de um modelo ótimo [11]. O modelo ótimo irá definir a perspectiva de futuro em que o agente irá se basear para escolher as suas ações atuais. Existem três modelos que geralmente são adotados:

a) Modelo de Horizonte Finito: Dado um determinado instante de tempo, o modelo deve otimizar a recompensa esperada pelos próximos passos h, sem considerar aqules realizados no passado. Em geral, o modelo pode trabalhar de uma maneira não estacionária, alterando o seu comportamento ao longo de cada um dos seus k passos, com $k \in [0, h]$, otimizando a estratégia para um horizonte de $h - k$ movimentos, ou adotando a mesma para todos os passos subsequentes, baseado na definição inicial para um horizonte de h passos. O valor da recompensa otimizado, nesse caso é:

$$\mathbb{E} \sum_{t=0}^{h} r_t \tag{3.5}$$

b) Modelo de Horizonte Infinito: Neste modelo, considera-se a recompensa de longo prazo, penalizando-se as recompensas futuras por um fator γ, com $\gamma \in [0, 1]$. Espera-se que o modelo convirja para um valor de recompensa, logo o valor gama indica a velocidade com que as recompensas futuras são penalizadas. Neste caso, o valor da recompensa otimizada é:

$$\mathbb{E} \sum_{t=0}^{\infty} \gamma^t r_t \tag{3.6}$$

c) Modelo de Recompensa Média: A recompensa otimizada é o valor médio da recompensa a longo prazo. Neste caso as recompensas iniciais são desvalorizadas em relação à recompensa de longo prazo. O valor otimizado nesse caso é:

$$\mathbb{E}(\frac{1}{h}\sum_{t=0}^{h} r_t) \tag{3.7}$$

Pode-se observar que o modelo definido de Regret definido utiliza um padrão de recompensa média.

A principal dificuldade de se trabalhar com o modelo estocástico é a determinação das escolhas dos *arms* ao longo do tempo. Para que o agente adote uma estratégia sólida de escolha dos *arms,* ele deve ter conhecimento suficiente de cada um deles para se traçar um perfil da distribuição estatística de suas recompensas. Em aplicações reais esta determinação é puramente experimental e deve ser determinada aproveitando-se de parte das ações disponíveis do agente.

Considere um agente escolhendo entre um conjunto de k arms, com o objetivo de maximar a sua recompensa ao longo de rodadas. Suponha também que no início das m primeiras rodadas, o agente vem escolhido em proporções maiores o *arm* i , percebendo que o mesmo tem performado melhor que os outros. O agente pode utilizar essa informação e continuar explorando o mesmo *arm* continuamente, ou pode utilizar o restante de suas rodadas para explorar outros *arms* que não vem sendo utilizados para se obter conhecimento sobre os mesmos.

O grande problema no caso do aprendizado por reforço é que o agente ao escolher uma ação simplesmente recebe uma recompensa, e não uma diretriz de qual teria sido ação dado um determinado estado. Desta forma, a única maneira que o agente tem de encontrar a melhor política de escolha do *arm* para determinado estado é realizar teste de forma contínua, até que o número de execuções forneça alguma validade estatística.

Esta avaliação de troca entre experimentar soluções distintas que ainda não foram testadas de forma satisfatória e explorar soluções que apresentam um bom desempenho é clássica nos problemas de aprendizado por reforço. Diversos modelos tentam ponderá-la, de modo a obter uma política otimizada para determinado processo. Uma maneira de entender como as etapas entre experimentação/exploração são otimizadas no aprendizado por reforço é através da abordagem markoviana de distribuição das recompensas, utilizando a noção de utilidade de um estado.

Thrun [12] define utilidade a partir da ótica da programação dinâmica, criando uma política de escolha ótima mapeando os estados em ações de um agente, assumindo um processo estático no tempo.

A utilidade $V^{\pi}(s)$, nesse caso, mede o valor esperado da recompensa futura acumulada, assumindo que a mesma é ponderada por um fator de desconto $\gamma \in (0, 1)$,

sendo s o estado atual em que o sistema se encontra , π a política atual utilizada e r_t a recompensa esperada no instante de tempo t, trabalhando num horizonte de tempo infinito:

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{t=t_0}^{\infty} \gamma^t r_t | s\Big] \tag{3.8}$$

Dado que o processo é definido como estático, podemos re-escrever a equação como:

$$V^{\pi}(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s\Big] \tag{3.9}$$

Quanto maior a utilidade, maior o valor esperado da recompensa obtida. Quando o objetivo do agente é maximar a recompensa, ele seleciona a uma ação a que maximiza a função:

$$f^*(a) = \sum P_{ss'}(a).V^{\pi^*}(s') \tag{3.10}$$

Sendo s' o estado subsequente ao estado s e π^* a política de escolha otimizada, encontrada através da identificação da função $V^{\pi^*}(s')$. Esta função, em geral, é encontrada de maneira iterativa e assíncrona. Seja $\hat{V}^t$ a estimativa da política no instante de tempo t e $\hat{\pi}^t$ a política obtida neste mesmo instante de tempo, maximizando a medida de exploração na seleção de uma ação. Desta forma temos:

$$f(a) = \sum P_{ss'}(a).\hat{V}^t(s') \tag{3.11}$$

Quando a ação a é performada, a estimativa de recompensa $\hat{V}^t$ no estado atual s é atualizada com o recebimento da recompensa imediata r_t. A estimativa $\hat{V}^{t+1}$ para o estado subsequente, é dada pela equação de comparação de reforço:

$$\hat{V}^{t+1}(s) = \begin{cases} (1-\eta).\hat{V}^t(s) + \eta.(r_t + \gamma.\hat{V}^t(s')) & \text{se s em t e s' em t+1} \\ \hat{V}^t(s) & \text{caso contrário} \end{cases} \tag{3.12}$$

Sendo η a taxa de aprendizado, geralmente adotada como um valor baixo. Em geral, a atualização dos estados na regra (2.12), não busca apenas maximar o valor de $f(.)$ para a seleção da ação. A seleção de ação é misturada com um termo de experimentação, que garante que a função não fique presa em um mínimo local. O termo de correção $r_t + \gamma.\hat{V}^t(s')$ é conhecido como erro de diferença temporal .

As próximas sessões discutem de maneira prática as políticas de controle de experimentação/exploração implementadas por alguns algoritmos específicos. Estes costumam ser implementados tanto em aplicações com perfil de distribuição estatístico estocásticos quanto em aplicações com perfil estatístico markoviano.

3.3 Algoritmos de política *Greedy*

Como visto nos capítulos anteriores, o ponto central de implementação de algoritmos com soluções de aprendizado por reforço é a escolha da política de controle exploratória. Os algoritmos que discutiremos nessa sessão são os algoritmos *greedy* ("gananciosos", traduzindo do inglês"), citados por Kuleshov [13] em seu estudo sobre os *Muli-Armed Bandits*.

Estes algoritmos, apesar de simples, já apresentam ferramentas mais sofisticadas e que saem do contexto puramente experimental e aleatório, introduzindo uma política de exploração bem definida.

3.3.1 Algoritmo Greedy

O primeiro algoritmo mostrado é o *greedy*. Neste caso a política de escolha dos *arms* é puramente exploratória e busca otimizar a recompensa final escolhendo de forma contínua o *arm* que possui a maior probabilidade de retorna da recompensa. Esta escolha basea-se em uma etapa inicial experimental, dedicada para definição das probabilidades de retorno dos mesmos. Neste caso, a política de escolha das ações é dada por:

$$a = argmax_{a\ inA}(\mathbb{E}(R_t(a)) \tag{3.13}$$

Sendo $\mathbb{E}(R_t(a)$ o valor esperado da ação a no instante de tempo t .

Considere um agente escolhendo entre os *arms*, com acesso à n rodadas de escolha. Sejam as distribuições de probabilidade de retorna das recompensas em cada *arm* estocásticas, $R_t(i) \in [0, 1]$ a recompensa instantânea da seleção do *arm* i no instante t binária e n_0 o número de rodadas inciais definidas como período experimental. O algoritmo *greedy* pode ser definido da seguinte forma:

1. Nas n_0 primeiras rodadas, selecione um arm i de forma aleatoria;

2. Ao final das n_0 rodadas, compute a probabilidade de retorno da recompensa θ_i do arm i através da expressão:

$$\theta_i(n_0 + 1) = \frac{\sum_{n=0}^{n_0} R_i(n)}{\sum_{n=0}^{n_0} a(n) = i} \tag{3.14}$$

Sendo $a(t)$ 1 quando o arm i foi selecionado no intante n e 0, caso contrário.

3. Nas n rodadas seguintes, o arm i é selecionado no instante t através da expressão:

$$a(t) = argmax_{i \in k} \theta_i(t) \tag{3.15}$$

4. Ao longo de cada rodada, após a definida da ação e o recebimento da recompensa, atualize as probabilidade utilizando a a expressão:

$$\theta_i(t) = \frac{\sum_{n=0}^{t} R_i(n)}{\sum_{n=0}^{t} a(n) = i} \tag{3.16}$$

3.3.2 Algoritmo ϵ-Greedy

Este algoritmo melhora a performance do algoritmo greedy comum, introduzindo o parâmetro $\epsilon \in [0, 1]$ para a etapa pós experimental inicial. Este parâmetro indica a porcentagem do tempo que será dedicada para etapas experimentais futuras.

Em geral adota-se valores pequenos para o parâmetro , de modo a garantir que as etapas exploratórias sejam predominantemente dominantes em frequência. A vantagem em relação ao algoritmo *greedy* inicial é que nem sempre o número de ações alocadas para a etapa experimental inicial é suficientemente grande.

A definição do algoritmo segue o mesmo padrão do algoritmo *greedy* anterior, introduzindo uma etapa intermediária entre os passos 3 e 4, e alterando o passo quatro de acordo com a saída do passo 3:

1. Nas n_0 primeiras rodadas, selecione um *arm* i de forma aleatoria;

2. Ao final das n_0 rodadas, compute a estimativa de probabilidade de retorno da recompensa θ_i do *arm* i através da expressão:

$$\theta_i(n_0 + 1) = \frac{\sum_{n=0}^{n_0} R_i(n)}{\sum_{n=0}^{n_0} a(n) = i} \tag{3.17}$$

Sendo $a(t)$ 1 quando o *arm* i foi selecionado no instante t e 0, caso contrário.

3. A cada instante de tempo , selecionar um valor α aleatoriamente, considerando uma distribuição estatística uniforme no intervalo entre $[0, 1]$.

4. Caso $\alpha > \epsilon$, selecionar o *arm* i através da expressão (2.18). Caso contrário, selecionar o *arm* i de forma aleatória.

$$a(t) = argmax_{i \in k} \theta_i(t) \tag{3.18}$$

5. Ao longo de cada rodada, após a definida da ação e o recebimento da recompensa, atualize as estimativas de probabilidade utilizando a expressão:

$$\theta_i(t) = \frac{\sum_{n=0}^{t} R_i(n)}{\sum_{n=0}^{t} a(n) = i} \tag{3.19}$$

3.3.3 Algoritmo com ϵ-Greedy com decaimento

Esta segunda versão do algoritmo ϵ-Greedy busca otimizar o processo de alocação de ações para etapas experimentais diminuindo o fato ϵ ao longo do tempo. A premissa é de que, conforme o número de ações aumenta e os *arms* vão sendo selecionados, a variância da estimativa de probabilidade de retorno da recompensa associada a cada *arm* diminui, uma vez que se utilizam mais dados para compor a estimativa.

O fator ϵ é diminuído através de uma função $\gamma(t)$ após a execução da etapa puramente experimental no momento inicial. Nesse caso o algoritmo pode ser definido como:

1. Nas n_0 primeiras rodadas, selecione um arm i de forma aleatoria;

2. Ao final das n_0 rodadas, compute a estimativa de probabilidade de retorno da recompensa θ_i do *arm* i através da expressão:

$$\theta_i(n_0 + 1) = \frac{\sum_{n=0}^{n_0} R_i(n)}{\sum_{n=0}^{n_0} a(n) = i} \tag{3.20}$$

Sendo $a(t)$ 1 quando o *arm* i foi selecionado no instante t e 0, caso contrário.

3. Atualize o parâmetro, sendo ϵ_0 o valor inicial de ϵ em $t = n_0$

$$\epsilon(t) = \epsilon_0 . \gamma(t) \tag{3.21}$$

4. A cada instante de tempo , selecionar um valor α aleatoriamente, considerando uma distribuição estatística uniforme no intervalo entre $[0, 1]$.

5. Caso $\alpha > \epsilon(t)$, selecionar o *arm* i através da expressão (2.18). Caso contrário, selecionar o *arm* i de forma aleatória.

$$a(t) = argmax_{i \in k} \theta_i(t) \tag{3.22}$$

6. Ao longo de cada rodada, após a definida da ação e o recebimento da recompensa, atualize as estimativas de probabilidade utilizando a expressão:

$$\theta_i(t) = \frac{\sum_{n=0}^{t} R_i(n)}{\sum_{n=0}^{t} a(n) = i} \tag{3.23}$$

3.4 O algoritmo de Exploração de Boltzman (*Softmax*)

Os algoritmos *greedy*, apesar de apresentarem uma alternativa mais otimizada para testes AB, ainda selecionam os *arms* de forma completamente aleatória na rodada de experimentação. Esta solução não olha de forma direta as estimativas de probabilidade de retorno da recompensa ao longo do tempo e podem escolher o *arm* com a menor probabilidade de retorno durante a etapa exploratória.

O algoritmo apresentado a seguir determina a probabilidade de retorno do *arm* com base nas suas estimativas de recompensa média ao longo do tempo, utilizando uma distribuição de Boltzman. Neste caso não existem etapas puramente experimentais, adotando-se sempre o *arm* com a maior estimativa de probabilidade de retorno.

Kuleshov [6] define o algoritmo de estimativa de probabilidade de retorno de um *arm* i como sendo:

$$\theta_i(t+1) = \frac{e^{\hat{\mu}_i(t)/\tau}}{\sum_{j=1}^{k} e^{\hat{\mu}_i(t)/\tau}} \tag{3.24}$$

Sendo $\hat{\mu}_i(t)$ a estimativa de recompensa média do *arm* i no instante de tempo t. O parâmetro τ é conhecido como parâmetro de temperatura e controla o grau de aleatoriedade do experimento. Quando este parâmetro é nulo, a escolha dos *arms* é puramente exploratória, escolhendo sempre o *arm* com maior probabilidade de retorno com base na estimativa de valor médio. Quando este parâmetro tende ao infinito, a escolha dos *arms* segue uma distribuição de probabilidade 100% uniforme.

Uma vez definidas as estimativas de probabilidade a escolha do *arm* i no instante de tempo t segue como:

$$a(t) = argmax_{i \in k} \theta_i(t) \tag{3.25}$$

3.5 Algoritmos UCB

Uma das deficiências dos algoritmos vistos anteriormente é que os mesmos não observam as incertezas associadas à estimativa de recompensa de cada um dos *arms*

ao longo do tempo. Em geral, conforme se dá o processo de escolha dos *arms* ao longo do tempo, *arms* que tiveram um número menor de escolhas ao longo de cada iteração possuem um valor maior de variância atrelado aos mesmos. Essa desconsideração do fator de variância pode diminuir a estimativa ótima de retorno de um *arm* específico ao longo do tempo, tornando menor a recompensa final menor ao longo do processo.

Esta família de algor timos é conhecida como UCB (Upper Confidence Bound) e são baseados nos estudos de Lais e Robins [14] sobre regras de alocação adaptativas assintoticamente eficientes. Os algoritmos ponderam as probabilidades de retorno dos *arms* penalizando *arms* que tenham sido escolhidos com um número menor de ocorrências ao longo do experimento.

Neste estudo, focaremos na versão mais básica do algoritmo , o UCB-1. Este algoritmo, baseia-se no limite superior da recompensa. Neste caso, de forma análoga aos algoritmos ϵ-greedy, temos uma etapa inicial para estimativa das probabilidades de retorno das recompensas, seguida de uma etapa posterior, em que consirera-se o número de amostras utilizadas para estimar a recompensa média de cada *arm*.

Neste caso, o algoritmo pode ser definido da seguinte forma:

1. Nas n_0 primeiras rodadas, selecione um *arm* i de forma aleatoria;

2. Ao final das n_0 rodadas, compute a estimativa de probabilidade de retorno da recompensa θ_i do *arm* i através da expressão:

$$\theta_i(t) = \hat{X}_i(t) + c_i(t) \tag{3.26}$$

sendo:

$$\hat{X}_i(t) = \frac{\sum_{n=0}^{t} R_i(n)}{\sum_{n=0}^{t} a(n) = i} \tag{3.27}$$

e

$$c_i(t) = \sqrt{\frac{\xi.B.log(t)}{\sum_{n=0}^{t} a(t) = i}} \tag{3.28}$$

com $a(n)$ igual à 1 quando o *arm* i tiver sido selecionado no instante n e 0, caso contrário.

3. Nas n rodadas seguintes, o *arm* i é selecionado no instante t através da expressão:

$$a(t) = argmax_{i \in k} \theta_i(t) \tag{3.29}$$

O termo B é definido como o limite superior para a recompensa, e o termo ξ é um valor constante, adaptado para o problema. Pode-se observar que as expressões da estimativa de probabilidade utiliza-se do valor médio da recompensa $\hat{X}_i(t)$ e da função $c_i(t)$, conhecida como função de preenchimento. Esta função diminui conforme o *arm* i é selecionado múltiplas vezes ao longo do tempo. Desta forma, *arms* selecionados um número menor de vezes passam a ser escolhidos com uma maior frequência, até que se tenha outras amostras de retorno de recompensa dos mesmos.

O algoritmo UCB possui uma série de variações discutidas por diversos autores como Garivier e Moulines [15] em seu estudo de políticas UCB para aplicações de MAB não estacionárias, no qual são citadas duas variações do UCB: O UCB descontado, que adiciona um parâmetro empírico $\gamma \in (0, 1)$ para penalizar estimativas de recompensa mais antigas, e o UCB de janela móvel, que considera a recompensa de cada arm tomando uma janela de τ rodadas.

Não obstante, uma terceira versão do algoritmo é discutida por Aldibert e Munos [16] no seu estudo sobre otimização de algoritmos de MAB para ambientes estocásticos. Esta versão citada como V-UCB , re-estrutura a função de preenchimento regular do UCB-1, considerando o próprio valor da variância associada à estimativa de recompensa de cada *arm* de acordo com o número s de amostras tomadas.

4 Introdução ao *Thompson Sampling*

4.1 Modelos Bayesianos

Os algoritmos mais complexos vistos até o momento necessitam da informação da recompensa para a escolha do *arm* k no instante de tempo t para que estes apresentam um desempenho minimamente satisfatória. Em aplicações reais, por outro lado, esta informação nem sempre é entregue de forma instantânea. Em testes AB no contexto de *e-mail*, por exemplo, é necessário observar os envios de *e-mail* em uma determinada janela de tempo para avaliar os resultados do recebimento da recompensa.

O algoritmo visto na próxima sessão oferece uma solução funcional para este empecilho prático a partir de um modelo baseado na lógica Bayesiana. Estes modelos derivam dos estudos de Thomas Bayes [17], responsável por um dos mais importantes teoremas na teoria da probabilidade, o Teorema de Bayes.

O Teorema de Bayes, segundo Carlin e Louis [18], se inicia a partir de um modelo de amostragem para um conjunto de dados observados, $y = (y_1, y_2, ..., y_n)$ dado um vetor de parâmetros θ , dos quais não se tem conhecimento. O modelo de amostragem pode ser visto tanto como uma distribuição de probabilidade $f(y|\theta)$ ou simplesmente como uma probabilidade condicional $L(\theta; y)$, quando visto como uma função de θ ao invés de y .

O modelo Bayesiano parte do pressuposto que o parâmetro θ é aleatório e com uma distribuição de probabilidade que resume toda a informação que nos temos sobre esta variável e que não está associada ao conjunto de dados y. Não obstante, nos supomos que existem parâmetros η já conhecidos, os quais chamaremos de hiper-parâmetros, de modo que os conjunto de dados não oferece nenhuma informação adicionar sobre os mesmos. Logo $\pi(\theta) \equiv \pi(\theta|\eta)$.

A fórmula conhecida como teorema de Bayes, no que lhe concerne, permite inferir sobre o parâmetro θ a partir da sua distribuição à posteriori, dada por:

$$p(\theta|y) = \frac{p(y,\theta)}{p(y)} = \frac{p(y,\theta)}{\int p(y,\theta)d\theta} = \frac{f(y|\theta)\pi(\theta)}{\int f(y|\theta)\pi(\theta)d\theta} \tag{4.1}$$

Em geral, a versão do teorema de Bayes que observa-se na literaruta é a versão discreta. Partindo de um evento de interesse A e um conjunto de eventos B_j , $j = 1, 2, .., K$, mutualmente excludentes entre si e exaustivos, dadas as probabilidades $P(B_j)$ dos eventos B_j e a probabilidade condicional $P(A|B_j)$, podemos calcular a probabilidade a partir das regras condicionais da probabilidade:

$$P(B_j|A) = \frac{P(A \cap B_j)}{P(A)} = \frac{P(A \cap B_j)}{\sum_{j=1}^{K} P(A \cap B_j)} = \frac{P(A|B_j)P(B_j)}{\sum_{j=1}^{K} P(A|B_j)P(B_j)} \tag{4.2}$$

Pode-se observar que, em ambos em casos, a estimativa a posteriori parte da estimativa a priori e de um evento do qual se conhece o resultado. Esta abordagem torna-se útil dentro do contexto de MAB uma vez que o processo iterativo de seleção dos *arms* e recebimento da recompensa já disponibiliza parte da informação necessária para se trabalhar sob uma ótica Bayeasiana.

4.2 O algoritmo Thompson Sampling

O algoritmo apresentado a seguir utilizado uma abordagem pseudo Bayesiana para apresentar uma solução para o problema do MAB. Desenvolvido com base nos estudos probabilísticos de William R Thompson [19], este modelo trabalha com uma distribuição à priori fictícia para as probabilidades de retorno da recompensa em cada *arm* k, escolhendo o *arm* através de uma amostragem a partir desta distribuição e atualizando a sua distribuição à posteriori a partir da informação da recompensa recebida.

A vantagem de se trabalhar com esta abordagem é que ao se adotar uma distribuição probabilística no lugar de uma estimativa direta tem-se um controle das etapas experimentação no processo a partir da própria variância da distribuição. Quando adotado um modelo distribuição probabilística adequada, conforme a seleção de um mesmo arm k aumenta, o valor amostrado aleatoriamente irá tender para o valor médio da distribuição, que oferece uma estimativa sólida para a probabilidade de retorrno do mesmo arm.

Segundo Gopalan e Manner [20], há outra vantagem de se utilizar este modelo no caso de modelos *Bandits* em que as distribuições de retorno da recompensa não são necessariamente independentes entre si . A utilização de uma atualização de distribuição à posteriori permite capturar correlações complexas entre ações de cada *arm* de forma implícita. Não obstante, este modelo de atualização também se mostra robusto para o caso de MAB com distribuições de recompensas Markovianas, trabalhando de forma eficiente a transição de estados.

A escolha da distribuição probabilística para os arms varia conforme a aplicação. Uma das distribuições mais utilizadas é a distribuição Beta. Gupta e Nadarajah [21] mostram que esta distribuição é útil para variáveis aleatórias no intervalo $[0, 1]$, como uma distribuição à priori para probabilidade de ocorrência de um evento, no contexto da estatística bayeasiana.

A função densidade de probabilidade uma variável aleatória $X \sim beta(\alpha, \beta)$ é dada por:

$$f(x, \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)} \tag{4.3}$$

sendo:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{4.4}$$

e

$$\Gamma(n) = (n-1)! \tag{4.5}$$

Em termos das médias e variância, a função beta apresenta os seguintes valores:

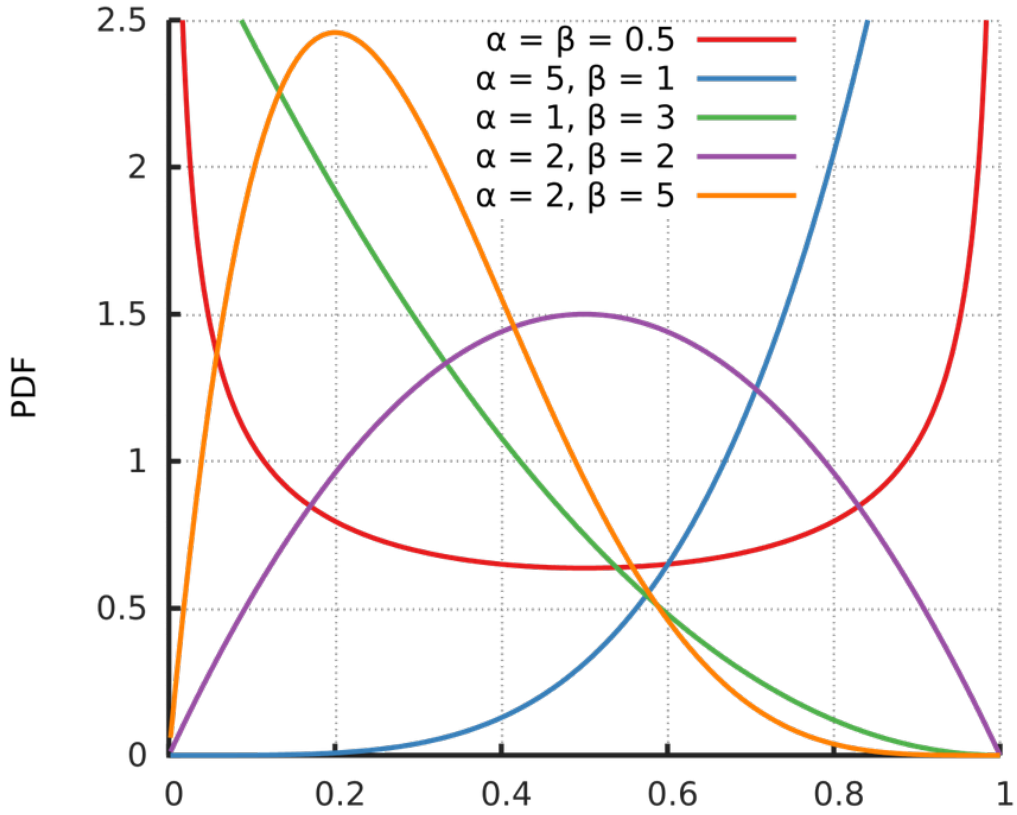
$$\mu = \frac{\alpha}{\alpha+\beta} \tag{4.6}$$

e

$$\sigma^2 = \frac{\alpha.\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} \tag{4.7}$$

Os parâmetros alfa e beta, como pode ser observado, definem a forma da distribuição. A figura abaixo mostra a função distribuição de probabilidade da função beta para alguns parâmetros alfa e beta distintos entre si.

Figura 1 – Distribuição Beta para diversos parâmetros α e β



Como pode-se observar, quanto maior o valor do parâmetro alfa em relação ao parâmetro beta, mais próximo de 1 a média da distribuição de probabilidade se aproxima. Esta relação é extremamente útil para se modelar a atualização da probabilidade à posteriori no Thomson Sampling.

Considere novamente o problema de um agente encarregado de escolher entre k *arms,* com acesso à n rodadas de escolha. Seja o valor de probabilidade de retorna das recompensas estocásticas, $R_t(i) \in [0, 1]$ e a recompensa instantânea da seleção do *arm* i no instante t binária. O algoritmo Thompson Sampling pode ser definido da seguinte forma:

1. Inicie a distribuição à priori de cada *arm* i e a fdp de cada *arm* com uma distrição beta com parâmetros α e β igual a 1;

2. Em cada instante de tempo t amostre o valor da distribuição de cada *arm* i a partir de sua distribuição beta.

3. Selecione o *arm* i que a partir da expressão:

$$a(t) = argmax_{i \in k} \theta_i(t) \tag{4.8}$$

4. Após o recebimento da recompensa, atualize os parâmetros da distribuição a partir da recompensa recebida, obtendo a sua distribuição a posteriori:

$$(\alpha_{a(t)}, \beta_{a(t)}) \rightarrow (\alpha_{a(t)} + r(t), \beta_{a(t)} + 1 - r(t)) \tag{4.9}$$

5 Simulações dos Algoritmos de *Multi-Armed Bandits* e Aplicação em Teste AB

Nesta etapa implementaram-se simulações com os algoritmos de teste AB discutidos na etapa anterior, tomando as mesmas condições ao longo de todas as simulações. Além disso, foi implementado um teste AB para *e-mail marketing em* um ambiente controlado, em parceria com uma corretora de investimentos para testar duas recomendações de produto em um e-mail disparado para clientes que possuíam vencimentos referentes à ativos de renda fixa. Nesta classe de investimentos o retorno é conhecido no momento da aplicação no produto.

As simulações e a aplicação para vencimentos foram implementadas em *Python*, segmentando cada estratégia de *Multi-Armed Bandi*t em uma classe de programação distinta. Para a aplicação em *e-mail* optou-se a implementação do *Thompson Sampling*, em razão do mesmo ter uma estabilidade melhor em regret e recompensa final, conforme será verificado na etapa de simulação.

Todas as simulações foram implementadas dentro de um mesmo escopo, seguindo as seguintes premissas:

- As recompensas dos *arms* eram puramente binárias. No caso de sucesso, o arm retornaria a recompensa 1, no caso de falha a recompensa 0;
- Os agentes poderiam escolher entre 7 *arms* distintos, com probabilidades de retorno da recompensa de 2,3%, 3%, 2,9%, 0,1%, 5%, 0,6% e 11% respectivamente;
- Em cada simulação os agentes teriam 10 000 tentativas para escolher cada um dos *arms*, recebendo o resultado da recompensa logo após a escolha do *arm;*
- Para os algoritmos que necessitavam de uma etapa inicial para definição de estimativa retorno dos *arms* (*UCB-1* e ϵ*-greedy*), foram alocadas as 300 tentativas iniciais da simulação para esta definição. Ao longo destas etapas, a seleção de *arm* é puramente aleatória;
- Para os algoritmos que possuíam parâmetros ajustáveis *(*ϵ*-greedy, Softmax e UCB-1*), foram realizadas simulações iniciais para ajustes de parâmetros e selecionar os que apresentavam o melhor resultado para comparação cruzada entre algoritmos. Cada simulação seguia as mesmas premissas apresentadas nos itens anteriores;
- O decaimento para o ϵ-greedy foi implementado na classe de forma linerar, caindo um valor percentual fixo em relação ao parâmetro ϵ adotado a cada 50 rodadas;

Para realizar as simulações foram desenvolvidas 7 classes *Python* distintas. A primeira classe denominada “Environment” define o ambiente em que as simulações serão realizadas. Ela recebe como entrada um primeiro vetor definindo as máquinas que serão utilizadas e um segundo vetor com a probabilidade de retorna da recompensa binária para cada uma delas .

Além disso, também é determinado o número de tentativas disponíveis para um agente trabalhar nas máquinas em uma mesma simulação como uma variável inteira.

A classe *Environment* possuía também um método definido como “Run”. Neste método, um dos agentes implementados através de outras classes escolhe um dos *arms*, recebe a recompensa baseada na probabilidade de retorno das mesmas e atualiza seus parâmetros para escolha da próxima máquina de acordo com a estratégia adotada.

A segunda classe é BaseSampler. Ela foi definida para agregar todos os parâmetros que seriam comuns a todos os agentes. Estes parâmetros são vetores com o valor da recompensa a cada execução do experimento, vetores com estimativa de regret para cada máquina ao longo do experimento, recompensa total acumulada, dentre outros.

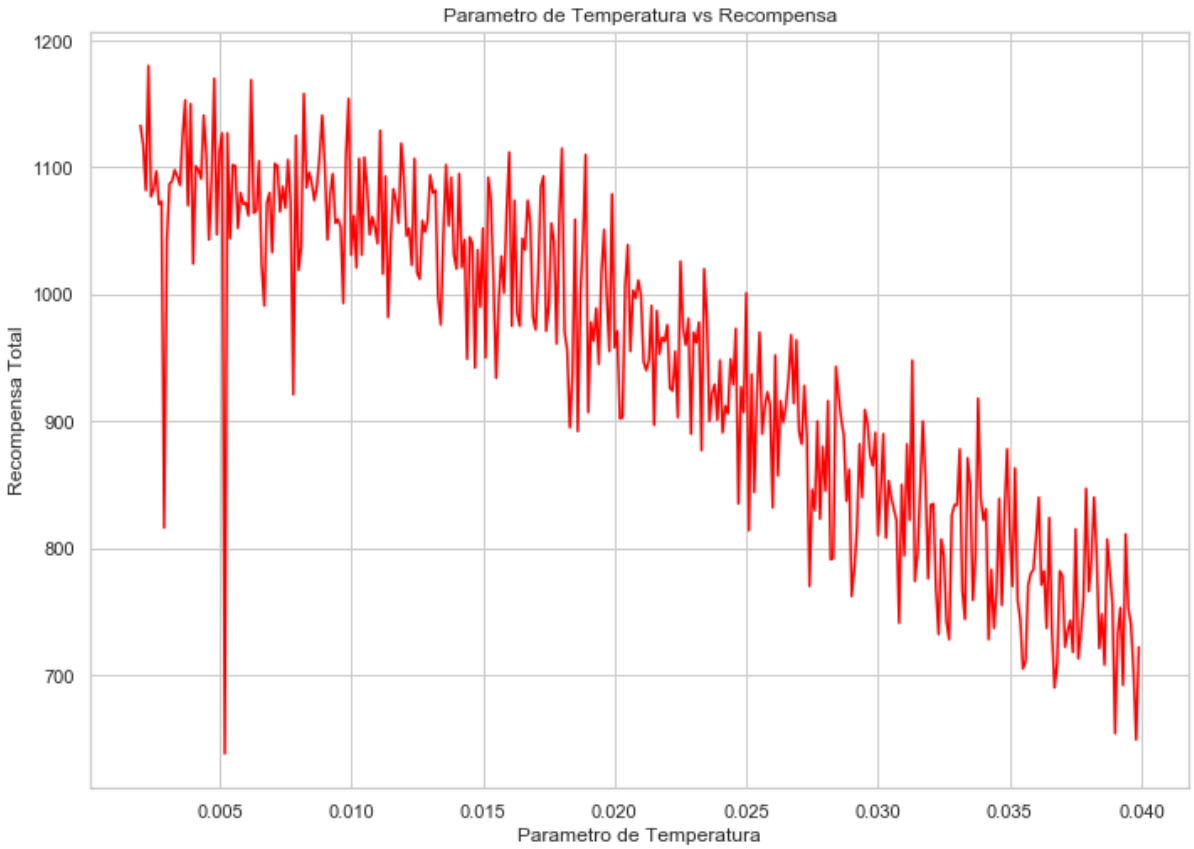
A partir da mesma foram criadas 5 outras classes, cada uma com uma estratégia para solução do problema Multi-Armed Bandit diferente: ϵ-greedy, implementando tanto o ϵ -greedy quanto o ϵ -greedy com decaimento, uma para o Softmax, uma para o UCB, uma para o Thompson Sampling e uma com uma escolha aleatória, como padrão de comparação, definida como RandomSampler.

5.1 Resultados das simulações de ajustes de Parâmetro

Para estabelecer um patamar aceitável de comparação entre os algoritmos, nesta etapa foi realiza uma otimização dos parâmetros de temperatura (τ) do *Softmax*, do parâmetro (ξ) do UCB-1 e da taxa de decaimento do ϵ-greedy (γ). A otimização foi implementada no contexto de simulação geral de modo a maximizar a recompensa final dada as condições experimentais adotadas.

As figuras abaixo mostram o resultado da recompensa em uma mesma simulação para um valor específico de cada um dos parâmetros:

Figura 2 – Simulação do Parâmetro de Temperatura (Algoritmo Softmax)

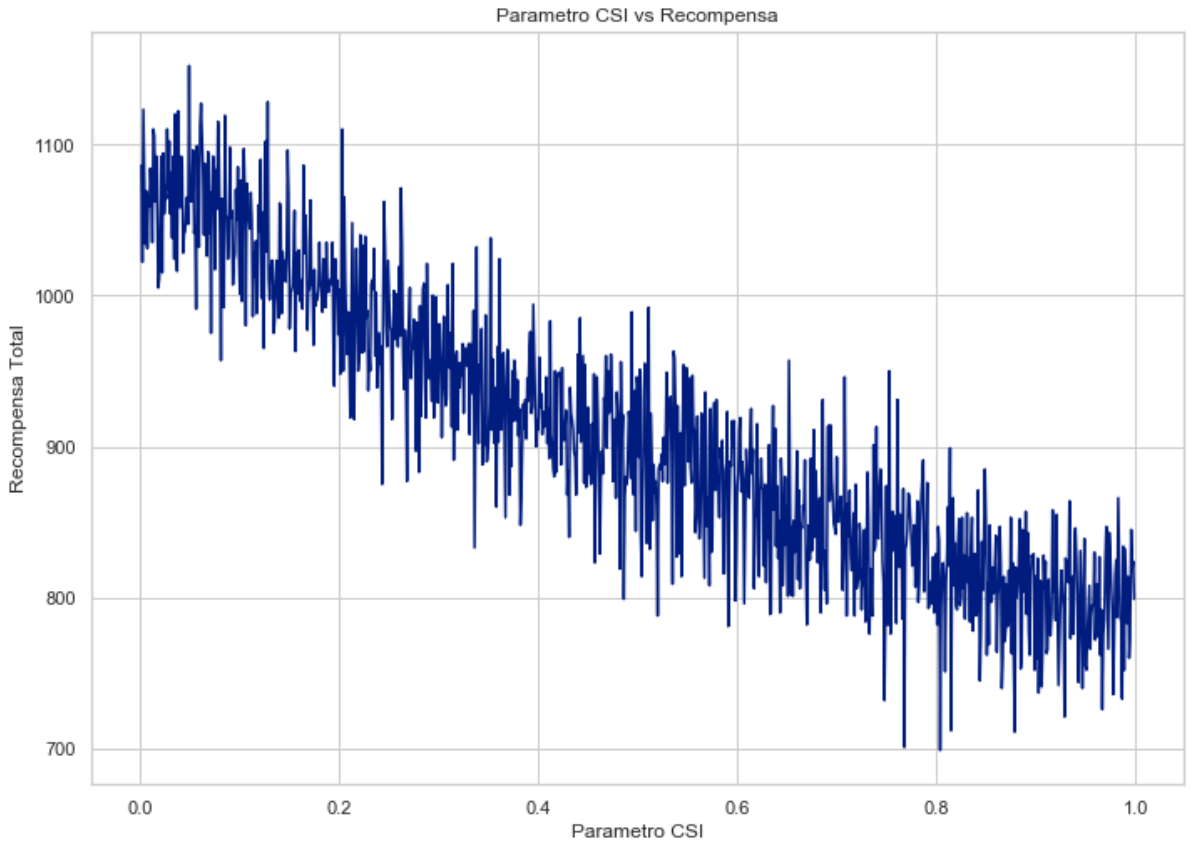


Pode-se que a partir de um parâmetro de temperatura de 0,01 a recompensa total diminui, indo de um patamar de 1100 para 700 por volta de 0,04. O aumento do parâmetro de temperatura leva a uma escolha mais uniforme dos *arms*, ignorando a informação de estimativa de probabilidade de retorno ao longo do experimento. Deste modo temos um perfil de escolha menos exploratório e mais experimental conforme o parâmetro é elevado.

Além disso, é notável em alguns caso à existência de alguns vales em que a recompensa atinge valores muito baixos, mesmo com parâmetros de temperatura menos elevados. Isso se deve ao fato da estimativa de retorno dos arms ser traduzida de forma direta em uma probabilidade de retorno para os mesmos através da função de Boltzman.

Como resultado o desempenho geral em uma mesma simulação pode diminuir caso os *arms* de menor probabilidade de retorno acabem gerando recompensas de forma constante nas etapas iniciais.

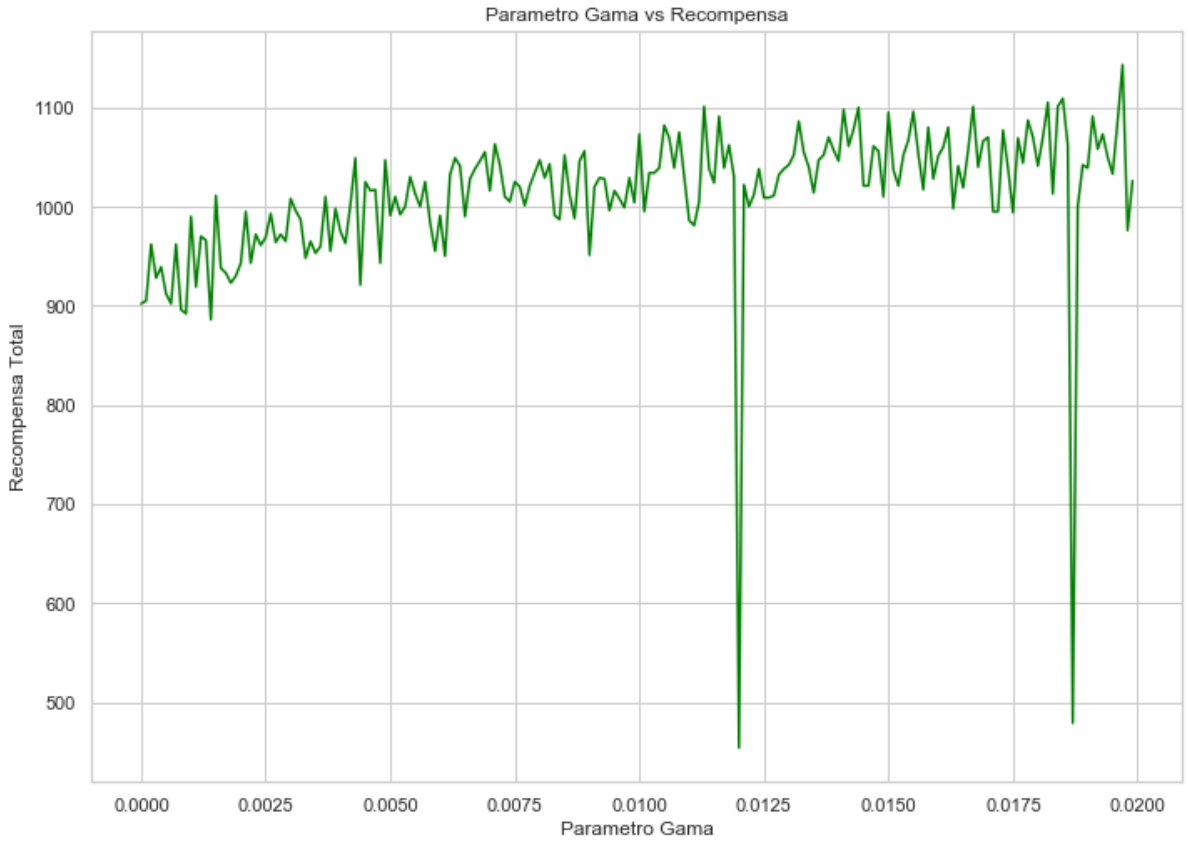
Figura 3 – Simuação do parâmetro ξ (Algoritmo UCB-1)



Verificamos para o UCB o mesmo padrão de diminuição de retorno da recompensa decrescente conforme o parâmetro ξ é aumentado, saindo de um patamar de 1050 para 800. O aumento nesse parâmetro leva a uma inflação na função de preenchimento, de modo que a incerteza gerada pelo número de escolhas de uma mesma máquina impacta mais que a sua estimativa de retorno. Nas etapas iniciais isto é positivo para garantir um perfil de experimentação adequado, mas pode ser prejudicial nas etapas finais quando a estimativa de retorno possui uma variância mais baixa.

Outro fator importante a ser observado é que o algoritmo é um pouco mais estável que o Softmax, não apresentando valores muito discrepantes de recompensa final em nenhuma simulação. O fato do mesmo ponderar a escolha dos *arms* baseado na frequência do seu uso e não apenas na sua recompensa impede que seja realizado um número excessivo de escolhas de um mesmo *arm* que não apresente um bom desempenho ao longo da simulação.

Figura 4 – Simulação do parâmetro γ (Algoritmo ϵ-greedy com decaímento)



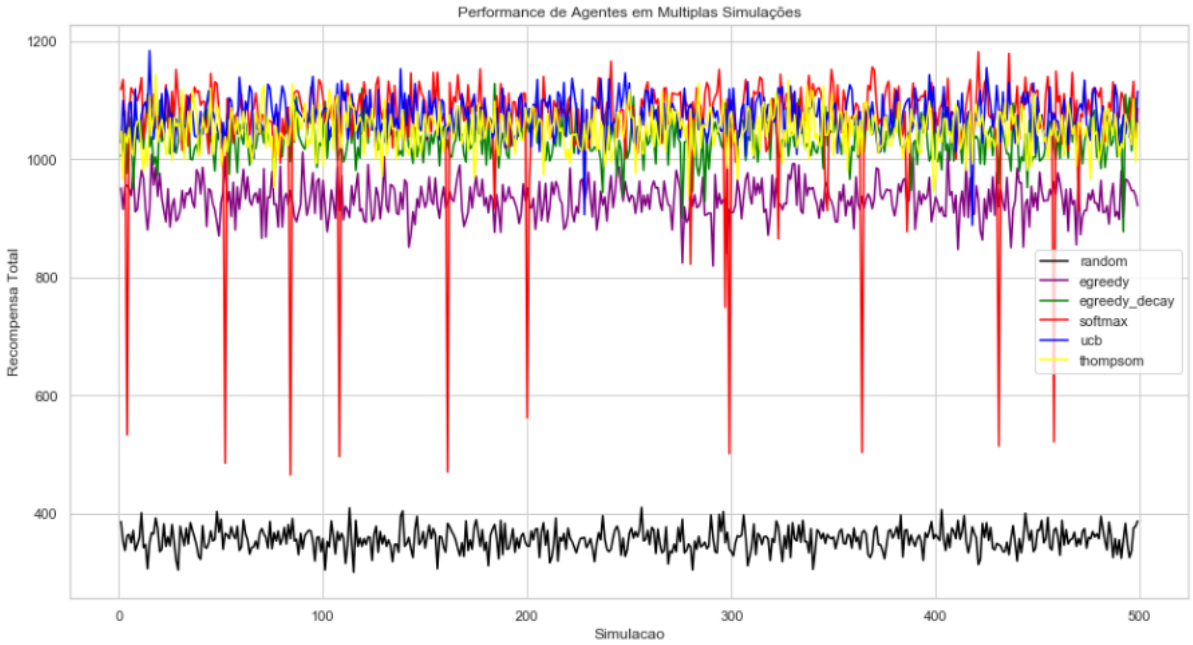
No caso do e-greey, o aumento na taxa de decaimento do parâmetro gama reflete em recompensas maiores. Neste caso este aumento reduz etapas puramente experimentais em que as máquinas são escolhidas sem nenhum critério, otimizando a escolha das mesmas. Os vales também estão presentes porque a escolha dos *arms* é diretamente atrelada à sua estimativa de retorno, sem nenhuma adaptação ao longo do experimento.

Ao final das simulações, adotaram-se os valores de 0,002 para o parâmetro de temperatura e de 0,01 para os parâmetros γ e ξ.

5.2 Comparação de Desempenho entre Algoritmos MAB

Nesta etapa foram realizadas 500 simulações para comparar o desempenho de múltiplos agentes, cujo resultado pode ser observado na figura 5:

Figura 5 – Recompensa Final de Múltiplos Estratégias ao Longo da Mesma Simulação



Inicialmente, podemos verificar que todas as estratégias possuem uma performance em geral duas vezes superior à estratégia puramente aleatória, que é a essencialmente a estratégia aplicada em um modelo tradicional de Testes AB. Isto indica que utilizar uma estratégia que administre experimentação e exploração em proporções razoáveis já permite obter ganhos muito superiores à estratégias aleatórias.

No quesito estabilidade, o Softmax apresentou a pior performance: apesar de ter a maior recompensa em múltiplas simulações a recompensa total chegou a cair para 600. Este resultado é extremamente ruim dado que a média dos resultados das simulações estava entre o intervalo de 1000 e 1200. O desempenho do ϵ-greedy apesar de superior à estratégia aleatória, ainda se encontrava muito abaixo do desempenho dos outros algoritmos.

Para aplicações práticas, em geral, é aceitável renunciar a um desempenho superior em desempenho para ter uma estabilidade maior ao longo do tempo. Neste aspecto, exclui-se o Softmax e o ϵ-greedy com decaimento como candidatos para a implementação prática na etapa seguinte.

A segunda análise observa tanto o perfil do regret ao longo da simulação quanto o regret acumulado ao final da mesma. Os gráficos abaixo apresentam essas duas métricas em 3 simulações distintas.

Figura 6 – Regret da Tentativa - Simulação 1

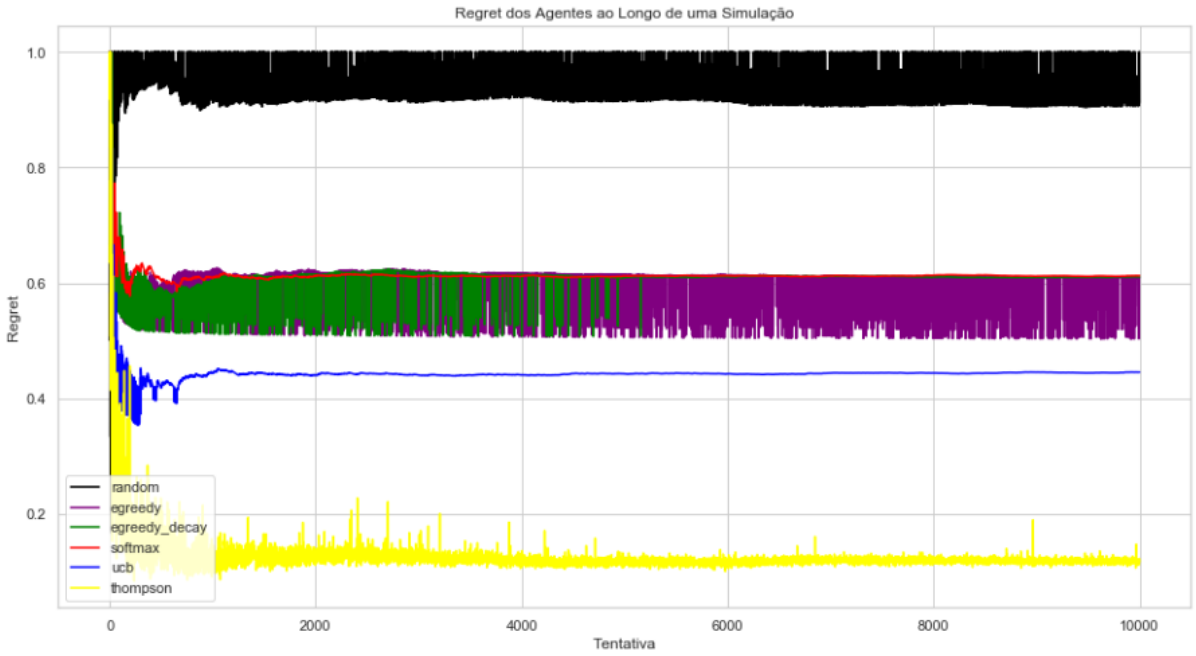


Figura 7 – Regret da Tentativa - Simulação 2

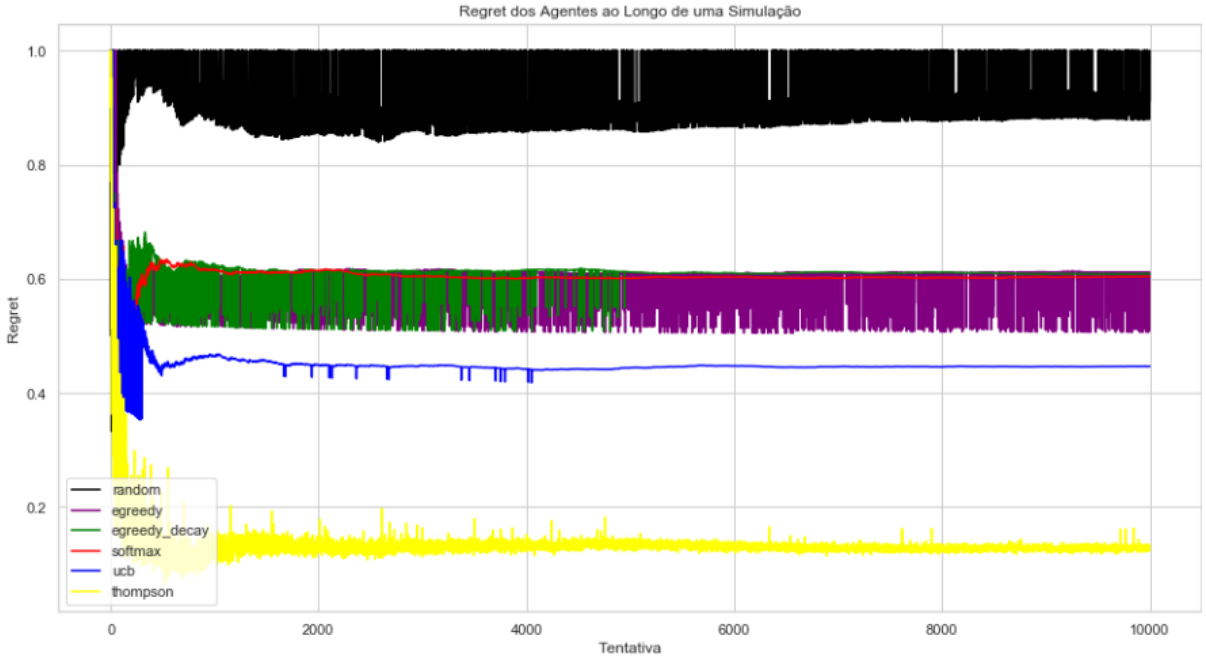


Figura 8 – Regret da Tentativa - Simulação 3

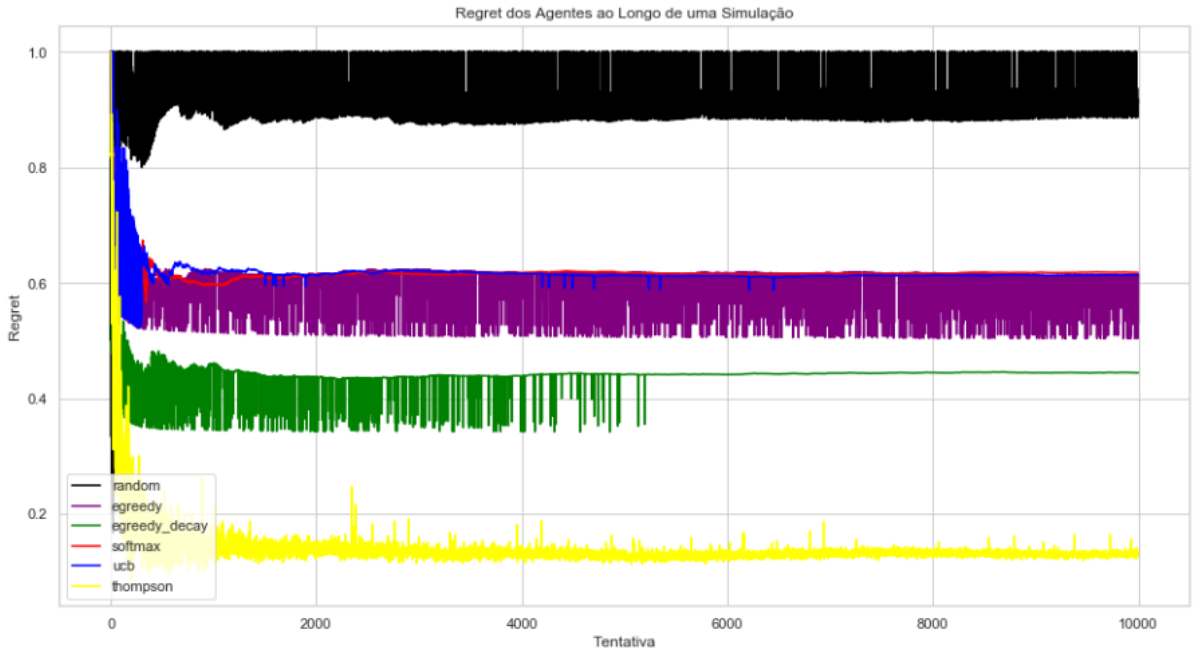


Figura 9 – Regret Acumulado - Simulação 1

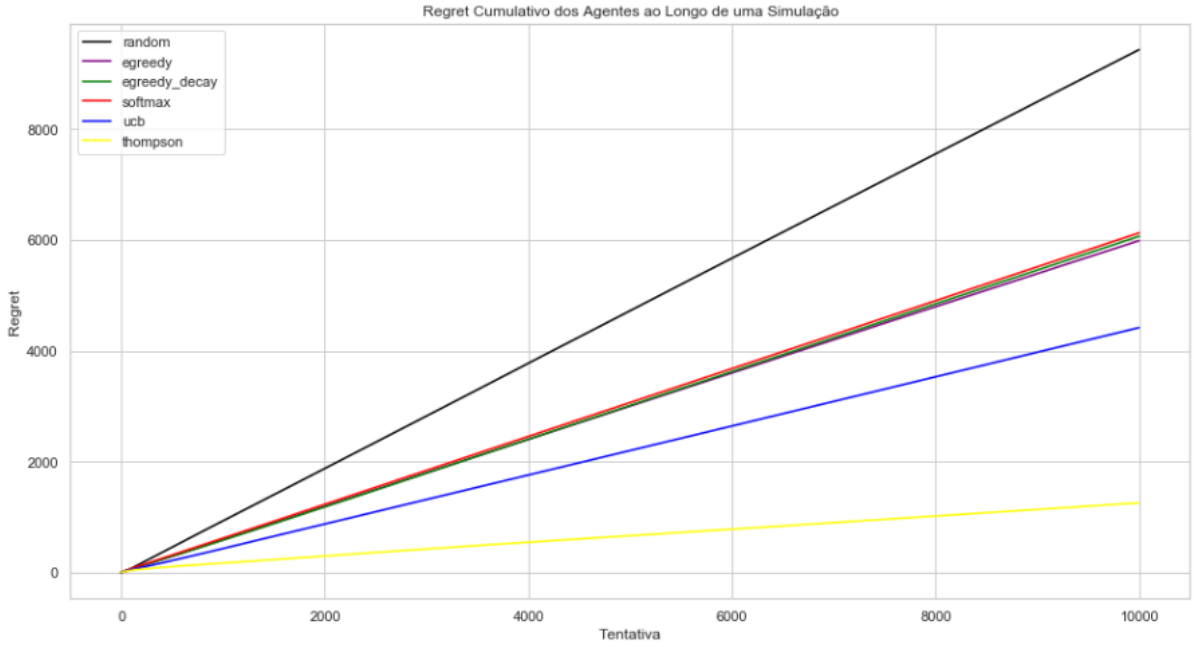


Figura 10 – Regret Acumulado - Simulação 2

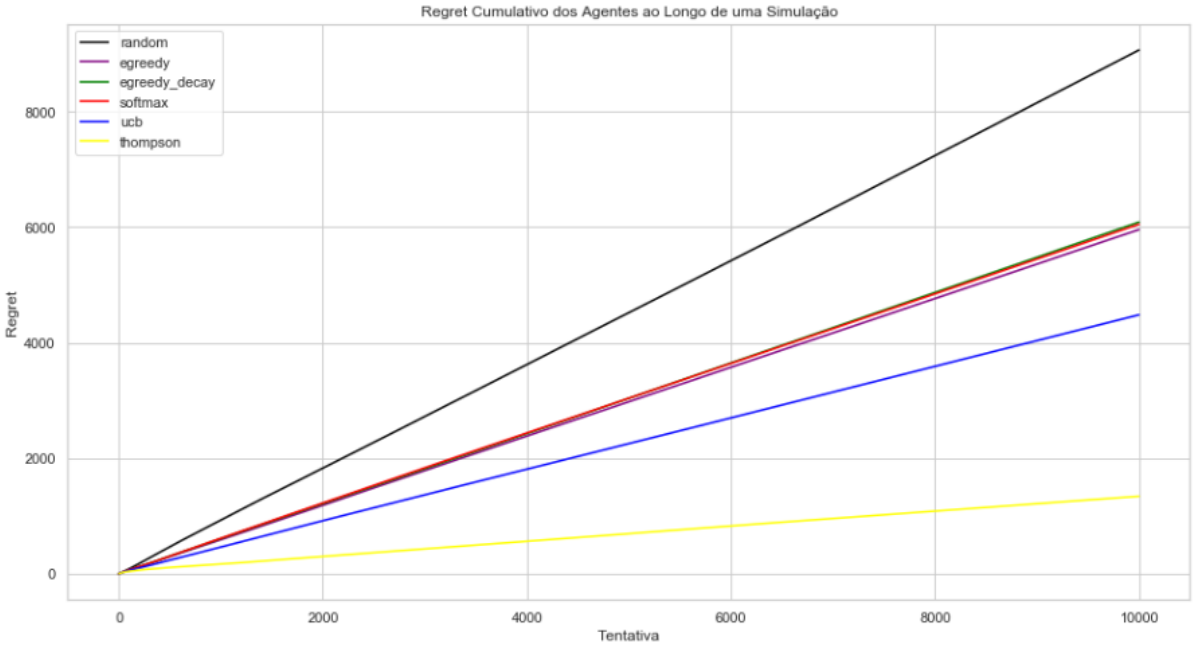
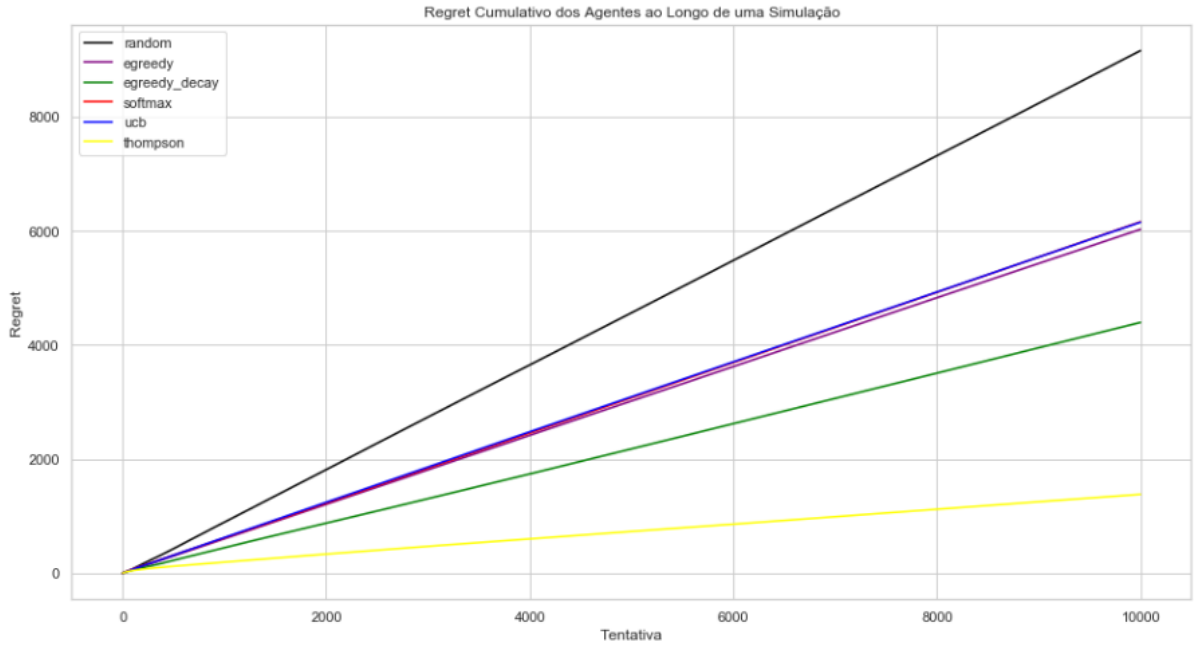


Figura 11 – Regret Acumulado - Simulação 3



Verifica-se que a estratégia aleatória também não possui nenhuma otimização de regret , quase sempre apresentando o valor máximo. A estratégia ϵ-greedy tem um regret que apesar de diminuir ao longo do experimento varia quase sempre em uma faixa muito ampla. Esta variação é corrigida ao implementar a estratégia de decaimento e converge a um valor final, no ponto em que zera-se o valor de ϵ.

Nos algoritmos softmax e UCB-1 o regre tem um decaimento rápido e uma convergência rápida, cujo valor pode variar muito dependendo do desempenho do agente ao longo da simulação. O algoritmo que melhor performa em termos do regret é

o Thompson. O decaimento do regret é extremamente rápido e é estável ao ponto que não possui uma dependência forte ao resultado isolado da simulado.

O *Thompson* de fato é o único que consegue combinar de forma simultânea a estimativa de retorno do arm com a variância da mesma. No *Thompson* a escolha do *arm* é feita com base em uma distribuição probabilística não uniforme, cuja variância vai diminuindo ao longo do experimento conforme os parâmetros da distribuição Beta e Alfa aumentam. Isto permite uma variação entre etapas experimentais/explorátórias muito mais otimizada e que confere ao algoritmo uma estabilidade maior também em termos do regret.

5.3 Aplicação Prática: Teste AB com e-mails

Nesta etapa apresentam-se os resultados da simulação com o teste de recomendação de produto em e-mail, utilizando o algoritmo Thompson Sampling. O teste foi feito em cima do mesmo *do mesmo* e-mail com duas ofertas de produtos distintas.

Os testes foram realizados em parceria com uma corretora de investimentos. O objetivo era divulgar produtos para alguns clientes, otimizando uma métrica de negócio própria da empresa, que era constantemente acompanhada após a realização do envio do e-mail.

O algoritmo Thompson Sampling implementado nas simulações foi transformado em uma classe *Python* e incorporada à biblioteca de desenvolvimento da equipe responsável pelos testes na empresa. A classe recebida os resultados da métrica de negócio e normalizava os valores em um intervalo de 0 a 1, para trabalhar em uma escala próxima à binária. No caso, desejava-se que o valor desta métrica fosse o menor possível.

Para fins da aplicação prática foram realizadas duas alterações no algoritmo. Uma vez que a abertura dos e-mails não ocorre necessariamente na data de envio, os parâmetros alfa e beta das versões AB eram atualizados diariamente, com base no resultado de resgates anteriores. Também foi adicionado um parâmetro de correção, para que a diferença entre os envios nos grupos não excedesse 10%. Deste modo, garantia-se que não houvesse uma disparidade grande no número de envios no início do experimento, dando uma boa base para avaliação do mesmo.

Diariamente, após os resultados dos envios dos e-mails anteriores serem estimados, os valores de alfa e beta eram calculados novamente. Para atualização somavam-se os valores da soma da recompensa normalizada para estimar alfa e o total de envios realizados para a mesma base para estimar os valores de beta.

Com base nesses parâmetros estimava-se um valor de theta utilizando uma distribuição beta para cada cliente, realizando a alocação com o valor máximo da mesma. O resultado foi avaliado em uma janela de 10 dias consecutivos, de 01 de abril até 10 de abril.

Figura 12 – Variação do Parâmetro Alfa ao Longo do Experimento

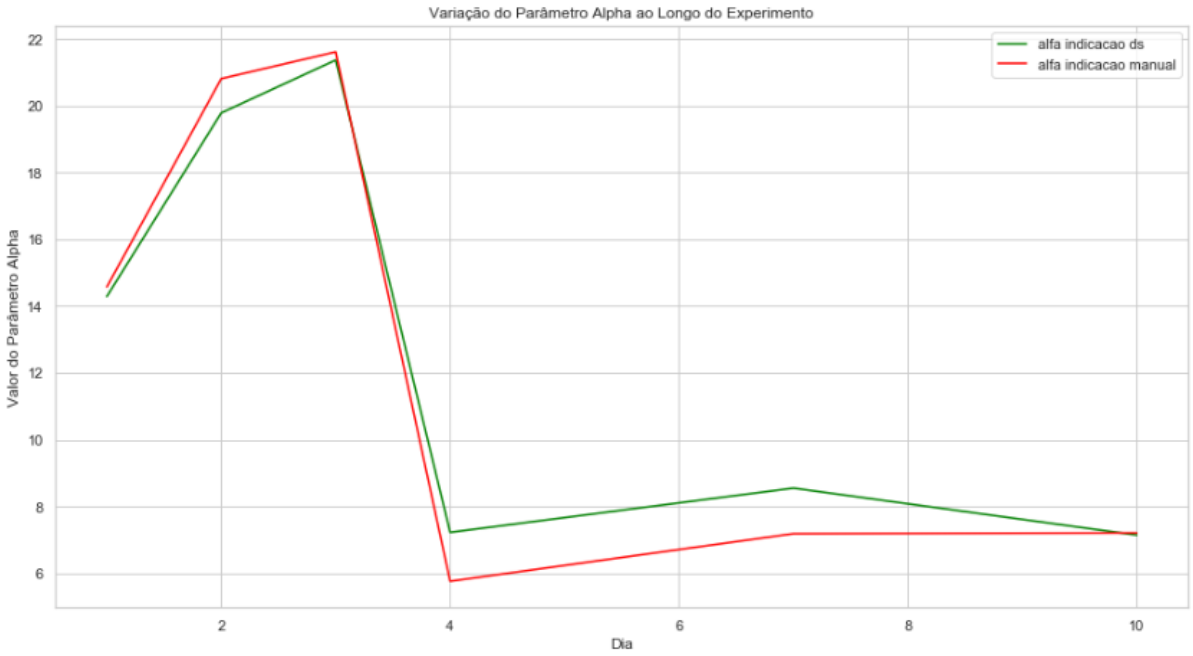


Figura 13 – Variação do Parâmetro Beta ao Longo do Experimento

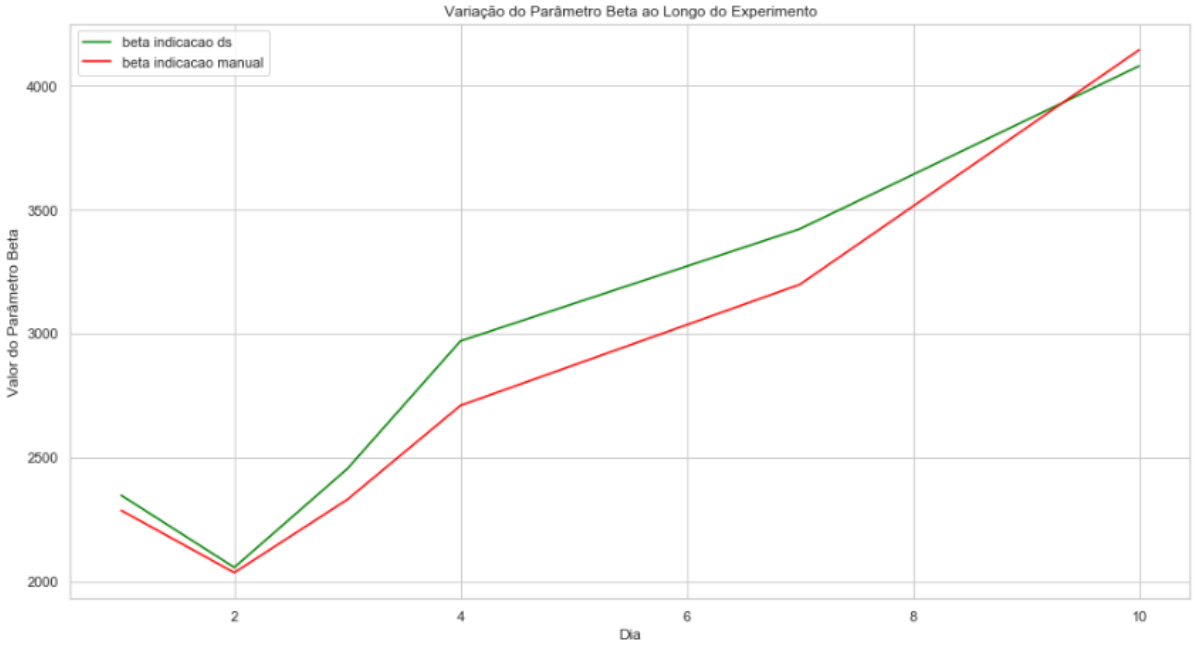
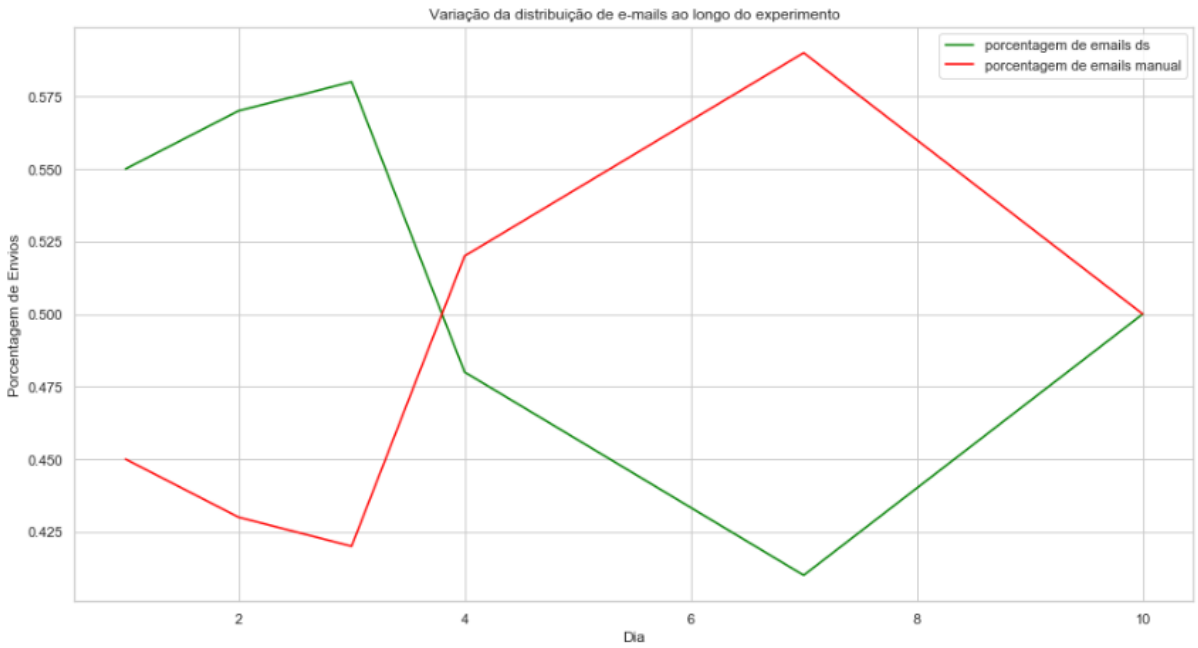


Figura 14 – Variação da Distribuição de e-mails para os grupos ao longo do Experimento



Pode-se observar que o parâmetro alfa começa em um patamar grande no início do experimento, caindo para um valor mais baixo após o quarto de dia. Este comportamento reflete o tempo de maturação para o início do cálculo da métrica de negócio, conforme avaliado pela empresa.

No que diz respeito à distribuição dos e-mails entre os grupos, pode ser observada uma distribuição similar dos e-mails entre os dois grupos, não havendo uma diferença significativa entre os mesmos.

6 Conclusão

Observou-se que os resultados de recomendação do modelo possuíam um grau elevado de semelhança. A empresa diagnosticou que o processo como os produtos eram selecionados possuíam uma grande similaridade entre si, o que explicou o resultado obtido

Em geral, grande parte dos testes AB executados via *e-mail* obtêem resultado inconclusivo, ainda que realizados disparos em massa para a base inteira. O *e-mail marketing,* além de ser uma ferramenta sujeita a taxa de entrega dos provedores, tende a perder a eficácia conforme o número de disparos para um mesmo cliente aumentam. Este efeito é maior ainda quando os e-mails não são 100% transacionais, como é o caso de e-mails com recomendação de produtos.

No caso do e-mail marketing, a implementação do Thompson Sampling encontra alguns empecilhos, dado o tempo necessário para maturação dos resultados e a baixa adesão ao canal de comunicação. A estratégia pode ser interessante para exibição de banners em canais de web e em dispositivos mobile em que as respostas dos usuários tendem a ser mais rápidas, por outro lado.

Outro fator natural que dificulta a implementação do Thompson Sampling é a integração dos dados com as ferramentas que realizam os envios de e-mail. A implementação de múltiplas integrações pode impactar a coleta dos dados uma vez que é preciso garantir que todas as etapas funcionem de forma correta.

Para aplicações de teste AB com menor tempo de experimento e variação mais dinâmica é possível realizar a implementação de algoritmos na própria interface do usuário. Neste modelo, realiza-se consultas no *backend* para estruturar a segmentação e envio das versões para os clientes. O problema dessas consultas é que elas aumentam o tempo de carregamento das páginas, sendo prejudiciais para o desempenho das mesmas.

Uma sugestão para implementações futuras de teste AB pela empresa foi adaptar o algoritmo para escolhas ao nível de cliente com novas recomendações de produto, ponderando a escolhas dos produtos recomendados com base nas características dos clientes.

Existem inúmeras possibilidades que podem ser exploradas, desde o oferecimento de produtos para segmentos específicos de clientes até o oferecimento de preços mais competitivos com base em interações menores. Aplicações mais complexas podem abrir espaço para combinação do Thompson com outros métodos de otimização, como regressões lineares, árvores de classificação, dentre outros.

Referências Bibliográficas:

[1] CUI, Geng; WONG, Man Leung; LUI, Hon-Kwong. Machine Learning for Direct Marketing Response Models: Bayesian Networks with Evolutionary Programming. **Management Science** v. 52, n. 4, p. 597–612 , 2006.

[2] BUBECK, Sébastien. Regret Analysis of Stochastic and Nonstochastic Multi-armed Bandit Problems. **Foundations and Trends® in Machine Learning**, vol. 5, no. 1, p. 1–122, 2012.

[3] ISHII, Shin; YOSHIDA, Wako and YOSHIMOTO, Junichiro. Control of exploitation–exploration meta-parameter in reinforcement learning. **Neural Networks**, vol. 15, no. 4-6, p. 665–687, 2002.

[4] SIROKER, Dan; KOOMEN, Pete and HARSHMAN, Cara. **A/B testing: the most powerful way to turn clicks into customers**. [s.l.]: John Wiley & Sons, Inc., 2015.

[5] AUER, Peter and ORTNER, Ronald. UCB revisited: Improved regret bounds for the stochastic multi-armed bandit problem. **Periodica Mathematica Hungarica**, vol. 61, no. 1-2, p. 55–65, 2010.

[6] KULESHOV, Volodymyr and PRECUP Dona. Algorithms for the multi-armed bandit problem. **Journal of Machine Learning Research,** vol 1, p. 1-48, 2000

[7] DEVANAND, R. and KUMAR, P. Empirical study of Thompson sampling: Tuning the posterior parameters. 2017.

[8] ROBBINS, Herbert. Some Aspects of the Sequential Design of Experiments. **Herbert Robbins Selected Papers**, p. 169–177, 1985.

[9] KOCH, Karl-Rudolf. **Introduction to Bayesian statistic**. [s.l.]: Springer, 2007.

[10] BELLMAN, Richard. A Markovian Decision Process. **Indiana University Mathematics Journal**, vol. 6, no. 4, p. 679–684, 1957.

[11] KAELBLING, Leslie; LITTMAN, Michael and MOORE, Andrew Reinforcement Learning: A Survey. **Journal of Artificial Intelligence Research**, vol 4, p. 237-285, 1996.

[12] LANGFORD, John. Efficient Exploration in Reinforcement Learning. **Encyclopedia of Machine Learning and Data Mining**, p. 389–392, 2017.

[13] NIEUWDORP, Thijs. **Dare to Discover: The Effect of the Exploration Strategy on an Agent's Performance**. 2017

[14] LAI, T.l and ROBBINS, Herbert. Asymptotically efficient adaptive allocation rules. **Advances in Applied Mathematics**, vol. 6, no. 1, p. 4–22, 1985

[15] GARIVIER, Aurélien and MOULINES, Eric. On Upper-Confidence Bound Policies for Switching Bandit Problems. **Lecture Notes in Computer Science Algorithmic Learning Theory**, p. 174–188, 2011.

[16] AUDIBERT, Jean-Yves; MUNOS, Rémi and SZEPESVÁRI, Csaba. Tuning Bandit Algorithms in Stochastic Environments. **Lecture Notes in Computer Science Algorithmic Learning Theory**, p. 150–165, 2007.

[17] BAYES. An Essay Towards Solving A Problem In The Doctrine Of Chances. **Biometrika**, vol. 45, no. 3-4, p. 296–315, 1958.

[18] CARLIN, Bradley P.; LOUIS, Thomas A. and CARLIN, Bradley P. **Bayesian methods for data analysis**. [s.l.]: CRC Press, 2009.

[19] THOMPSON, William R. On the Likelihood that One Unknown Probability Exceeds Another in View of the Evidence of Two Samples. **Biometrika**, vol. 25, no. 3/4, p. 285, 1933.

[20] GOPALAN, Aditya; MANNOR, Shie and MANSOUR Yishay. **Thompson Sampling for Complex Online Problems**, 2014.

[21] GUPTA, A. K. and NADARAJAH, Saralees. **Handbook of beta distribution and its applications**. [s.l.]: Marcel Dekker, 2004.