FLÁVIO SALES TRUZZI

Modelagem e Soluções para Redes de Anúncios

São Paulo
2014

FLÁVIO SALES TRUZZI

Modelagem e Soluções para Redes de Anúncios

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Ciências.

São Paulo
2014

FLÁVIO SALES TRUZZI

Modelagem e Soluções para Redes de Anúncios

Dissertação apresentada à Escola
Politécnica da Universidade de São Paulo
para obtenção do título de Mestre em
Ciências.

Área de Concentração:
Engenharia de Computação

Orientadora: Prof. Dra. Anna Helena Reali
Costa

São Paulo
2014

FICHA CATALOGRÁFICA

Truzzi, Flavio Sales

Modelagem e soluções para redes de anúncios / F.S. Truzzi.

-- São Paulo, 2014.

80 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Inteligência artificial 2.Processos de Markov 3.Aprendizado por reforço 4.Redes de anúncios I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a minha orientadora, Prof. Dra. Anna Helena Reali Costa, por ter confiado na minha capacidade, me aceitando como orientado durante as minhas duas iniciações científicas e durante este trabalho de mestrado, exigindo sempre o máximo de seus alunos e me inspirando sempre a ser um melhor pesquisador e pessoa.

Gostaria de agradecer ao Prof. Dr. Fábio Gagliardi Cozman, que me apresentou este problema, me ajudando a definir o escopo deste trabalho, participando em minhas publicações, sempre com conselhos extremamente pertinentes.

Agradeço ao Prof. Dr. Valdinei Freire da Silva, por sua paciência e contribuição para esta pesquisa, sendo que sem ele ela não seria possível.

Agradeço a todos os membros do Laboratório de Técnicas Inteligêntes da USP, pelas discussões construtivas e um ambiente amigável durante todo este trabalho de pesquisa.

Agradeço a empresa Yahoo, por ceder uma base de dados extremamente rica que foi utilizada para a avaliação e comparação de um dos algoritmos propostos neste trabalho.

Agradeço a minha namorada Mariana, pelo carinho, amizade, paciência, apoio e pelas revisões que eu a forcei fazer.

Expresso minha gratidão especial a minha família, minha mãe, Vera Lucia Sales Truzzi, meu pai, Claudio Roberto Truzzi, meus irmãos Vitor Sales Truzzi e Rafael Sales Truzzi, pelo apoio quando tomei a decisão de fazer o mestrado, e sua contribuição emocional e financeira.

Por fim, agradeço a CAPES pela bolsa de mestrado que foi fundamental para a execução deste trabalho de mestrado.

“The best way to predict your future
is to create it!”

Abraham Lincoln

RESUMO

Redes de Anúncios (*Ad Networks*) são redes que promovem a distribuição de anúncios pela internet, de forma a maximizar o lucro total gerado pela exibição dos anúncios nos *websites*. Estas redes tipicamente operam através do modelo de negócios chamado CPC (Custo por Clique), em que o anunciante paga um determinado valor somente se algum usuário clicar em seu anúncio. A escolha de como o intermediador planeja a distribuição dos anúncios aos *websites* é de extrema importância, já que a taxa de cliques nos anúncios é extremamente baixa. Atualmente a alocação dos anúncios tem sido feita através de uma solução aproximada baseada na alocação ótima definida com dados de um período anterior, a qual é calculada através de programação linear aliada à utilização de heurísticas. Entretanto, este sistema claramente é um processo de decisão sequencial em que diversas restrições são aplicáveis, como por exemplo: o orçamento dos anunciantes, limites mínimos do número de exibições de cada anúncio, categorias dos anúncios, entre outras. Neste trabalho argumenta-se que MDPs (*Markov Decision Processes*) fornecem uma melhor modelagem para o problema, já que conseguem levar em conta a dinâmica do sistema, considerando, por exemplo, que um anúncio que tem poucas chances de ser clicado consiga ser alocado de forma eficiente em relação ao retorno de longo prazo, mesmo quando outros anúncios proveriam um lucro maior a curto prazo. No entanto, devido ao grande número de estados, utilizar uma solução ótima através de MDPs é impraticável. Portanto analisa-se o desempenho relativo entre o estado da arte e a modelagem ótima, obtendo garantias de que a solução aproximada baseada em programação linear não está longe da solução ótima, e que em problemas grandes (similares aos encontrados na prática) essa diferença pode ser ignorada. Por fim, propõe-se uma modelagem baseada em aprendizado por reforço para a solução deste problema, utilizando duas abordagens, uma desconsiderando informações de contexto e outra considerando informações de contexto. Aqui argumenta-se que o uso de aprendizado por reforço é mais apropriado para a solução do problema de alocação de anúncios, já que ele é capaz de adaptar sua política de alocação em função das mudanças que ocorrem como, por exemplo, no perfil do usuário.

Keywords: Inteligência Artificial, Processos de Decisão Markov, Aprendizado por Reforço, Redes de Anúncios.

ABSTRACT

Ad Networks promote the distribution of ads in the internet, so as to maximize the revenue generated by their display of ads in websites. These networks typically operate using the CPC (Cost per Click) business model, where the advertiser pays a monetary value when a user clicks in its advertisement. The choice of how the Ad Network distributes ads to websites is of utmost importance, since the rate of clicks on ads is extremely low. The allocation of ads has been done by an approximate solution based on data from an early period of time, which is calculated using linear programming combined with heuristics. However, this problem is clearly a sequential decision process in which multiple sequential restrictions apply, such as: the budget of the advertisers, minimum limits on the number of views for each campaign, categories of advertisements. In this dissertation we argue that MDPs (Markov Decision Processes) provide a better model for the problem, since they can automatically take into account the dynamics of the system, considering, for example, an ad with little chance of being clicked can be allocated in an efficient way, even when other ads would provide a higher profit in the short term. However, due to the large number of states, an optimal solution through MDPs is impractical; therefore we analyze here the relative performance between the linear programming and the MDP approaches, deriving guarantees that the approximate solution based on linear programming is not far from the MDP optimal solution, and in large problems (similar to those found in practice) this difference can be disregarded. Finally, we propose a model based on reinforcement learning using two different approaches, one disregarding the contextual information, and the other using contextual information. We argue that the use of reinforcement learning is more suitable for solving the problem of allocation of ads, since it is able to adapt its allocation policy to reflect changes that occur, e.g., in the user profile.

Keywords: Artificial Intelligence, Markov Decision Processes, Reinforcement Learning, Ad Networks.

LISTA DE FIGURAS

2.1	Fluxograma da distribuição de anúncios.	22
3.1	Exemplo de Intervalos para 3 campanhas, C1, C2 e C3.	29
4.1	O ciclo de percepção-ação – O agente observa o seu estado atual, executa uma ação, recebe uma recompensa e observa um novo estado.	32
4.2	Alguns exemplos de próximos estados ao executar a ação 1 no estado $s = [B_1, B_2, 1]$	37
5.1	Possíveis configurações com um usuário e duas campanhas: (i) campanhas disjuntas, (ii) campanhas parcialmente sobrepostas e (iii) uma campanha totalmente sobreposta a outra.	40
5.2	Desempenho relativo da solução por MDP em relação à solução baseada na formulação por programação linear, para $\mathbb{T}_2 = 50.000$ (cenários 2 ao 4) e para o caso descrito no cenário 1, com orçamentos variando de 1 até 50.	44
5.3	Desempenho relativo da solução ótima em relação à solução baseada na formulação por programação linear, para $\mathbb{T}_2 = 50,000$ (cenários 2 ao 4) e para o pior caso descrito no cenário 1 com orçamentos variando de 150 até 200.	45
6.1	Ciclo de atuação de um agente modelado por aprendizado por reforço.	49
6.2	Densidade de Probabilidade da Função Beta para diferentes parâmetros de α e β	56
6.3	Arrependimento médio para um problema com horizonte igual a 15.000 tomadas de decisão para os algoritmos: <i>AdBandit</i> , <i>Thompson Sampling</i> , Bayes-UCB e UCB1, médias calculadas sobre 1.000 simulações.	61
6.4	Arrependimento médio para um problema com horizonte igual a 15.000 tomadas de decisão para os algoritmos: <i>AdBandit</i> , <i>Thompson Sampling</i> e <i>Bayes-UCB</i> ; médias calculadas sobre 1.000 simulações.	62
6.5	Distribuição do arrependimento considerando um experimento com horizonte de 15.000 tomadas de decisão, valores calculados sobre 1.000 simulações.	63
6.6	Conjunto de <i>Bandit</i> . A cada requisição, verifica-se o perfil do usuário, e utiliza-se um <i>bandit</i> específico.	64

6.7	<i>Today Module</i> extraído de http://us.yahoo.com , visita em: 18 de fevereiro de 2014.	70
6.8	Comparação de desempenho medido em <i>CTR</i> entre os algoritmos LBP, TSCb, e ETS para os primeiros 9,31 milhões de registros da base de dados <i>R6B - Yahoo! Front Page Today Module User Click Log Dataset, version 2.0</i>	72

LISTA DE TABELAS

1.1	Receitas por tipo de mídia	18
5.1	Desempenho relativo da solução por MDP em relação à solução baseada na formulação por programação linear, para $\mathbb{T}_2 = 50.000$ (cenários 2 ao 4) e para o cenário 1 ($\mathbb{T}_2 \rightarrow \infty$) e $B_1 = 500$	45
6.1	Probabilidades utilizadas no experimento.	60
6.2	Informação do primeiro registro da base de dados do <i>Today Module</i>	71
6.3	Avaliação de algoritmos simples para a base de dados sobre os 9,310 milhões de registros.	72

LISTA DE ALGORITMOS

1	Algoritmo Simplex	26
2	Iteração de Valor	33
3	Algoritmo ϵ -greedy(Q, s, ϵ)	49
4	Policy Evaluator (com horizonte finito)	53
5	<i>Thompson Sampling</i> para MABs modelados por distribuições de Bernoulli .	56
6	<i>Thompson Sampling</i> para <i>Bandits</i> modelados por distribuições estocásticas em geral	57
7	Algoritmo UCB1	57
8	Algoritmo Bayes-UCB	58
9	Algoritmo Adbandit	60
10	<i>Thompson Sampling for Contextual Bandits</i>	65
11	<i>Extended Thompson Sampling</i> para MABs modelados por distribuições es- tocásticas (ETS).	69

LISTA DE ABREVIATURAS

CPI	Custo por Impressão
eCPI	Custo efetivo por impressão
CPA	Custo por Ação
CTR	Probabilidade de clique
CPC	Custo por Clique
MC	Monte Carlo
MDP	Processo de Decisão Markoviano
MAB	<i>Multi-Armed Bandit</i>
RTB	<i>Real Time Bidding</i>
TS	<i>Thompson Sampling</i>
LP	Programação Linear (<i>Linear Programming</i>)
HLP	<i>The Highest LP Policy</i>
SLP	<i>The Stochastic LP Policy</i>
UCB	Algoritmo <i>Upper Confidence Bound</i>
UCB-Bayes	Algoritmo UCB-Bayes
LBP	<i>Linear Bayes Policy</i>
TSCb	<i>Thompson Sampling for Contextual bandits</i>
ETS	<i>Extended Thompson Sampling</i>

LISTA DE SÍMBOLOS

P_{req}	Probabilidade de ocorrência de uma requisição
$\mathcal{G}$	Conjunto de perfis de usuários
$P_G(i)$	Função de probabilidade de um usuário pertencer um a um determinado perfil de usuário i
$\mathcal{C}$	Conjunto de campanhas de anúncios
B_k	Orçamento da campanha k medido em cliques
S_k	Instante de tempo correspondente ao início de operação da campanha k
L_k	Tempo de vida de uma campanha k
cc_k	Custo por clique, valor monetário que uma campanha k paga por clique
$\vec{x}$	Vetor de atributos de usuário
$CTR(i, k)$	Probabilidade de clique entre o perfil de usuário i e a campanha k
$CTR(x_i, k)$	Probabilidade de clique associado a um atributo $x_i \in \vec{x}$ e uma campanha k
$u(\vec{x})$	Usuário u dado por um vetor de atributos
$\mathcal{S}$	Espaço de estados
$\mathcal{A}$	Espaço de ações
$\mathcal{D}$	Sequência finita de números naturais, denotando épocas de decisão
$\mathcal{T}$	Função de transição que especifica a probabilidade $\mathcal{T}(s, a, s', t)$ de se transitar de um estado s para um estado s' , quando executada a ação a no instante de tempo t
$\mathcal{R}$	Função de recompensa que retorna uma recompensa numérica finita $r = \mathcal{R}(s, a, s', t)$ quando o sistema vai do estado s para o estado s' como resultado da execução da ação a na época de decisão t
π	Política determinística não estacionária

$V^\pi(s, t)$	Valor de uma política π
π^*	Política determinística não estacionária ótima
τ	Horizonte, número de épocas de decisões, tal que $\tau = \mathcal{D} $
$\max$	Máximo de uma função, ou argumento
$\min$	Mínimo de uma função, ou argumento
$\arg \max_x$	Argumento de x da função que maximiza alguma função
$\arg \min_x$	Argumento de x da função que minimiza alguma função
$s.t.$	Sujeito à
c	Vetor de coeficientes de uma função linear
c^T	Transposto do vetor ou matriz c
A	Matrix de coeficientes
$r(a, t)$	Recompensa da ação a no instante de tempo t
$\mathcal{B}(\mu)$	Amostra de uma distribuição de Bernoulli com valor esperado μ
$\mathcal{R}^{-1}$	Arrependimento acumulado
$N_{a, \tau}$	Variável aleatória que representa o número de vezes que uma ação a foi escolhida até o instante de tempo $t = \tau$
$g(\vec{x}, a_t)$	Recompensa associada a um vetor de atributos de usuário e uma ação a tomada no instante t
θ	Estimativa de probabilidades de sucesso
θ_a	Estimativa de probabilidade de sucesso associado à ação a
μ	Probabilidade de sucesso
μ_a	Probabilidade de sucesso associado à ação a
ψ^t	Distribuição a priori que modela as ações no instante de tempo t
ψ_a^t	Distribuição a priori que modela a ação a no instante de tempo t
$\nu_{\theta_a}(r_t)$	Distribuição que modela a função de recompensa r_t

$Beta(\alpha, \beta)$	Distribuição Beta com parâmetros α e β
$B(\alpha, \beta)$	Função beta com parâmetros α e β
$\Gamma(\cdot)$	Função gamma
$\delta(\cdot, \cdot)$	Divergência de Kullback-Leibler
G_A	Recompensa acumulada esperada de um algoritmo de <i>multi armed-bandits</i> A
$\mathbb{E}_D$	Valor esperado definido de acordo com um processo independente e identicamente distribuído com uma distribuição D
h_t	História de uma avaliação de um algoritmo de <i>multi armed-bandit</i>
$\hat{G}_A$	Valor esperado de recompensas durante a avaliação de um algoritmo de <i>multi armed-bandit</i>
T	Contador de eventos válidos durante a avaliação de um algoritmo de <i>multi armed-bandit</i>
$P_{groupSite}$	Função de probabilidade de um usuário pertencer a um determinado grupo
$\bar{h}_i$	Valor médio de alocações feitas para um determinado grupo i
$\mathcal{I}$	Lista ordenada dos inícios e términos de todas as campanhas de anúncios
$\mathcal{J}_j$	Intervalos definidos pelas restrições de tempo da campanha j
$\mathbb{T}_j$	Intervalo de tempo de uma campanha j
sup	Função supremo
inf	Função ínfimo
E_k	Instante de tempo que marca o fim do tempo de operação da campanha k
e	Número de Euler
$\gg$	Muito maior
ln	Função logaritmo natural
$Q(p, \alpha, \beta)$	Função quantil para a probabilidade p da distribuição Beta com parâmetros α e β
S_a	Contagem do número de sucessos da ação a

F_a	Contagem do número de falhas da ação a
$\hat{\mu}_a$	Aproximação da probabilidade de sucesso da ação a
$P(a)$	Probabilidade de clique de uma campanha a
$P(a i)$	Probabilidade de clique de uma campanha a dado um perfil de usuário i
$\mathcal{N}$	Distribuição gaussiana
$P(x_i)$	Probabilidade de ocorrência de um atributo de contexto x_i
$P(x_i a)$	Probabilidade de ocorrência de um atributo de contexto x_i dado uma ação a
$\mathcal{F}$	Base de conhecimento de ações
$\mathcal{X}$	Base de conhecimento de atributos de contexto
$\mathcal{X}_a$	Base de conhecimento de atributos de contexto associado a ação a
ρ_{a,x_i}^{t-1}	Distribuição Beta que modela o comportamento de um atributo x_i associado a uma ação a

SUMÁRIO

1	Introdução	18
1.1	Objetivos	19
1.2	Contribuição	19
1.3	Organização do Texto	20
2	Descrição do Problema	21
2.1	Modelo básico	21
2.2	Extensão ao modelo básico	23
3	Redes de Anúncios	25
3.1	Programação Linear	25
3.2	Panorama atual das soluções para Redes de Anúncios	26
3.3	Discussão	30
4	Redes de Anúncios como um Processo de Decisão Markoviano	31
4.1	Processos de Decisão Markovianos	31
4.2	Redes de Anúncios como um MDP	34
4.2.1	Estados	34
4.2.2	Ações	35
4.2.3	Função de Transição	35
4.2.4	Função de Recompensa	37
5	Diferenças entre a Solução Ótima e o Estado da Arte	39
5.1	Construção de um caso ruim para a solução por Programação Linear . . .	40
5.2	Experimentos	43
5.3	Diferenças entre a complexidade assintótica das soluções	46
5.4	Considerações Finais	46
6	Redes de Anúncios como um problema de <i>Multi-Armed Bandit</i>.	48
6.1	Aprendizado por Reforço	48
6.1.1	<i>Multi-Armed Bandits</i> (MAB)	50
6.2	Avaliação entre diferentes algoritmos para <i>Multi-Armed Bandits</i> com contexto	52
6.3	Redes de Anúncios como um MAB sem informação de contexto	53
6.3.1	Thompson Sampling	54

6.3.2	<i>Upper Confidence Bound</i> (UCB1)	57
6.3.3	Bayes-UCB	58
6.3.4	Algoritmo <i>AdBandit</i>	58
6.3.5	Comparação entre os algoritmos de MAB	59
6.4	Redes de Anúncios como um MAB com informação de contexto	63
6.4.1	<i>Thompson Sampling for Contextual Bandits</i>	64
6.4.2	<i>Linear Bayes Policy</i>	65
6.4.3	<i>Extended Thompson Sampling</i> (ETS)	66
6.5	Comparação entre algoritmos de MAB com contexto	69
7	Conclusão e Trabalhos Futuros	74
	Referências Bibliográficas	76

1 INTRODUÇÃO

As receitas com publicidade na internet têm crescido rapidamente desde seu início em meados da década de 90. De 2003 até 2013 a taxa composta de crescimento anual foi de 20%. Nos primeiros seis meses de 2013 as receitas totalizaram, apenas nos Estados Unidos da América, \$20,1 bilhões de dólares, um crescimento de 17,8% sobre os seis primeiros meses de 2012 (Internet Advertising Bureau, 2013).

Este mercado é dividido entre mídias diferentes, sendo que as principais em volume de receitas são descritas na Tabela 1.1 (Internet Advertising Bureau, 2013).

Tabela 1.1 – Receitas por tipo de mídia

Mídia	Receita (%)
<i>Search</i>	43%
<i>Banners</i>	19%
<i>Mobile</i>	16%
Outros	22%

A apresentação de anúncios através de *banners* em *websites* é um dos modos de publicidade mais comuns, possuindo 19% do mercado (Internet Advertising Bureau, 2013), mantendo aproximadamente esta mesma fatia de mercado desde 2006 (Internet Advertising Bureau, 2013).

As empresas de publicidade digital trabalham em sua maioria utilizando duas formas básicas de modelo de negócio: um modelo *online* ou um modelo *offline*. Em ambos os modelos existe um intermediador que possui contratos com *websites*, que são aqueles que possuem os espaços onde os *banners* serão exibidos.

O modelo *online* é representado pelo *Real Time Bidding*; neste modelo os anunciantes participam de um leilão, normalmente seguindo o modelo de Vickrey (AUSUBEL; MILGROM, 2006), onde o valor a ser pago pelo ganhador corresponde ao valor que o segundo colocado estaria disposto a pagar. Os anunciantes neste modelo dão lances para comprar uma exibição de anúncio para um usuário específico.

No modelo *offline*, por outro lado, os anunciantes fazem um contrato com o intermediador, doravante chamado de Rede de Anúncios. Os anunciantes criam campanhas especificando um conjunto de anúncios, um modelo de precificação, seu orçamento, seu número mínimo de impressões (i.e. o número mínimo de vezes que a campanha deve ser vista por usuários) e suas restrições de tempo (início e fim da campanha). A Rede de

Anúncios decide como distribuir os anúncios para os usuários de forma a maximizar a receita da rede, respeitando as restrições impostas pelos anunciantes. Este trabalho trata apenas do modelo de negócio *offline*.

Existem diversos modelos de precificação (CHATTERJEE; HOFFMAN; NOVAK, 2003; NOVAK; HOFFMAN, 1997). Os mais utilizados no mercado são o Custo por Impressão (CPI), o Custo por Ação (CPA), e o Custo por Clique (CPC). No modelo CPI, o anunciante paga pelo número de impressões de sua campanha, neste trabalho uma impressão corresponde a uma exibição de um anúncio em um *website*. No modelo CPA, o anunciante paga caso o usuário faça uma determinada ação, como por exemplo: preencher um formulário ou fazer uma compra na loja do anunciante. No modelo CPC, o anunciante paga apenas quando um usuário clicar em seu anúncio. Este trabalho foca o modelo de precificação CPC.

O modelo de negócio *offline* é, em sua essência, um processo de decisão sequencial em que a Rede de Anúncios tem que decidir qual campanha será exibida para cada usuário em um momento específico do tempo, observando as condições de orçamento, quanto cada campanha paga por clique, e a relação entre cada campanha e o perfil do usuário.

Apesar de alguns trabalhos citarem que o problema de alocação de anúncios para usuários em *websites* seja um processo de decisão sequencial (GIRGIN et al., 2012, 2010a), não havia na literatura, até onde se tem conhecimento, um modelo que captasse a dinâmica deste processo de decisão.

1.1 Objetivos

O problema de alocação de anúncios para usuários é interessante, pois, apesar de existir grande difusão e muito investimento na publicidade digital, a maioria dos trabalhos utilizam heurísticas ou aproximações por programação linear, sem nenhuma garantia teórica de que estas aproximações são efetivas para a solução do problema, ignorando o fato de que o problema original consiste em um problema de decisão sequencial.

O objetivo deste trabalho é propor uma forma de modelagem para redes de distribuição de anúncios por meio de um Processo de Decisão Markoviano (MDP). E a partir do conteúdo adquirido com o estudo de como essa abordagem se comporta em relação às soluções usuais, propor extensões ao modelo utilizando aprendizado por reforço de forma que o modelo se torne o mais adequado possível para a aplicação em um sistema real.

1.2 Contribuição

A principal contribuição deste trabalho consiste na modelagem do problema de alocação de anúncios em *websites*, a avaliação teórica e experimental de como uma solução ótima

baseada nesta modelagem se comporta perante aproximações lineares presentes na bibliografia, e dois algoritmos baseados em aprendizado por reforço para a solução deste problema, considerando-o com e sem informação de contexto, analisando-os experimentalmente.

1.3 Organização do Texto

O restante deste documento está organizado da seguinte forma:

- **Capítulo 2 – Descrição do Problema** apresenta uma descrição do problema de distribuição de anúncios, descrevendo os principais conceitos utilizados na modelagem.
- **Capítulo 3 – Redes de Anúncios** fornece uma revisão dos trabalhos relacionados com o problema de alocação de anúncios;
- **Capítulo 4 – Redes de Anúncios como um MDP** descreve a modelagem do problema descrito no Capítulo 2 como um Processo de Decisão Markoviano;
- **Capítulo 5 – Diferenças entre a Solução Ótima e o Estado da Arte** faz uma análise teórica e experimental sobre a diferença de desempenho entre o estado da arte para soluções de problemas de alocação de anúncios e a solução utilizando o modelo proposto através de um Processo de Decisão Markoviano;
- **Capítulo 6 – Redes de Anúncios como um problema de *Multi-Armed Bandit*** descreve uma abordagem através de Aprendizado por Reforço, utilizando a teoria de *Multi-Armed Bandits*, com e sem a utilização de informação de contexto, comparando os algoritmos tradicionais a dois algoritmos propostos: o AdBandit e o *Extended Thompson Sampling*;
- **Capítulo 7 – Conclusão e Trabalhos Futuros.** resume as principais contribuições do trabalho e indica trabalhos futuros.

2 DESCRIÇÃO DO PROBLEMA

Redes de Anúncios são redes que promovem a distribuição de anúncios em *websites*. Anunciantes produzem anúncios que são alocados em campanhas, *publishers* são *websites* que contêm espaços para a exibição de anúncios. Campanhas são projetadas pelos anunciantes, e são caracterizadas principalmente por: seu conjunto de anúncios, sua data de início de operação, sua data de término de operação e um modelo de precificação. Uma impressão consiste na manifestação de um anúncio em um *website*, ou seja, sua exibição para algum usuário (MUTHUKRISHNAN, 2009).

Existem diversos modelos de precificação, como, por exemplo: (i) o custo por impressões (CPI), em que o anunciante paga um valor fixo para cada visualização; (ii) custo por ação (CPA), em que o anunciante paga para um tipo de ação específica decorrente da exibição dos anúncios, como, por exemplo, uma compra em uma loja *online*; (iii) custo por clique (CPC) em que o anunciante paga uma quantia monetária fixa para cada vez que um usuário clicar em um anúncio. Existem também modelos híbridos; no entanto, a descrição de tais modelos foge ao escopo deste trabalho que trata apenas do modelo CPC, já que este é o modelo mais usual dentre os trabalhos revisados.

É interessante notar que as empresas e usuários costumam misturar métricas e modelos, criando novos significados para as mesmas abreviaturas e palavras. Por exemplo, a métrica chamada CPC (custo por clique) existe em praticamente todos os modelos de precificação, e corresponde ao custo efetivo de um clique não importando qual modelo de precificação tenha sido utilizado. Outra métrica muito importante a ser definida é o custo esperado por impressão (*effective cost per impression* ou eCPI), que é calculada através de toda a receita gerada dividida pelo número de impressões.

2.1 Modelo básico

Inicialmente, considera-se que os anunciantes fazem contratos com a Rede de Anúncios para mostrar seus anúncios em *websites*, fornecendo para a rede um conjunto de campanhas. Considera-se que este contrato seja feito antes do início do processo de distribuição dos anúncios. A Figura 2.1 mostra o fluxo da distribuição de anúncios.

Toda vez que um usuário faz uma requisição de página a um *website* que está dentro da Rede de Anúncios (passo 1 da Figura 2.1), o *website* faz uma requisição de anúncio para exibir para este usuário (passo 2).

Usuários são caracterizados por seu perfil de usuário (conhecido pela Rede de Anúncios). A Rede de Anúncios decide qual campanha será alocada para a requisição recebida, e um anúncio desta campanha é enviado para o *website* (passo 3). A impressão é exibida ao usuário (passo 4), que pode ou não clicar no anúncio (passo 5).

Este processo sequencial pode ser formalizado da seguinte forma: a cada instante de tempo t existe uma probabilidade P_{req} de que uma requisição seja feita para um *website* pertencente ao inventário da Rede de Anúncios. Considera-se que as requisições de páginas seguem uma distribuição de Bernoulli com uma probabilidade de sucesso P_{req} . O conjunto de usuários possíveis é $\mathcal{G}$, e $P_G : \mathcal{G} \rightarrow [0, 1]$ com $\sum_{i \in \mathcal{G}} P_G(i) = 1$, P_G é a função de probabilidade de um usuário pertencer a um determinado perfil de usuário i .

Figura 2.1 – Fluxograma da distribuição de anúncios.

Seja o conjunto $\mathcal{C}$ o conjunto de campanhas. Uma campanha $k \in \mathcal{C}$ é descrita por uma tupla $\langle B_k, S_k, L_k, cc_k \rangle$, onde:

- B_k corresponde ao orçamento em número de cliques;
- S_k corresponde ao início de operação da campanha k ;
- L_k corresponde ao tempo de vida da campanha;
- cc_k corresponde ao preço por clique pago pela campanha.

As campanhas podem estar ativas ou inativas, sendo que apenas campanhas ativas podem ser escolhidas pela Rede de Anúncios. Uma campanha está ativa se, para $\forall t$, seja válido que $S_k < t < S_k + L_k$ e se $B_k > 0$.

Uma vez escolhida a campanha k , um de seus anúncios é exibido ao usuário em um *website* (i.e. é feita uma impressão). O usuário pode ou não clicar neste anúncio com probabilidade $CTR(i, k)$, sendo que CTR, do inglês, *click-through rate*, corresponde à probabilidade de clique associada ao par de campanha e perfil de usuário, e $CTR : \mathcal{G} \times \mathcal{C} \rightarrow [0, 1]$. A ordem de grandeza do CTR em problemas reais é da ordem de 10^{-4} (GIRGIN et al., 2012).

Um clique em um anúncio de uma campanha k gera uma receita igual a cc_k , sendo que uma parte deste valor vai para o *website* e outra para a Rede de Anúncios. O objetivo da Rede de Anúncios é escolher qual campanha deve ser alocada para cada requisição, maximizando uma função de utilidade. Neste trabalho, assume-se que a Rede de Anúncios está interessada em maximizar sua receita esperada.

2.2 Extensão ao modelo básico

Como forma de extensão ao modelo descrito anteriormente, pode-se considerar a possibilidade de que os usuários não sejam discriminados previamente em perfis conhecidos $\mathcal{G}$, mas sim serem descritos por um vetor de atributos de usuário $\vec{x}$.

Desta forma cada requisição de usuário (passo 1 da Figura 2.1) é caracterizada por um vetor de atributos $\vec{x}$. Estes atributos identificam características sobre os gostos deste usuário, dizendo, por exemplo, se este usuário já clicou em algum anúncio anterior, sua idade, sexo, país, fuso horário, histórico de navegação, compras conhecidas, entre outras informações possivelmente relevantes.

Este vetor de usuário seria entregue junto com a requisição de anúncio (passo 2 da Figura 2.1), permitindo à Rede de Anúncios fazer uma escolha mais informada de um anúncio de forma a maximizar as chances de que o anúncio escolhido seja mais relevante ao usuário, e conseqüentemente aumentando a taxa de cliques (CTR).

Considera-se que o conhecimento destes atributos e sua caracterização sejam feitas pelos *websites*, e que estes atributos informem evidências sobre a caracterização de qual perfil de $\mathcal{G}$ um determinado usuário pertence, assume-se também que cada atributo é independente e não mutuamente exclusivo.

Desta forma, considera-se que a probabilidade de clique de um usuário u , que é descrito por um vetor de atributos $\vec{x}$, é igual à multiplicação das probabilidades não nulas de clique associadas a cada um dos atributos $x_i \in \vec{x}$. Denota-se essa probabilidade de clique associada a um atributo por $CTR(x_i, k)$.

Define-se, portanto, que a probabilidade de um anúncio k ser clicado por um usuário u é:

$$CTR(u(\vec{x}), k) = \prod_{x_i \in \vec{x}}^{CTR(x_i, k) \neq 0} CTR(x_i, k). \quad (2.1)$$

Neste trabalho, utilizaremos as duas modelagens descritas anteriormente. O modelo básico é utilizado como base para apresentar as soluções atuais descritas no Capítulo 3, e para a modelagem do problema como um Processo de Decisão de Markov feita no Capítulo 4. O modelo estendido apresentado na seção 2.2 é utilizado para a modelagem do problema como um problema de Aprendizado por Reforço utilizando a teoria de *Multi-Armed Bandits* no Capítulo 6.

3 REDES DE ANÚNCIOS

Este capítulo apresenta um panorama das soluções do problema de alocação de anúncios, apresentando as principais bibliografias utilizadas hoje, na indústria, para a solução do problema de alocação de anúncios em *websites*.

A maioria das soluções atuais modelam o problema de alocação de anúncios como um problema de programação linear. Desta forma é conveniente definirmos o que é programação linear e pelo menos um método de solução.

3.1 Programação Linear

Programação Linear (*Linear Programming*) é um método matemático para minimização ou maximização de uma função linear sujeita a restrições lineares. Isto pode ser definido na forma canônica como:

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & Ax \leq b, \quad x \geq 0, \end{array} \quad (3.1)$$

onde c é um vetor de coeficientes da função que se deseja maximizar, x é um vetor de variáveis, A é uma matriz de coeficientes, e b é um vetor.

Existem diversos métodos para resolver um problema de programação linear e, embora existam algoritmos polinomiais (TARDOS, 1986), o algoritmo simplex e suas variantes correspondem aos algoritmos mais utilizados (DANTZIG, 2003) e são em média muito eficientes, apesar de ter tempo exponencial no pior caso (SCHRIJVER, 1998).

O algoritmo simplex pode ser visto no Algoritmo 1 (SCHRIJVER, 1998). Apesar de a descrição do algoritmo utilizar um número máximo de iterações, Schrijver (SCHRIJVER, 1998) prova que o algoritmo para em um número finito de iterações com uma solução ótima ou uma indicação de que não há solução.

O algoritmo inicia utilizando uma solução inicial que está dentro da região factível, i.e. um vértice do poliedro definido pelas restrições do problema. Depois escolhe-se um subsistema da matriz de restrições (passo 4); existem diversas estratégias para este passo de forma a se evitar ciclos, possibilitando uma convergência mais rápida. Depois o algoritmo prossegue testando os vértices do poliedro formado pelas restrições do problema até que o máximo seja obtido. Neste trabalho utilizamos a ferramenta *IBM ILOG CPLEX Optimizer* para resolver os problemas de programação linear através do algoritmo simplex.

Algoritmo 1: Algoritmo Simplex

Entrada: c : um vetor de coeficientes de tamanho n , b : um vetor de restrições de tamanho m , A : uma matriz $m \times n$ de coeficientes de restrições ordenados por $a_1x \leq b_1, \dots, a_mx \leq b_m$; $maxIter$: número máximo de iterações

Saída: x : solução em forma de um vetor de variáveis de tamanho n

- 1 $iter \leftarrow 0$;
- 2 $x_{iter} \leftarrow$ solução inicial que seja um vértice da região $P = \{x | Ax \leq b\}$;
- 3 **enquanto** $iter < maxIter$ **faça**
- 4 Escolha um subsistema $A_{iter}x \leq b_{iter}$ de $Ax \leq b$ tal que $A_{iter}x_{iter} = b_{iter}$ e que A_{iter} é não singular;
- 5 Calcule u tal que $c = uA$ e que u é 0 nos componentes fora de A_{iter} ;
- 6 **se** $u \geq 0$ **então**
- 7 **retorna** x_{iter} (Solução ótima)
- 8 **senão**
- 9 $i^* \leftarrow$ o menor índice para o qual u tenha um componente negativo u_{i^*} ;
- 10 $y \leftarrow$ um vetor tal que $ay = 0$ para cada linha a de A_{iter} se $a \neq a_{i^*}$ e $a_{i^*}y = -1$;
- 11 **se** $ay \leq 0$ para cada linha a de A **então**
- 12 **retorna** Solução não é limitada
- 13 **se** $ay > 0$ para alguma linha a de A **então**
- 14 $\lambda_{iter} \leftarrow \min \left\{ \frac{b_j - a_j x_{iter}}{a_j y} \mid 1 \leq j \leq m; a_j y > 0 \right\}$;
- 15 $j^* \leftarrow \arg \min_j \left\{ \frac{b_j - a_j x_{iter}}{a_j y} \mid 1 \leq j \leq m; a_j y > 0 \right\}$;
- 16 $A_{iter} \leftarrow$ matriz resultante da troca da linha i^* pela j^* ;
- 17 $x_{iter+1} \leftarrow x_{iter} + \lambda_{iter}y$;
- 18 $iter \leftarrow iter + 1$;
- 19 **retorna** x_{iter}

3.2 Panorama atual das soluções para Redes de Anúncios

Diversas propostas de otimização já foram feitas para a alocação de anúncios. Langheinrich e outros (LANGHEINRICH et al., 1999) são uma das referências mais antigas sobre alocação de anúncios. Eles propuseram uma abordagem utilizando programação linear para tratar da alocação de anúncios em sistemas de busca que, apesar de não ser o foco deste trabalho, é relevante pois apresentou um trabalho focado na maximização dos cliques, sempre se preocupando com a implementação em um ambiente de produção real. Em sua modelagem apresentam o problema de alocação de anúncios como um problema de transporte (DANTZIG, 1998), que consiste em determinar a forma mais eficiente de se enviar um recurso disponível, porém limitado, a determinados locais onde este recurso é necessário. Este trabalho foi a primeira definição formal, que se tem conhecimento, do papel de uma rede de anúncios. No entanto, a escolha do anúncio para o usuário era feita

baseando-se em palavras chaves e aprendiam as preferências do usuário apenas em um horizonte muito curto de tempo, mantida pela sessão do usuário no site de busca.

Chickering e Heckerman (CHICKERING; HECKERMAN, 2000) utilizaram uma modelagem um pouco diferente do problema descrito no capítulo 2 em que, ao invés de perfis de usuários, eram modelados grupos de *sites*, ou seja, ao invés de saber qual o perfil de usuário, sabia-se que o usuário estaria navegando em um *site* de um determinado grupo. Assumindo $P_{groupSite}$ como a função de probabilidade de se pertencer a um determinado grupo, eles sugeriram estimar um valor médio de alocações feitas para cada grupo de *sites*, como:

$$\bar{h}_i = |\mathcal{D}| P_{req} P_{groupSite}(i),$$

sendo $|\mathcal{D}|$ o número de épocas de decisão do problema, isto é, um número de requisições que este sistema teria de responder; e P_{req} a probabilidade de existir uma requisição em um instante de tempo. Então, resolviam o seguinte problema linear:

$$\max_{\bar{x}} \quad \sum_{i \in \mathcal{G}} \sum_{k \in \mathcal{C}} cc_k CTR(i, k) \bar{x}_{ik} \quad (3.2)$$

$$s.t. \quad \sum_{i \in \mathcal{G}} cc_k \bar{x}_{ik} \leq B_k, \quad \forall k \in \mathcal{C} \quad (3.3)$$

$$\sum_{k \in \mathcal{C}} \bar{x}_{ik} \leq \bar{h}_i, \quad \forall i \in \mathcal{G} \quad (3.4)$$

$$\bar{x}_{ik} \geq 0, \quad \forall i \in \mathcal{G}, \forall k \in \mathcal{C}, \quad (3.5)$$

onde cc_k corresponde ao valor pago pela campanha k quando seu anúncio é clicado, $CTR(i, k)$ corresponde à probabilidade de clique associado entre a campanha k e o grupo de *websites* i , B_k corresponde ao orçamento da campanha k .

A partir desta solução $\bar{x}$, a cada instante de tempo em que ocorreria uma requisição de um usuário vindo de um *website* pertencente a um grupo i , alocava-se probabilisticamente a campanha k com a probabilidade:

$$\frac{\bar{x}_{ik}}{\sum_{i \in \mathcal{G}} \bar{x}_{ik}}.$$

O maior problema desta modelagem é que, ao invés de se considerar toda a distribuição de probabilidade de como as requisições podem ocorrer, utiliza-se apenas uma média $\bar{h}_i$.

Chen e outros (CHEN et al., 2011) trabalharam com o problema de *Real-time bidding* (RTB), utilizando uma formulação similar, em que se considerava apenas uma única

alocação para se realizar em cada instante de tempo, modelando o problema da seguinte forma:

$$\max_x \quad \sum_{i \in \mathcal{G}} \sum_{k \in \mathcal{C}} cc_k CTR(i, k) x_{ik} \quad (3.6)$$

$$s.t. \quad \sum_{i \in \mathcal{G}} cc_k x_{ik} \leq B_k, \quad \forall k \in \mathcal{C} \quad (3.7)$$

$$\sum_{k \in \mathcal{C}} x_{ik} \leq 1, \quad \forall i \in \mathcal{G} \quad (3.8)$$

$$x_{ik} \geq 0, \quad \forall i \in \mathcal{G}, \forall k \in \mathcal{C}, \quad (3.9)$$

utilizando o problema dual desta formulação por programação linear, vem:

$$\min_{\alpha, \beta} \quad \sum_k B_k \alpha_k + \sum_i \beta_i \quad (3.10)$$

$$s.t. \quad \alpha_k + \beta_i \geq cc_k CTR(i, k), \quad \forall i \in \mathcal{G}, \forall k \in \mathcal{C} \quad (3.11)$$

$$\alpha_k \geq 0, \quad \forall k \in \mathcal{C} \quad (3.12)$$

$$\beta_i \geq 0, \quad \forall i \in \mathcal{G}, \quad (3.13)$$

onde B_k corresponde ao orçamento da campanha k , cc_k corresponde ao custo por clique pago pela campanha k , α_k corresponde a uma estimativa do lance utilizado para o leilão de Vickrey no RTB, e β_i pode ser entendido como uma medida de custo por impressão, no entanto esta variável não é relevante para o problema de RTB sendo utilizada apenas para calcular o menor lance possível para o leilão. É importante notar a diferença sutil entre $\bar{x}$ (Equação 3.2) e x (Equação 3.6), sendo o primeiro uma média do número de alocações de anúncios para um determinado grupo de usuários, e a segunda uma variável que indica se um anúncio k deve ter uma alocação de anúncio para um único usuário vindo de um grupo de *sites* i .

A modelagem mais recente e mais promissora que se tem conhecimento foi proposta por Girgin e outros (GIRGIN et al., 2012). Ela é baseada em outros dois artigos dos mesmos autores (GIRGIN et al., 2010a, 2010b) e pode ser encarada como uma melhoria direta da formulação proposta por Chen (Equação 3.6), considerando diversas alocações ao invés de uma única alocação. A grande diferença foi perceber que não era necessário considerar uma alocação para cada instante de tempo, mas sim considerar apenas as restrições de tempo das campanhas, criando intervalos de tempo. Consideramos que esta modelagem representa o estado da arte em alocação de anúncios para o problema de alocação de anúncios, desconsiderando a extensão descrita anteriormente na seção 2.2,

isto é, considerando que um usuário é descrito apenas pelo seu perfil de origem, e não como um vetor de atributos de usuário. Esta proposta é formalizada a seguir.

Seja $\mathcal{I}$ a lista ordenada criada sobre o conjunto definido por $\{S_k\}_{k \in \mathcal{C}} \cup \{S_k + L_k\}_{k \in \mathcal{C}}$, i.e. a lista ordenada dos inícios e termos de todas as campanhas. Seja $\mathcal{J}_j = [\mathcal{I}_{j-1}, \mathcal{I}_j[$, com $1 \leq j < |\mathcal{I}|$, os intervalos definidos pelas restrições de tempo de todas as campanhas. Considere os números $\mathbb{T}_j = \sup\{\mathcal{J}_j\} - \inf\{\mathcal{J}_j\}$, estes valores correspondem ao tamanho de um intervalo de tempo j .

Por exemplo, na Figura 3.1 estão definidas três campanhas: C1, C2 e C3. Seus tempos de início e fim definem 5 intervalos. Considerando que $E_k = S_k + L_k$, vem que $\mathcal{I} = \{S_2, S_3, S_1, E_3, E_2, E_1\}$, e tem-se os seguintes intervalos $\mathcal{J}_1 = [S_2, S_3[$, $\mathcal{J}_2 = [S_3, S_1[$, $\mathcal{J}_3 = [S_1, E_3[$, $\mathcal{J}_4 = [E_3, E_2[$, $\mathcal{J}_5 = [E_2, E_1[$, sendo que cada intervalo tem seu tamanho definido por: $\mathbb{T}_1 = S_3 - S_2$, $\mathbb{T}_2 = S_1 - S_3$, $\mathbb{T}_3 = E_3 - S_1$, $\mathbb{T}_4 = E_2 - E_3$, $\mathbb{T}_5 = E_1 - E_2$. Observe ainda que $E_1 = S_1 + L_1$, $E_2 = S_2 + L_2$ e $E_3 = S_3 + L_3$.

Figura 3.1 – Exemplo de Intervalos para 3 campanhas, C1, C2 e C3.

O problema de otimização pode então ser definido como:

$$\max_{x_{j,i,k}} \sum_{j \in \mathcal{J}} \sum_{i \in \mathcal{G}} \sum_{k \in \mathcal{C}} cc_k CTR(i, k) x_{j,i,k}, \quad (3.14)$$

$$s.t. \sum_{k \in \mathcal{C}_j} x_{j,i,k} \leq P_{req} P_G(i) \mathbb{T}_j, \quad \forall i \in \mathcal{G}, \forall j \in \mathcal{J}, \quad (3.15)$$

$$\sum_{i \in \mathcal{G}} \sum_{j \in \mathcal{J}} CTR(i, k) x_{j,i,k} \leq B_k, \quad \forall k \in \mathcal{C}, \quad (3.16)$$

$$x_{j,i,k} \geq 0, \forall j \in \mathcal{J}, \forall k \in \mathcal{C}, \quad \forall i \in \mathcal{G}. \quad (3.17)$$

As variáveis $x_{j,i,k}$ estimam quantos anúncios da campanha k devem ser exibidos aos usuários do perfil i no intervalo j . A função objetivo tenta maximizar a receita total da Rede de Anúncios. O primeiro conjunto de restrições (Inequação 3.15) garante que a solução não utilizará mais requisições que o número de requisições esperado para cada perfil de usuário i no intervalo j . O segundo conjunto de restrições (Inequação 3.16) garante que o número esperado de cliques obtido em cada campanha não exceda o orçamento da sua respectiva campanha. O último conjunto de restrições (Inequação 3.17) garante

que apenas respostas factíveis sejam possíveis. Claramente $x_{j,i,k}$ deveria ser uma variável inteira, já que não é possível a alocação de parcelas do anúncio; no entanto, ressalta-se que este fator foi ignorado pelos autores originais dessa modelagem.

Essa aproximação ao problema original descrito no capítulo 2 e modelada pelas Inequações 3.14-3.17 estima o número médio de anúncios de cada campanha que deve ser exibido para um determinado grupo de usuários em cada intervalo de tempo, mas não fornece nenhuma forma de como se utilizar esta solução.

Girgin e outros (GIRGIN et al., 2012) propuseram duas formas para utilizar esta solução. A primeira, chamada de *The Highest LP Policy* (HLP), que escolhe a campanha utilizando a seguinte função $\pi_{LP}(i, j) = \arg \max_k x_{j,i,k} / \sum_k x_{j,i,k}$, subtraindo 1 de $x_{j,i,k}$ a cada clique. E a segunda, *The Stochastic LP Policy* (SLP), escolhe estocasticamente utilizando como probabilidade a seguinte função: $x_{j,i,k} / \sum_k x_{j,i,k}$. No entanto apenas o método *HLP* foi utilizado nos experimentos contidos nos artigos originais. E ainda existe a necessidade de uma melhor avaliação de políticas alternativas para o uso da solução do problema de alocação de anúncios quando modelado como um problema de programação linear.

3.3 Discussão

As abordagens revisadas neste capítulo sempre se baseiam em um número médio de alocações, seja por um grupo de sites (proposta por Chickering e Herckerman), considerando apenas uma única alocação de anúncio (proposta por Chen e outros), ou por um número médio de alocações por campanha por intervalo de tempo (proposta por Girgin e outros).

No entanto, estas abordagens baseiam-se em uma abordagem determinística, enquanto o problema de alocação de anúncios é um problema estocástico.

Sendo assim, elas ignoram em parte a dinâmica do problema de alocação de anúncios, em que os orçamentos das campanhas variam de acordo com os cliques do usuário em tempo real. Por exemplo, considerando a proposta de Girgin, em um problema com apenas duas campanhas concorrendo no mesmo intervalo de tempo, pode-se imaginar um cenário em que a solução por programação linear apresente apenas alocações para uma destas campanhas; mesmo que na média o orçamento desta campanha termine no final do intervalo de tempo proposto, pode ocorrer que o orçamento desta campanha se esgote antes, já que as probabilidades utilizadas são baseadas em alguma amostra de dados anterior, não correspondendo com a realidade atual do problema, fazendo com que, por um certo tempo, nenhum anúncio seja exibido para o usuário, mesmo que exista orçamento disponível na outra campanha. Este efeito será explorado no Capítulo 5.

No Capítulo 4, a seguir, são expostas as principais bases teóricas para a modelagem do problema como um Processo de Decisão de Markov que será formalizado na Seção 4.2.

4 REDES DE ANÚNCIOS COMO UM PROCESSO DE DECISÃO MARKOVIANO

Neste Capítulo é apresentado um modelo baseado em Processos de Decisão Markovianos para o problema descrito no Capítulo 2. Para isto, primeiramente, são apresentados os conceitos fundamentais para o entendimento deste modelo. A Seção 4.1 introduz o modelo utilizado neste trabalho para a modelagem do problema de alocação de anúncios, o Processo de Decisão Markoviano, e o algoritmo utilizado neste trabalho para sua solução, o algoritmo de Iteração de Valor e na Seção 4.2 é apresentada a modelagem do problema descrito no Capítulo 2 como um Processo de Decisão Markoviano.

4.1 Processos de Decisão Markovianos

Existem diversas formulações para os Processos de Decisão Markovianos. Neste trabalho é utilizada a formulação descrita em (MAUSAM; KOLOBOV, 2012).

Um Processo de Decisão Markoviano finito totalmente observável com tempo discreto em horizonte finito é uma tupla $\langle \mathcal{S}, \mathcal{A}, \mathcal{D}, \mathcal{T}, \mathcal{R} \rangle$, onde:

- $\mathcal{S}$ é o espaço de estados, um conjunto finito dos possíveis estados do sistema, sendo estes totalmente observáveis;
- $\mathcal{A}$ é o conjunto finito das ações que um agente pode tomar; denota-se por $\mathcal{A}(s, t)$ o conjunto de ações válidas no instante de tempo t quando o agente está no estado s ;
- $\mathcal{D}$ é uma sequência finita de números naturais, que denota as épocas de decisão em que ações devem ser tomadas;
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{D} \rightarrow [0, 1]$ é uma função de transição que especifica a probabilidade $\mathcal{T}(s, a, s', t)$ de se transitar de um estado s para um estado s' , quando executada a ação a , no instante de tempo t ;
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{R}$ é uma função de recompensa que retorna uma recompensa numérica finita $r = \mathcal{R}(s, a, s', t)$ quando o sistema vai do estado s para o estado s' como resultado da execução da ação a na época de decisão t .

Um agente modelado por um MDP está constantemente em um ciclo de percepção-ação (Figura 4.1): a cada instante de tempo t o agente observa seu estado atual $s \in \mathcal{S}$ e recebe uma recompensa r , então o agente escolhe uma ação $a \in \mathcal{A}(s, t)$, que o transfere

para um estado s' de acordo com a função de probabilidade $\mathcal{T}$, repetindo este ciclo até alguma condição de parada, como por exemplo $t = |\mathcal{D}| + 1$, isto é, não há mais épocas de decisão válidas.

É importante notar que este sistema não é determinístico. Dado que, com base na função de probabilidade $\mathcal{T}$, uma mesma ação quando tomada sob os mesmos parâmetros (estado de origem e instante de tempo) pode fazer com que o agente transite para diferentes estados de destino. Outro fato a se notar é que a função de probabilidade de transição é dependente do tempo.

Figura 4.1 – O ciclo de percepção-ação – O agente observa o seu estado atual, executa uma ação, recebe uma recompensa e observa um novo estado.

Resolver um MDP consiste em encontrar uma política de ações para controlá-lo de forma que a execução destas ações maximize a recompensa acumulada ao longo das épocas de decisão. A solução para o MDP é, em geral, um mapeamento $\pi : \mathcal{S} \times \mathcal{D} \rightarrow \mathcal{A}$, ou seja, uma política determinística não estacionária que define qual ação deve ser executada em qualquer estado $s \in \mathcal{S}$ e em qualquer época de decisão $t \in \mathcal{D}$. Define-se o valor esperado total de uma política π em qualquer instante de tempo t para todos os estados $s \in \mathcal{S}$ como:

$$V^\pi(s, t) = \mathbf{E} \left[\sum_{i=t}^{|\mathcal{D}|} \mathcal{R}(s_i, \pi(s_i, i), s_{i+1}, i) \middle| s_t = s \right], \quad (4.1)$$

e esse valor esperado é chamado de função valor da política π .

Uma solução ótima para um MDP é uma política π^* sujeita à função valor V^* , que é chamada de função valor ótima. Esta função domina todas as outras funções valores para quaisquer outras políticas, ou seja, para qualquer outra política π , V^* deve satisfazer $V^*(s, t) \geq V^\pi(s, t) \forall s \in \mathcal{S}, \forall t \in \mathcal{D}$. Em outras palavras, uma solução ótima para um MDP consiste em uma política π^* que é pelo menos tão boa quanto qualquer outra política.

O Princípio da Otimalidade (PUTERMAN, 1994) afirma que a função valor ótima V^* existe, é markoviana (i.e. a política não depende do caminho até o estado atual), e para $\forall s \in \mathcal{S}$ com $1 \leq t \leq \tau$, com $\tau = |\mathcal{D}|$ sendo o horizonte de decisão:

$$V^*(s, t) = \max_{a \in \mathcal{A}_t} \left[\sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s', t) [\mathcal{R}(s, a, s', t) + V^*(s', t + 1)] \right], \quad (4.2)$$

onde $\mathcal{A}_t$ denota o subconjunto das possíveis ações na época de decisão t .

A política ótima π^* correspondente à função valor ótima é obtida para todo $s \in \mathcal{S}$ e $t \in \mathcal{D}$,

$$\pi^*(s, t) = \arg \max_{a \in \mathcal{A}_t} \left[\sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s', t) [\mathcal{R}(s, a, s', t) + V^*(s', t + 1)] \right]. \quad (4.3)$$

A solução recursiva dada pela equação 4.2 define uma forma de solução para o MDP. Ela foi originalmente proposta por Richard Bellman em 1957, e pode ser formulada como um algoritmo conhecido como Iteração de Valor (*Value Iteration*), descrito no Algoritmo 2.

No passo 1, define-se o *scrap value*, que é o valor de se estar em um estado quando não há mais nenhuma época de decisão, i.e. $t > |\mathcal{D}| = \tau + 1$; considera-se que não há uma preferência por um estado específico de parada, portanto, utiliza-se um *scrap value* igual a 0 para todos os estados. Então, calcula-se iterativamente os valores para os estados na época de decisão t utilizando os valores dos estados da época de decisão $t + 1$ (passo 5), extraindo simultaneamente a política associada a este valor (passo 6).

Algoritmo 2: Iteração de Valor

Entrada: $\mathcal{S}$: espaço de estados, $\mathcal{A}$: conjunto de ações, $\mathcal{D}$: épocas de decisão, $\mathcal{T}$: função de transição, $\mathcal{R}$: função de recompensa

Saída: π^* : política ótima

- 1 $V^*(s, \tau + 1) \leftarrow 0$ para todo estado $s \in \mathcal{S}$;
 - 2 $t \leftarrow \tau$;
 - 3 **repita**
 - 4 **para cada** $s \in \mathcal{S}$ **faça**
 - 5 $V^*(s, t) \leftarrow \max_{a \in \mathcal{A}_t} [\sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s', t) [\mathcal{R}(s, a, s', t) + V^*(s', t + 1)]]$;
 - 6 $\pi^*(s, t) \leftarrow \arg \max_{a \in \mathcal{A}_t} [\sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s', t) [\mathcal{R}(s, a, s', t) + V^*(s', t + 1)]]$;
 - 7 $t \leftarrow t - 1$;
 - 8 **até** $t = 0$;
 - 9 **retorna** π^* ;
-

Uma vez de posse da política ótima π^* , o agente observa o seu estado atual e aplica a ação indicada pela política para um dado instante de tempo. No problema de alocação de anúncios, analogamente, a Rede de Anúncios observa o seu estado atual, isto é: a configuração de todas as campanhas, se há uma requisição de um determinado perfil de

usuário para ser resolvido; e executa a ação descrita na política para seu estado em um dado instante de tempo.

4.2 Redes de Anúncios como um MDP

Girgin e outros (GIRGIN et al., 2010a) indicaram que o problema de alocação de anúncios poderia ser modelado como um MDP; no entanto, até onde se tem conhecimento, a primeira formulação completa do problema de alocação de anúncios como um MDP foi descrita por Truzzi e outros (TRUZZI et al., 2012). Esta seção apresenta a modelagem das Redes de Anúncios através de Processos de Decisão Markovianos descrita por Truzzi e outros (TRUZZI et al., 2012) com pequenas alterações. Como indicado na Seção 4.1, um MDP é formalizado através da descrição da tupla $\langle \mathcal{S}, \mathcal{A}, \mathcal{D}, \mathcal{T}, \mathcal{R} \rangle$, portanto para definir o MDP basta especificar cada um dos elementos dessa tupla.

4.2.1 Estados

Um estado $s \in \mathcal{S}$ é modelado como um vetor $s = [B_1, B_2, \dots, B_k, G]$, onde B_k corresponde ao orçamento restante de cada campanha em número de cliques, como definido no capítulo 2, e $G \in \mathcal{G}$ corresponde ao perfil do usuário que gerou a requisição. Quando a variável G é igual a 0 significa que naquele estado não há nenhuma requisição para ser resolvida. Por exemplo, considere 3 campanhas (C1, C2 e C3), e 3 perfis de usuários ($G = 1$, $G = 2$ e $G = 3$); um possível estado $s \in \mathcal{S}$ para $B_1 = 10$, $B_2 = 3$ e $B_3 = 4$, considerando que o usuário com o perfil de usuário 3 gerou uma requisição, seria:

$$s = \left[\underbrace{\overbrace{10, 3, 4}^{\text{Informação da Campanha}}, \overbrace{3}^{\text{Informação da Requisição}}}_{\text{Vetor de Estado}} \right]. \quad (4.4)$$

A informação da campanha indica quantos anúncios cada campanha pode pagar em um determinado estado. No exemplo acima, a campanha 1 pode pagar por 10 cliques, a campanha 2 pode pagar 3 cliques, e a campanha 3, por 4 cliques. A informação da requisição contém qual foi o perfil de usuário que gerou a requisição; neste exemplo, o perfil de usuário 3 gerou uma requisição.

A partir deste estado, os possíveis próximos estados são: $[9, 3, 4, G]$, $[10, 2, 4, G]$, $[10, 3, 3, G]$, e $[10, 3, 4, G]$, onde $G \in \mathcal{G} \cup \{0\}$, ou seja, dado que a modelagem do problema descrita no capítulo 2 se assume que a geração de requisições segue uma distribuição de Bernoulli, existindo no máximo uma única requisição para ser resolvida e, consequentemente, o orçamento das campanhas pode variar no máximo de 1 clique entre um estado e outro.

4.2.2 Ações

Uma ação neste MDP corresponde à escolha de qual campanha do conjunto $\mathcal{C}$ terá seu anúncio exibido para um usuário de um perfil pertencente ao conjunto de perfis de usuário $\mathcal{G}$ em uma época de decisão pertencente ao conjunto $\mathcal{D}$. O conjunto de ações possíveis é $\mathcal{A} = \{0, 1, \dots, |\mathcal{C}|\}$, com o seguinte significado:

$$a = k, \quad (4.5)$$

se $k > 0$, então k corresponde ao índice da campanha a ser exibida, e se $k = 0$, então a Rede de Anúncios não irá alocar nenhuma campanha nesta época de decisão.

Como as campanhas podem estar ativas ou inativas, em cada época de decisão $t \in \mathcal{D}$ o subconjunto de ações permitidas em cada instante é $\mathcal{A}_t$. Tem-se que $0 \in \mathcal{A}_t$ para todo $t \in \mathcal{D}$; e que $k > 0 \in \mathcal{A}_t$ se $S_k \leq t < S_k + L_k$ para todo $t \in [0, \tau]$, onde $\tau = |\mathcal{D}|$ e $B_k > 0$, ou seja, só há alocação de campanhas ativas.

Utilizando como exemplo a configuração temporal de três campanhas como descrito na Figura 3.1, pode-se definir os seguintes subconjuntos de ações:

- $\mathcal{A}_{S_2 < t \leq S_3} = \{2, 0\}$,
- $\mathcal{A}_{S_3 < t \leq S_1} = \{2, 3, 0\}$,
- $\mathcal{A}_{S_1 < t \leq E_3} = \{1, 2, 3, 0\}$,
- $\mathcal{A}_{E_3 < t \leq E_2} = \{1, 2, 0\}$,
- $\mathcal{A}_{E_2 < t \leq E_1} = \{1, 0\}$,

onde $\mathcal{A}_{t_1 < t \leq t_2} = \cup_{t \in [t_1, t_2]} \mathcal{A}_t$.

4.2.3 Função de Transição

A função de transição $\mathcal{T}$ deve, para todas as ações a e todos os estados s e s' em todos os instantes t , estar sujeita às seguintes restrições $0 \leq \mathcal{T}(s, a, s', t) \leq 1$ e $\sum_{s' \in \mathcal{S}} \mathcal{T}(s, a, s', t) = 1$, i.e. a função deve definir uma distribuição de probabilidade discreta sobre o conjunto de estados. Como simplificação, considera-se que a função de transição não varia no tempo, i.e. $\mathcal{T}(s, a, s') = \mathcal{T}(s, a, s', t)$ para $\forall t$.

O elemento G do vetor de estado não depende do estado anterior. E os elementos B_k referentes aos orçamentos das campanhas dependem apenas do estado anterior e da ocorrência de cliques.

Dado os estados $s = [B_1, B_2, \dots, B_j, G]$ e $s' = [B'_1, B'_2, \dots, B'_j, G']$, a função de transição $\mathcal{T}$ é:

$$\mathcal{T}(s, a, s') = P_t(G') \times \prod_{k \in \mathcal{C}} P(B'_k | B_k, a, G), \quad (4.6)$$

onde $P_t(G')$ é:

$$P_t(G') = \begin{cases} (1 - P_{req}) & \text{se } G' = 0, \\ P_{req} \times P_G(G) & \text{se } G' \in \mathcal{G}, \end{cases} \quad (4.7)$$

e $P(B'_k | B_k, a, G)$ é igual a:

$$P(B'_k | B_k, a, G) = \begin{cases} 1 & \text{se } B'_k = B_k \text{ e } (a \neq k \text{ ou } \\ & G = 0 \text{ ou } B_k = 0), \\ CTR(G, k) & \text{se } B'_k = B_k - 1 \text{ e } \\ & (a = k \text{ e } G > 0 \text{ e } B_k > 0), \\ 1 - CTR(G, k) & \text{se } B'_k = B_k \text{ e } (a = k \text{ e } \\ & G > 0 \text{ e } B_k > 0), \\ 0 & \text{caso contrário.} \end{cases} \quad (4.8)$$

A Equação 4.7 corresponde à probabilidade de existir uma requisição no próximo estado, e é dependente apenas dos parâmetros do problema (P_G e P_{req}). A Equação 4.8 depende do estado anterior, da ação a ser executada, da existência de uma requisição no estado anterior e da probabilidade de clique associada ao grupo e campanha. É interessante notar que a função de probabilidade $\mathcal{T}$ está definida para qualquer combinação de estados e ações e é invariante no tempo.

Como exemplo, considere um problema em que $P_{req} = 0,9$, $\mathcal{G} = \{1, 2\}$, $B_1 = B_2 \geq 2$, $P_G(G) = \frac{1}{G}$, e $CTR(i, k) = i \times k \times 10^{-4}$. A partir de um estado $s = [B_1, B_2, 1]$, há a possibilidade de 12 futuros estados. Na Figura 4.2 estão alguns exemplos; nela,

- $P1 = \frac{1}{1} \times 1 \times (1 - 1 \times 1 \times 10^{-4})$,
- $P2 = \frac{1}{1} \times 1 \times 1 \times 10^{-4} \times 1$,
- $P3 = \frac{1}{1} \times 1 \times 1 \times 10^{-4} \times 0$,
- $P4 = \frac{1}{G} \times (1 - G \times 1 \times 10^{-4}) \times 0$.

Figura 4.2 – Alguns exemplos de próximos estados ao executar a ação 1 no estado $s = [B_1, B_2, 1]$.

4.2.4 Função de Recompensa

A função de recompensa $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{D} \rightarrow \mathbb{R}$ atribui uma recompensa numérica finita quando o sistema vai do estado s para o estado s' como o resultado da ação a na época de decisão t . No entanto, nesta modelagem, assumimos que esta recompensa não varia no tempo, e que a recompensa no instante da aplicação da ação a independe do próximo estado s' utilizando o valor esperado desta ação. Redefinimos, portanto, a função recompensa como $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, sendo:

$$\mathcal{R}(s, k) = \begin{cases} cc_k \times CTR(G, k) & \text{se } k > 0, G > 0 \text{ e } B_k > 0, \\ 0 & \text{caso contrário,} \end{cases} \quad (4.9)$$

onde cc_k e $CTR(G, k)$ foram definidos no capítulo 2, e especificam, respectivamente, o quanto a campanha k paga por clique e a probabilidade de clique entre a campanha k e o perfil de usuário G correspondente ao estado s . A intuição desta função de recompensa é que ela representa uma avaliação local da ação tomada pela Rede de Anúncios. Este valor corresponde ao custo esperado de uma exibição, eCPI, i.e. a probabilidade de clique multiplicada pelo valor pago por clique. A modelagem do problema como um MDP possibilita que seja calculada uma solução ótima para o MDP.

Esta solução ótima consiste de uma política π não estacionária, isto é, um mapeamento que, para cada um dos estados definidos em cada instante de tempo, retorna a melhor ação a ser utilizada, levando em conta todas as possibilidades de combinações entre os orçamentos das campanhas e o perfil de usuário. Neste trabalho utilizamos para calcular a solução ótima, quando necessário, o algoritmo de Iteração de Valor, descrito no Algoritmo 2.

O próximo Capítulo explora a diferença de desempenho entre a solução ótima criada a partir de um MDP e a solução baseada em programação linear proposta por Girgin e outros (GIRGIN et al., 2012).

5 DIFERENÇAS ENTRE A SOLUÇÃO ÓTIMA E O ESTADO DA ARTE

A formulação do problema através de programação linear baseia-se em uma visão determinística, apesar de o problema da Rede de Anúncios ser um problema estocástico.

As requisições seguem uma distribuição multinomial, com parâmetros:

$$(1 - P_{req}), P_{req}P_G(1), P_{req}P_G(2), \dots, P_{req}P_G(|\mathcal{G}|).$$

As ações da Rede de Anúncios não afetam essa distribuição, já que a Rede de Anúncios não tem controle sobre quais usuários interagem com os *websites*, então considerar que a distribuição de usuários para cada perfil de usuário possa ser determinística não afeta a solução ótima.

Os cliques nos anúncios seguem uma distribuição binomial com probabilidade de sucesso dada por $CTR(i, k)$, mas limitada pelo orçamento da campanha k . A formulação por programação linear assume que a ocorrência de cliques é determinística, já que é calculada baseando-se apenas nas médias das alocações, não possuindo uma solução geral que cubra todas as combinações de estados possíveis. Isso afeta a solução do problema, não importando qual heurística esteja sendo utilizada para a aplicação da política determinada (*HLP* ou *SLP*).

Ambas as heurísticas de utilização proposta por Girgin e outros (GIRGIN et al., 2012) não consideram a dinâmica dos cliques. Quando os cliques ocorrem um nível abaixo de seu valor esperado, estas soluções podem alocar campanhas com baixos CTRs, ou até mesmo não alocar nenhuma campanha, em detrimento de campanhas que possam ter um CTR maior.

Girgin e outros (GIRGIN et al., 2012) sugerem um aumento artificial no orçamento das campanhas com um CTR mais elevado, de forma a diminuir este efeito. No entanto não existe uma forma simples de escolher o quanto este orçamento deveria ser elevado para melhorar a solução, e ainda assim esta solução poderia não ser ótima.

Sabendo-se que soluções através da formulação por programação linear são intrinsecamente sub-ótimas, pode-se perguntar quão distantes estas soluções estão da solução ótima encontrada ao resolver o problema modelado por MDP.

Para se avaliar esta diferença entre a solução ótima encontrada ao resolver o problema modelado por um MDP e a solução através da formulação por programação linear considera-se um problema com a seguinte configuração: um único perfil de usuário e duas campanhas.

Figura 5.1 – Possíveis configurações com um usuário e duas campanhas: (i) campanhas disjuntas, (ii) campanhas parcialmente sobrepostas e (iii) uma campanha totalmente sobreposta a outra.

Nesta configuração são possíveis três casos: (i) as campanhas não compartilham intervalos; (ii) as campanhas compartilham parcialmente intervalos; e (iii) uma campanha cujo período de operação está em intersecção com os intervalos de outra campanha. Estas configurações estão ilustradas na Figura 5.1.

5.1 Construção de um caso ruim para a solução por Programação Linear

Considere as configurações de campanhas apresentadas na Figura 5.1. Intuitivamente, quanto maior a sobreposição entre as campanhas, mais difícil é de se obter uma solução, já que quanto mais sobreposição houver, mais campanhas existem para se escolher, permitindo ocorrer o efeito de que apenas uma campanha seja alocada durante o intervalo completo de sobreposição. Portanto, consideramos que o caso (iii) seja o pior caso dentre os apresentados. Utilizaremos uma instância específica do caso (iii), ajustando os parâmetros de forma a produzir um cenário muito ruim para as políticas que utilizam a solução da formulação por programação linear.

Para simplificar a notação, nesta seção vamos utilizar $ctr_k = CTR(1, k)$. Considere então o seguinte cenário:

$$P_{req} = 1, \tag{5.1}$$

$$S_1 < S_2, \tag{5.2}$$

$$L_1 \rightarrow \infty, \quad (5.3)$$

$$L_2 \rightarrow \infty, \quad (5.4)$$

$$ctr_2 \rightarrow 0, \quad (5.5)$$

$$B_2 = 1, \quad (5.6)$$

$$cc_1 = cc_2 = 1, \quad (5.7)$$

$$ctr_1 = \frac{B_1}{P_{req}(S_2 - S_1)}, e \quad (5.8)$$

$$0 < \lim_{L_2 \rightarrow \infty, ctr_2 \rightarrow 0} L_2 \times ctr_2 < \epsilon. \quad (5.9)$$

O limite da Equação 5.9 existe se for feita uma mudança de variáveis apropriada para L_2 e ctr_2 , por exemplo: $L_2 = t^2$, $ctr_2 = 1/t^2$, com $t \rightarrow 0$. De forma simplificada ele serve apenas para dizer que o máximo retorno financeiro que pode ser obtido pela campanha 2 é limitado por um valor ϵ , que por construção é menor ou igual a B_2

Neste cenário, como a campanha 1 possui um $eCPI$ maior, fica claro que a solução ótima é exibir a campanha 1 até que seu orçamento se esgote e, só então, iniciar a exibição da campanha 2.

As políticas criadas com a solução da formulação por programação linear irão alocar, no primeiro intervalo, $P_{req} \times \mathbb{T}_1$ requisições para a campanha 1, enquanto, no segundo intervalo, irão alocar $P_{req} \times \mathbb{T}_2$ requisições para a campanha 2. Este comportamento surge pelo fato de que, em média, o orçamento da campanha 1 acaba no fim do primeiro intervalo.

Pode-se, então, calcular o resultado da execução da política ótima ($V^{\pi_{MDP}}$) e os resultados utilizando as políticas definidas para a solução da formulação por programação linear ($V^{\pi_{LP}}$).

Proposição 1. *O valor $V^{\pi_{MDP}}$ obtido pela solução ótima através da formulação com o MDP para o cenário definido satisfaz*

$$B_1 < V^{\pi_{MDP}} < B_1 + \epsilon.$$

Demonstração. Seja P_{remain}^t a probabilidade de, na época de decisão t , ainda existir orçamento restante na campanha 1. Como $\mathbb{T}_2 \rightarrow \infty$ e $ctr_1 > 0$, existe $t_{\epsilon_1} > 0$ tal que $P_{remain}^{t_{\epsilon_1}} > \epsilon_1/B_1$. Dado que $\epsilon_1 = \lim_{L_2 \rightarrow \infty, ctr_2 \rightarrow 0} L_2 \times ctr_2$ corresponde ao valor de se exibir a campanha 2 infinitamente. Tem-se que

$$V^{\pi_{MDP}} > B_1(1 - P_{remain}^{t_{\epsilon_1}}) + \epsilon_1 > B_1.$$

O limite superior da proposição é trivial já que $\lim_{L_2 \rightarrow \infty, ctr_2 \rightarrow 0} L_2 \times ctr_2 < \epsilon$. Intuitivamente, esta limitação significa que o valor máximo obtido pela política ótima está limitada entre o orçamento da campanha 1 e o orçamento da campanha 1 somado com o valor máximo obtido pela utilização da campanha 2. $\square$

Proposição 2. *O valor $V^{\pi_{LP}}$ obtido pela solução baseada na solução por programação linear para o cenário definido satisfaz*

$$\begin{aligned} V^{\pi_{LP}} &\leq \sum_{b=1}^{B_1} b \times \binom{\mathbb{T}_1}{b} (ctr_1)^b (1 - ctr_1)^{\mathbb{T}_1 - b} \\ &\quad + \sum_{b=B_1+1}^{\mathbb{T}_1} B_1 \times \binom{\mathbb{T}_1}{b} (ctr_1)^b (1 - ctr_1)^{\mathbb{T}_1 - b} + \epsilon. \end{aligned}$$

Demonstração. Dado que a campanha 1 é utilizada apenas no primeiro intervalo, a Rede de Anúncios tem $\mathbb{T}_1$ chances para consumir o orçamento B_1 , já que $P_{req} = 1$. Portanto, o valor obtido no primeiro intervalo é o valor esperado do número de sucessos limitado pelo orçamento B_1 de uma distribuição binomial com $\mathbb{T}_1$ tentativas com taxa de sucesso ctr_1 . O fator ϵ corresponde ao valor esperado de se exibir a campanha 2 infinitamente. $\square$

Proposição 3. *O desempenho relativo entre a solução ótima e a solução com programação linear é dado por:*

$$\frac{V^{\pi_{MDP}}}{V^{\pi_{LP}}} = \lim_{\mathbb{T}_1 \rightarrow \infty} \left(1 - \left(1 - \frac{1}{\mathbb{T}_1} \right)^{\mathbb{T}_1} \right)^{-1} = \frac{e}{e-1} \approx 1.582.$$

Demonstração. Tem-se que $V^{\pi_{MDP}}/V^{\pi_{LP}}$ aumenta com $\mathbb{T}_1$ e o número de elementos no primeiro somatório de $V^{\pi_{LP}}$ cai com o decrescimento do orçamento B_1 , assim como o valor do segundo somatório; no limite tem-se que $B_1 = 1$. Fazendo $\epsilon \rightarrow 0$, tem-se que $V^{\pi_{MDP}} \rightarrow B_1 = 1$ e

$$V^{\pi_{LP}} \rightarrow \sum_{b=1}^{\mathbb{T}_1} \binom{\mathbb{T}_1}{b} (ctr_1)^b (1 - ctr_1)^{\mathbb{T}_1 - b} = 1 - \left(1 - \frac{1}{\mathbb{T}_1} \right)^{\mathbb{T}_1}.$$

$V^{\pi_{LP}}$ chega em seu valor mínimo quando $\mathbb{T}_1 \rightarrow \infty$. Como $V^{\pi_{MDP}} \rightarrow 1$, o desempenho relativo é dado simplesmente por $(V^{\pi_{LP}})^{-1}$. Tomando o limite

$$\lim_{\mathbb{T}_1 \rightarrow \infty} \left(1 - \left(1 - \frac{1}{\mathbb{T}_1} \right)^{\mathbb{T}_1} \right)^{-1}$$

tem-se:

$$\frac{e}{e-1} \approx 1,582.$$

□

Este resultado mostra como a solução ótima do problema modelado por MDP pode ser melhor em relação à solução baseada na formulação por programação linear. No pior caso para a solução por programação linear, o desempenho relativo é de 1,582, i.e. a solução ótima do MDP pode ser até 58,2% melhor que a solução guiada pela solução baseada em programação linear. No entanto, em cenários reais tem-se que $B_k \gg 1$ e, assim o desempenho relativo é muito menor que 1,582. Este fato é demonstrado experimentalmente, na próxima seção.

5.2 Experimentos

Para comparar os resultados obtidos pela solução ótima calculada pelo MDP e a solução baseada na formulação por programação linear, um experimento foi conduzido para a configuração (iii) da Figura 5.1, explorando diferentes valores para os parâmetros B_1 , B_2 , $\mathbb{T}_1$, $\mathbb{T}_2$ e $CTR(1,2)$ e utilizando $CTR(1,1)$ como na seção anterior, isto é:

$$CTR(1,1) = \frac{B_1}{P_{req}(S_2 - S_1)} = \frac{B_1}{P_{req}\mathbb{T}_1}.$$

Os experimentos foram conduzidos para mostrar como a razão $V^{\pi_{MDP}}/V^{\pi_{LP}}$ se comporta com o aumento do orçamento, comparando quatro diferentes cenários: o primeiro cenário utiliza o caso descrito na seção anterior, i.e. $B_2 = 1$, $\mathbb{T}_2 \rightarrow \infty$, $CTR(1,2) \rightarrow 0$. O segundo cenário relaxa o tamanho do segundo intervalo utilizando $\mathbb{T}_2 = 50.000$. Neste caso, a Rede de Anúncios não teria um tempo infinito para consumir a campanha 1. O terceiro cenário relaxa o orçamento e o CTR da campanha 2 utilizando $B_2 = B_1$, $CTR(1,2) = 0,1 \times CTR(1,1)$. Neste caso, a campanha 1 é muito mais atrativa que a campanha 2, mas é possível obter receita no segundo intervalo de tempo. Finalmente, no quarto cenário, relaxa-se o CTR da campanha 2 para que tenha um valor próximo do CTR da campanha 1, utilizando $CTR(1,2) = 0,5 \times CTR(1,1)$.

A Figura 5.2 mostra o desempenho relativo das duas soluções para os quatro cenários descritos. Pode-se ver claramente que a diferença entre a solução ótima e a solução

aproximada baseada na formulação por programação linear é uma função decrescente com o aumento do orçamento das campanhas. A Figura 5.2 mostra o desempenho relativo com um orçamento crescente até $B_k = 50$ e a Figura 5.3 mostra o desempenho relativo para o orçamento crescente de $B_k = 150$ até $B_k = 200$.

Com um orçamento de 20 cliques, o desempenho relativo cai para um valor abaixo de 10%; com um orçamento igual a 70 cliques, todos os cenários têm um desempenho relativo menor que 5%. Na Figura 5.3 pode-se ver que o cenário com $CTR(1,2) = 0$ fica com o mesmo desempenho que o cenário 1, i.e. $T_2 = 50.000$ é suficiente para que ele tenha o mesmo comportamento que o cenário 1; note, também, que o desempenho relativo continua caindo. Para $B_k = 200$, a diferença relativa cai para 2,89% no cenário 1 e para 1,91% no cenário 4.

Figura 5.2 – Desempenho relativo da solução por MDP em relação à solução baseada na formulação por programação linear, para $T_2 = 50.000$ (cenários 2 ao 4) e para o caso descrito no cenário 1, com orçamentos variando de 1 até 50.

Também foi calculado o desempenho relativo para um orçamento de 500 cliques para os quatro cenários propostos. A Tabela 5.1 mostra os resultados, que variam de 1,21% até 1,81%.

Figura 5.3 – Desempenho relativo da solução ótima em relação à solução baseada na formulação por programação linear, para $T_2 = 50,000$ (cenários 2 ao 4) e para o pior caso descrito no cenário 1 com orçamentos variando de 150 até 200.

Em problemas reais, como descrito anteriormente, o orçamento das campanhas são da ordem de 10.000 cliques (GIRGIN et al., 2012). No entanto, dado o tamanho do espaço de estados para este caso ($|\mathcal{S}| = 10^8$), apenas foi calculado o desempenho relativo para o cenário 1, o pior cenário para a solução por programação linear descrita na Seção anterior, utilizando $T_1 = 10^8$ e $CTR(1, 1) = 10^{-4}$. Neste caso, o desempenho relativo foi de 1,0040, ou seja, a solução ótima através do MDP oferece menos de 0,4% de melhora em relação a soluções baseadas na formulação por programação linear.

Tabela 5.1 – Desempenho relativo da solução por MDP em relação à solução baseada na formulação por programação linear, para $T_2 = 50.000$ (cenários 2 ao 4) e para o cenário 1 ($T_2 \rightarrow \infty$) e $B_1 = 500$.

Cenário 1	Cenário 2	Cenário 3	Cenário 4
1,81%	1,81%	1,64%	1,21%

5.3 Diferenças entre a complexidade assintótica das soluções

Ambas as soluções possuem algoritmos para solução em tempo polinomial em relação ao número de variáveis (ou estados); no entanto, a complexidade espacial das soluções são muito diferentes. As restrições da formulação do estado da arte têm complexidade $O(|\mathcal{J}| \times |\mathcal{G}| + |\mathcal{C}| + |\mathcal{J}| \times |\mathcal{G}| \times |\mathcal{C}|)$; mas, por definição, $1 \leq |\mathcal{J}| \leq 2 \times |\mathcal{C}|$, tornando o número de restrições da ordem $O(|\mathcal{G}||\mathcal{C}|^2)$. O número de variáveis também possui esse mesmo limite assintótico, pelas mesmas razões.

O tamanho da política a ser encontrada pela solução ótima é igual a $|\mathcal{S}| \times |\mathcal{D}|$, com $|\mathcal{S}| = (|\mathcal{G}| + 1) \times \prod_{k \in \mathcal{C}} (B_k + 1)$. Se considerarmos um $B_{min} = \min_{k \in \mathcal{C}} \{B_k\}$, tem-se que $|\mathcal{S}| \geq (|\mathcal{G}| + 1) \times (B_{min} + 1)^{|\mathcal{C}|}$. Este comportamento exponencial faz com que essa modelagem possa ser utilizada apenas em problemas pequenos.

5.4 Considerações Finais

Apesar de o MDP possuir teoricamente um resultado melhor que a solução por programação linear, ele possui a maldição da dimensionalidade, possuindo um espaço de estados que cresce exponencialmente de acordo com os parâmetros do problema (número de campanhas, orçamento e número de perfis de usuário), o que sugere que uma abordagem por programação linear é de fato uma solução adequada ao problema.

Por outro lado, ambas as soluções são limitadas. A solução ótima modelando o problema por um MDP através de técnicas de planejamento (como por exemplo, o algoritmo de iteração de valor), necessita o conhecimento prévio de um modelo que descreve todas as características do problema. Já a solução por programação linear necessita que exista uma base de dados prévia, para se descrever as restrições do problema de programação linear de forma a se calcular qual a proporção da alocação de cada anúncio para cada grupo, como descrito na Expressão 3.14.

Ambas as soluções são, portanto, limitadas, já que precisam de boas estimativas de CTR, o que em geral não é possível de se obter, dado o ambiente dinâmico da internet, em que as probabilidades de clique estão sujeitas a variações abruptas.

Neste contexto pode-se afirmar que uma solução através de aprendizado por reforço se torna adequada, já que ela pode aprender as probabilidades de clique enquanto atua, podendo utilizar sempre a estimativa de CTR mais atualizada possível, além de poder atuar em ambientes dinâmicos em que os perfis de usuários mudam com o passar do tempo, e que novas campanhas de anúncios podem entrar e sair do sistema.

Aprender o MDP completo do problema através de aprendizado por reforço pode ser um problema complexo, dada a característica exponencial do crescimento do espaço de estados. Desta forma, é necessário criar abstrações de forma a simplificar o modelo.

No entanto, uma boa abstração, isto é, uma abstração capaz de diminuir significamente o espaço de estados e que tenha pouca influência sobre a qualidade da política encontrada, não é fácil de ser obtida, e pode ser diferente em cada instância do problema.

Decidiu-se, portanto, em abstrair o modelo apresentado no Capítulo 2, por um modelo de *Multi-Armed Bandits* que será descrito no Capítulo 6.

6 REDES DE ANÚNCIOS COMO UM PROBLEMA DE *MULTI-ARMED BANDIT*.

Este Capítulo apresenta duas modelagens para o problema de alocação de anúncios utilizando Aprendizado por Reforço, uma sem considerar informação de contexto, e outra considerando informação de contexto.

Inicialmente serão apresentados os principais conceitos necessários para seu entendimento. Na Seção 6.1 é introduzido o conceito de Aprendizado por Reforço, e uma modelagem específica simplificada do Aprendizado por Reforço, o *Multi-Armed Bandit* que será utilizado aqui para modelar o problema de alocação de anúncios como um problema de *Multi-Armed Bandits*. Na Seção 6.2 é apresentada a técnica para comparação entre algoritmos para *Multi-Armed Bandit*, a qual será utilizada nos experimentos descritos neste Capítulo.

Em sequência, apresenta-se os principais algoritmos para a solução de um MAB sem informação de contexto, e é proposto um algoritmo chamado AdBandit, que é comparado experimentalmente com as soluções usuais.

Finalmente, apresenta-se os principais algoritmos para a solução de um MAB com informação de contexto, propondo aqui um algoritmo chamado *Extended Thompson Sampling*, que será comparado experimentalmente com soluções existentes na literatura, em uma base em uma base de dados reais.

6.1 Aprendizado por Reforço

No aprendizado por reforço o agente utiliza suas próprias experiências de forma a aprender uma política, ou seja, aprende baseando-se nas suas interações com o ambiente, não possuindo um conhecimento explícito da solução do problema a priori. O agente neste contexto pode tomar dois tipos de ações, uma utilizando seu conhecimento baseado nas suas experiências (exploração), outra utilizando novas escolhas (exploração).

O agente utiliza recompensas numéricas (reforço ou recompensa) recebidas durante suas experiências com o ambiente. Este valor é diretamente relacionado ao sucesso de uma ação, sendo que o agente procura escolher as ações que maximizarão a sua recompensa acumulada através do tempo. No entanto, esta tomada de decisão não é algo simples já que não existe um método que resolva completamente um balanceamento entre a exploração e exploração de forma ótima. Em casos mais complexos, as ações tomadas afetam não somente a recompensa imediata como também as recompensas futuras, já que as ações

modificam os possíveis estados futuros. O ciclo de atuação de um agente modelado por aprendizado por reforço pode ser visto na Figura 6.1.

Figura 6.1 – Ciclo de atuação de um agente modelado por aprendizado por reforço.

Existem diversas estratégias para se balancear entre exploração e exploração, uma das mais simples é chamada de ϵ -greedy. Nesta estratégia a exploração ocorre baseando-se em uma pequena probabilidade ϵ das vezes, enquanto explora com probabilidade $1 - \epsilon$. Esta estratégia é descrita no Algoritmo 3. O algoritmo utiliza como entradas o parâmetro de exploração ϵ , o estado atual s , e uma estimativa da função valor V , denotada por Q . Note que esta aproximação pode ser calculada, por exemplo, através de uma função valor para uma política π , definida na Equação 4.1, calculando-se o valor esperado da ação a para todos os instantes de tempo t .

Algoritmo 3: Algoritmo ϵ -greedy(Q, s, ϵ)

Entrada: Q : estimativa da função valor dado um estado e ação, s : estado atual, ϵ : taxa de exploração

- 1 Amostre p através de uma distribuição uniforme de 0 a 1;
 - 2 **se** $p \leq \epsilon$ **então**
 - 3 // Exploração;
 - 4 $a \leftarrow \text{random } a \in \mathcal{A}(s)$;
 - 5 **senão**
 - 6 // Exploração;
 - 7 $a \leftarrow \max_{a'} Q(s, a')$;
 - 8 **return** a
-

Apesar deste método ser elegante, ele possui algumas importantes limitações: a proporção máxima em que uma ação ótima pode ser utilizada está limitada por $1 - \epsilon$, e a escolha do fator de exploração está diretamente ligada à velocidade de convergência para as ações ótimas.

Existem diversos métodos (SUTTON; BARTO, 1998) mais eficientes e eficazes para solução destes tipos de problemas como por exemplo: métodos de Monte Carlo (MC) que utilizam amostragem para estimar o modelo do MDP resolvido pelo aprendizado por reforço, e *Temporal-Difference Learning* (TD) que utiliza uma combinação entre o método de Monte Carlo e programação dinâmica; dentre esta última classe, pode-se citar por exemplo o algoritmo *Q-Learning* (WATKINS; DAYAN, 1992).

6.1.1 *Multi-Armed Bandits* (MAB)

A teoria dos *Multi-Armed Bandits* possui diversas aplicações, já que o MAB modela o problema de alocação dinâmica (GITTINS, 1979). O termo *Multi-Armed Bandit* foi criado com uma analogia ao jogo de caça niqueis (*one-armed bandit*). Considere que um agente enfrente repetidamente uma decisão entre n diferentes opções de ações. Depois de cada escolha o agente recebe uma recompensa numérica escolhida através de uma distribuição estacionária que depende da ação escolhida, sendo o objetivo do agente maximizar a recompensa acumulada esperada sobre algum período de tempo. Este problema é essencialmente um caso do problema mais geral de aprendizado por reforço.

Nas finanças, esta teoria foi utilizada para modelar qual opção de ação produz o maior retorno financeiro (BANK; FÖLLMER, 2003), como aprender a real demanda de um mercado enquanto se maximiza as recompensas descontadas (BERGEMANN; VÄLIMÄKI, 2006), e muitas outras aplicações (BERGEMANN; VÄLIMÄKI, 2006).

Na ecologia, esta teoria foi utilizada para modelar o forragear de abelhas, i.e. a saída da abelha à procura de alimento. Neste caso o modelo foi aplicado na modelagem do comportamento de abelhas que precisam escolher entre diferentes espécies de flores que diferem em recompensas nutricionais (KEASAR et al., 2002).

No contexto dos ensaios clínicos, o MAB foi usado extensivamente como forma de se escolher entre diferentes tratamentos que precisam ser experimentados em um paciente minimizando a sua perda de saúde (WANG, 1991; THOMPSON, 1933; POLONIECKI, 1978).

Este modelo também é utilizado em arcabouços para testes A/B, aplicações que permitem um experimento controlado para se descobrir dentre duas opções A e B qual tem o melhor desempenho de acordo com alguma métrica. Neste contexto espera-se aprender qual conteúdo ou *layout* de página possui a maior taxa de conversão. Um exemplo destes arcabouços é o *Google Website Optimizer*.¹

O MAB já foi utilizado extensivamente no *marketing online*. Neste problema cada anúncio pode ser modelado como uma ação do MAB; aqui, um agente precisa escolher

¹ Para mais informações: <https://support.google.com/analytics/answer/2844870?hl=en>, acesso em 24 de março de 2014.

qual dos possíveis anúncios deve ser exibido para os usuários de forma a maximizar as receitas (PANDEY; CHAKRABARTI; AGARWAL, 2007; DEVANUR; KAKADE, 2009; PANDEY; OLSTON, 2006; CHAKRABARTI et al., 2009).

Modelagem dos MABs

O problema *Multi-armed Bandit* considera que um agente encontra repetidamente uma escolha com diferentes possíveis ações; depois de cada aplicação da ação escolhida, o agente recebe uma recompensa amostrada de uma distribuição de probabilidade estacionária que depende da ação escolhida. Neste trabalho utilizamos a modelagem do *Multi-armed Bandit* para modelar o problema de alocação de anúncios.

Este problema pode ser definido por um conjunto de ações $\mathcal{A}$ e um horizonte finito τ . Cada uma das ações $a \in \mathcal{A}$ são modeladas por uma distribuição de Bernoulli com uma probabilidade de sucesso desconhecida μ_a .

A cada instante de tempo t , $0 < t < \tau$, o agente escolhe uma ação a e recebe uma recompensa imediata $r(a, t) = \mathcal{B}(\mu_a)$, onde $\mathcal{B}(\mu_a)$ corresponde a uma amostra na distribuição de Bernoulli com valor esperado μ_a .

Pode-se considerar que o agente age seguindo uma política π . O agente procura maximizar a sua recompensa total esperada acumulada até o instante de tempo τ , ou de forma equivalente, minimizar seu arrependimento acumulado esperado ($\mathcal{R}^{-1}$):

$$\mathcal{R}^{-1}(\pi, \tau) = \tau\mu^* - \mathbf{E} \left[\sum_{t=1}^{\tau} r(\pi(t), t) \right] \quad (6.1)$$

$$= \sum_{a \in \mathcal{A}} (\mu^* - \mu_a) \mathbf{E}[N_{a,\tau}], \quad (6.2)$$

onde $\mu^* = \max_a \mu_a$ denota o valor esperado da melhor ação, $r(\pi(t), t)$ é a recompensa no instante t dada a ação a definida por uma política não estacionária π no instante t , e que $N_{a,\tau}$ é uma variável aleatória que representa o número de vezes que uma ação a foi escolhida até o instante de tempo $t = \tau$.

Multi-Armed Bandits com contexto

O problema do *Multi-Armed Bandits* com contexto é uma generalização do problema do *Multi-Armed Bandits*. Neste problema considera-se que o agente tem a capacidade de perceber o contexto (ou seu ambiente) em que está atuando.

Neste cenário, assume-se que as recompensas obtidas pelo agente são uma função da informação de contexto. Um agente neste ambiente age em cada instante de tempo t da seguinte forma (LI et al., 2010):

- i o agente observa o conjunto de ações possíveis $\mathcal{A}_t$ e um vetor de atributos $\vec{x}_t$, chamado de vetor de contexto;
- ii o agente delibera baseado em suas observações passadas e em seu vetor de contexto e escolhe uma ação $a_t \in \mathcal{A}_t$ para executar;
- iii o agente recebe uma recompensa $r(a, t) = g(\vec{x}_t, a_t)$. Retornando ao Passo i.

Considera-se que o vetor de contexto $\vec{x}_t$ possa ser composto por qualquer valor real entre 0 e 1, podendo assumir um número de dimensões arbitrário. Estes atributos podem representar, por exemplo: a distância entre um objeto alvo e um robô, a intensidade de um pixel, informações sobre um usuário navegando em um *website*, etc.

Esta modelagem permite modelar o problema de alocação de anúncios em *websites*, considerando que um usuário possa ser descrito por um vetor de atributos, utilizando as campanhas de anúncios como as ações possíveis.

6.2 Avaliação entre diferentes algoritmos para *Multi-Armed Bandits* com contexto

Em geral, quando se trabalha com o problema de *Multi-Armed Bandits*, é necessária uma avaliação comparativa entre diferentes algoritmos para uma mesma base de dados.

A prática mais comum é utilizar simulações para um ambiente *online* e verificar o valor esperado das recompensas acumuladas entre os algoritmos a serem comparados.

No entanto, neste método existe a necessidade de se criar um simulador, sendo que em geral este simulador está intrinsecamente ligado à base de dados utilizada, não permitindo facilmente a comparação entre dois algoritmos em duas bases de dados distintas (LI et al., 2011).

Desta forma, neste trabalho utilizamos o método comparativo descrito por Li et. al. (LI et al., 2011), que provê um método geral e sem viés para avaliação de algoritmos de *Multi-Armed Bandits*.

Considera-se que a recompensa total acumulada por um algoritmo A é definida por:

$$G_A(T) = \mathbf{E}_D \left[\sum_{t=1}^T r_{t,a_t} \right], \quad (6.3)$$

onde o valor esperado $\mathbf{E}_D$ é definido de acordo com um processo independente e identicamente distribuído $(\vec{x}_t, r_{t,1}, \dots, r_{t,k})$ de acordo com uma distribuição D e um algoritmo A , onde $\vec{x}$ representa o vetor de contexto.

De forma similar, define-se que a recompensa total acumulada para uma política π de um agente, dado por um mapeamento do vetor de contexto para ações, $\pi : \mathcal{X} \mapsto \mathcal{A}$, é:

$$G_\pi(T) = \mathbf{E}_D \left[\sum_{t=1}^T r_{t,\pi(x_t)} \right], \quad (6.4)$$

podendo, desta forma, comparar diferentes políticas de atuação de forma consistente. O algoritmo descrito por Li et. al. para avaliação de algoritmos de *Multi-Armed Bandits* com contexto para horizontes finitos (LI et al., 2011) utilizado neste trabalho pode ser visto no Algoritmo 4.

Algoritmo 4: Policy Evaluator (com horizonte finito)

Input: A: algoritmo,
 S fluxo de eventos de tamanho L

```

1  $h_0 \leftarrow \emptyset$  // História da avaliação, inicialmente vazia
2  $\hat{G}_A \leftarrow 0$  // Valor esperado de recompensas, inicialmente zero
3  $T \leftarrow 0$  // Contador de eventos válidos, inicialmente zero
4 for  $t = 1, 2, 3, \dots, L$  do
5    Selecione o t-ésimo evento  $(x, a, r_a)$  de S
6    if  $A(h_t, x) = a$  then
7       $h_t \leftarrow \text{CONCATENAR}(h_{t-1}, (x, a, r_a))$ 
8       $\hat{G}_A \leftarrow \hat{G}_A + r_a$ 
9       $T \leftarrow T + 1$ 
10  else
11  |  $h_t \leftarrow h_{t-1}$ 
12 return  $\hat{G}_A/T$ 
```

Desta forma, a comparação entre dois algoritmos se resume a calcular o valor G_A , para todos os algoritmos que se deseja comparar, escolhendo como melhor algoritmo o algoritmo que obter o maior valor G_A .

6.3 Redes de Anúncios como um MAB sem informação de contexto

A modelagem sem informação de contexto pode ser feita da seguinte forma: as ações são um mapeamento das campanhas, i.e. cada ação possível no *Bandit* corresponde a uma campanha de anúncios $a \in \mathcal{A}$, sendo ignoradas as informações de orçamento e de perfil de usuário, considerando que o controle de quais ações estão disponíveis seja feito de forma exógena.

Neste caso, pode ser utilizado qualquer algoritmo para MABs, como por exemplo o *Thompson Sampling* (THOMPSON, 1933), o *Upper Confidence Bound* (UCB1) (AUER;

CESA-BIANCHI; FISCHER, 2002), o Bayes-UCB (KAUFMANN; CAPPÉ; GARIVIER, 2012), ou o *AdBandit* (TRUZZI et al., 2013).

6.3.1 Thompson Sampling

Thompson Sampling (THOMPSON, 1933) é um algoritmo proposto em 1933, tendo como foco inicial modelar problemas de alocação médica e ensaios clínicos, que são essencialmente um MAB.

Apenas recentemente foi provado que neste algoritmo o arrependimento cresce de forma logarítmica (AGRAWAL; GOYAL, 2012; KAUFMANN; KORDA; MUNOS, 2012), tornando-o ótimo segundo os critérios definidos por Lai e Robbins (LAI; ROBBINS, 1985), que provaram que no problema do *Multi-armed bandits* o crescimento do arrependimento é no mínimo logarítmico.

Este algoritmo utiliza uma formulação Bayesiana para o problema do *Multi-armed bandits*. O Algoritmo *Thompson Sampling* está descrito no Algoritmo 5.

Sua ideia é escolher as ações a se executar de acordo com a probabilidade da ação escolhida ser a melhor opção. O algoritmo inicia com um conhecimento a priori de cada uma das ações disponíveis, e atualiza seu conhecimento baseando-se nas suas interações com o ambiente.

Para decidir qual ação tomar, ele amostra cada uma das distribuições que modelam as ações e escolhe a ação que obteve o maior valor na amostragem.

Seja $\theta = (\theta_1, \theta_2, \dots, \theta_{|\mathcal{A}|})$ uma estimativa das probabilidades de sucesso de cada uma das ações com probabilidades de sucesso desconhecidas $\mu = (\mu_1, \mu_2, \dots, \mu_{|\mathcal{A}|})$. O parâmetro θ é amostrado da distribuição a priori $\psi^t = \{\psi_a^t\}_{1 \leq a \leq |\mathcal{A}|}$, considerando-se que as distribuições ψ_a^t são independentes.

Seja ψ^0 uma distribuição a priori inicial. A escolha de uma ação a gera uma recompensa imediata r_t , e as distribuições que modelam as ações são então atualizadas de forma Bayesiana:

$$\psi_a^t(\theta_a | r_t) \propto \nu_{\theta_a}(r_t) \psi_a^{t-1}(\theta_a), \quad (6.5)$$

onde $\nu_{\theta_a}(r_t)$ é a distribuição que modela a função de recompensa r_t . Neste trabalho modela-se os eventos através de distribuições de Bernoulli, já que o interesse está em modelar cliques em anúncios; consequentemente utiliza-se a distribuição Beta para modelar as distribuições das ações ψ_a^t , já que a distribuição Beta é conjugada da distribuição de Bernoulli.

A distribuição Beta possui dois parâmetros: α e β , e sua função densidade de probabilidade é dada por:

$$Beta(\alpha, \beta) = \begin{cases} \frac{\theta^{\alpha-1}(1-\theta)^{\beta-1}}{B(\alpha, \beta)} & \text{se } 0 < \theta < 1, \\ 0 & \text{caso contrário,} \end{cases} \quad (6.6)$$

onde $B(\alpha, \beta)$ é a função Beta, definida como:

$$B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} \quad (6.7)$$

$$= \frac{(\alpha - 1)!(\beta - 1)!}{(\alpha + \beta - 1)!} \quad (6.8)$$

Consequentemente a regra de atualização descrita na expressão 6.5 torna-se:

$$\begin{aligned} \psi_a^t(\theta_a | r_t) &\propto \left(\theta_a^{r_t} (1 - \theta_a)^{1-r_t} \right) \left(\theta_a^{\alpha-1} (1 - \theta_a)^{\beta-1} \right) \\ &= \theta_a^{r_t + \alpha - 1} (1 - \theta_a)^{(1-r_t) + \beta - 1} \end{aligned} \quad (6.9)$$

No algoritmo descrito no Algoritmo 5, quando não se conhece uma distribuição a priori, o algoritmo utiliza $(\alpha_a^0, \beta_a^0) = (1, 1)$, onde α e β são os parâmetros da distribuição ψ_a^0 . Neste caso a distribuição Beta tem a mesma forma que uma distribuição uniforme de 0 a 1, i.e. todas as ações possuem a mesma probabilidade de serem a melhor ação; portanto são escolhidas uniformemente. Um gráfico de como a densidade de probabilidade da distribuição Beta se comporta para diferentes parâmetros de α e β pode ser visto na Figura 6.2.

Os parâmetros α e β podem ser entendidos como simples contagens do número de sucessos e falhas das ações, i.e. uma contagem do número de vezes que uma ação retornou 1 ou 0. Este fato é uma das grandes vantagens de se modelar as recompensas como distribuições de Bernoulli, além do fato de que amostrar de uma distribuição Beta pode ser feito através de aproximação de Lanczos, não sendo necessário computar explicitamente os fatoriais da distribuição (LANCZOS, 1964).

A regra de atualização em geral depende da distribuição de recompensa utilizada, no entanto nós utilizamos o método descrito por Agrawal e Goyal (AGRAWAL; GOYAL, 2012), em que pouca modificação precisa ser feita, transformando uma distribuição estocástica qualquer em uma distribuição de Bernoulli, como indicado no Algoritmo 6 na linha 4.

O arrependimento apresentado pelo algoritmo *Thompson Sampling* (KAUFMANN; KORDA; MUNOS, 2012) é limitado superiormente por:

Figura 6.2 – Densidade de Probabilidade da Função Beta para diferentes parâmetros de α e β .

Algoritmo 5: *Thompson Sampling* para MABs modelados por distribuições de Bernoulli

Entrada: τ (horizonte), $\mathcal{A}$ (conjunto de ações), α_a^0, β_a^0 (parâmetros da distribuição ψ_a^0)

- 1 **para** $t = 1$ **até** τ **faça**
 - 2 Amostre $\hat{\theta}_a$ de acordo com $Beta(\alpha_a^{t-1}, \beta_a^{t-1}) \forall a \in \mathcal{A}$
 - 3 Execute a ação $\hat{a} = \arg \max_a \hat{\theta}_a$ e observe a recompensa r
 - 4 Atualize ψ_a^t de acordo com a Expressão 6.9
-

$$(1 + \epsilon) \sum_{a \in \mathcal{A}: \mu_a \neq \mu^*} \frac{\Delta_a(\ln(\tau) + \ln(\ln(\tau)))}{\delta(\mu_a, \mu^*)} + C(\epsilon, \mu_1, \dots, \mu_{|\mathcal{A}|}), \quad (6.10)$$

onde $\epsilon > 0$, $C(\epsilon, \mu_1, \dots, \mu_{|\mathcal{A}|})$ é uma constante dependente do problema, δ_a é a diferença entre a probabilidade da ação ótima μ^* e uma ação a , e $\delta(\cdot, \cdot)$ é a divergência de Kullback-Leibler que é uma medida não simétrica da diferença entre duas distribuições de probabilidades. Este limite superior indica que o arrependimento do algoritmo é no pior caso logarítmico, podendo ser considerado um algoritmo que provê uma solução ótima, já que, como provado por Lai e Robbins, no MAB o arrependimento cresce no mínimo de forma logarítmica.

Algoritmo 6: *Thompson Sampling* para *Bandits* modelados por distribuições estocásticas em geral

Entrada: τ (horizonte), $\mathcal{A}$ (conjunto de ações), α_a^0, β_a^0 (parâmetros da distribuição ψ_a^0)

- 1 **para** $t = 1$ **até** τ **faça**
- 2 Amostre $\hat{\theta}_a$ de acordo com $Beta(\alpha_a^{t-1}, \beta_a^{t-1}) \forall a \in \mathcal{A}$
- 3 Execute a ação $\hat{a} = \arg \max_a \hat{\theta}_a$ e observe a recompensa r
- 4 Faça uma amostragem de uma distribuição de Bernoulli com probabilidade de sucesso $\hat{r}$ e observe a recompensa r
- 5 Atualize ψ_a^t de acordo com a Expressão 6.9

6.3.2 Upper Confidence Bound (UCB1)

O *UCB1* (AUER; CESA-BIANCHI; FISCHER, 2002) é um algoritmo bastante tradicional, e consiste em uma aplicação do princípio do otimismo como forma de enfrentar a incerteza proposta por Lai (LAI; ROBBINS, 1985), isto é, apesar do algoritmo não saber quais são as melhores ações para se escolher, ele faz uma escolha otimista sobre o valor esperado de cada ação.

Neste algoritmo a tomada de decisão é baseada em um índice. Este índice consiste da soma de dois termos, o primeiro sendo a média da recompensa de cada ação possível e o segundo sendo um valor relacionado ao intervalo de confiança considerando uma cauda de acordo com o limite de Chernoff-Hoeffding (HOEFFDING, 1963), que provê um limite superior para a probabilidade de que a soma de variáveis aleatórias se desviem do seu valor esperado. O algoritmo pode ser visto no Algoritmo 7.

Algoritmo 7: Algoritmo UCB1

Entrada: τ (horizonte),
 $\mathcal{A}$ (ações)

- 1 Aplique cada uma das ações a uma vez
- 2 Observe as recompensas $r_a \forall a$
- 3 Seja $k_a = 1, \forall a \in \mathcal{A}$
- 4 Seja $\hat{\mu}_a = \frac{r_a}{k_a}$
- 5 **para** $t = |\mathcal{A}|$ **até** τ **faça**
- 6 Escolha a ação $\hat{a} = \arg \max_a \left(\hat{\mu}_a + \sqrt{\frac{2 \ln(t)}{k_a}} \right)$
- 7 Observe a recompensa r
- 8 $r_{\hat{a}} = r_{\hat{a}} + r$
- 9 $k_{\hat{a}} = k_{\hat{a}} + 1$
- 10 Faça $\hat{\mu}_{\hat{a}} = \frac{r_{\hat{a}}}{k_{\hat{a}}}$

Sua regra de decisão, descrita no Passo 6 do Algoritmo 7, controla tanto a exploração quanto a exploração. A exploração ocorre graças ao segundo fator da soma que corres-

ponde ao limite superior descrito acima. Este fator de exploração cresce com o tempo e decai com o número de vezes que a ação a for escolhida.

O algoritmo, desta forma, explora enquanto este fator é pequeno, e, a cada exploração, o fator de exploração das ações não escolhidas cresce até que a soma entre a média e este fator seja maior que a de alguma outra ação, fazendo com que o algoritmo explore novamente.

6.3.3 Bayes-UCB

O Bayes-UCB (KAUFMANN; CAPPÉ; GARIVIER, 2012) assume uma modelagem Bayesiana para o MAB de forma semelhante ao *Thompson Sampling*. No entanto, a tomada de decisão é baseada na função quantil (o inverso da função distribuição acumulada) da distribuição Beta.

Assim como o algoritmo *Thompson Sampling* descrito no Algoritmo 5, modela-se as ações com distribuições Beta, e considera-se que uma distribuição inicial para as ações é dada, assim como um parâmetro c para afinar sua resposta. O Bayes-UCB pode ser visto no Algoritmo 8. No algoritmo $Q(p, \alpha, \beta)$ corresponde à função quantil para a probabilidade p da distribuição Beta com parâmetros α e β .

Sua idéia consiste em utilizar a função quantil para criar intervalos de confiança sobre a probabilidade de sucesso de cada uma das ações, escolhendo a ação que possuir a maior confiança de ser a melhor ação em cada instante de tempo.

Algoritmo 8: Algoritmo Bayes-UCB

Entrada: τ (horizonte), $\mathcal{A}$ (ações), Π^0 (distribuição inicial de θ), c (parâmetro do problema)

1 **para** $t = 1$ **to** τ **faça**

2 **para** cada ação $a = 1, \dots, |\mathcal{A}|$ **faça**

3

$$q_a(t) = Q\left(1 - \frac{1}{t(\log \tau)^c}, \alpha_a^{t-1}, \beta_a^{t-1}\right) \quad (6.11)$$

4 Escolha a ação $\hat{a}_t = \arg \max_a q_a(t)$

5 Observe a recompensa r_a^t e atualize Π^t de acordo com a Expressão 6.9

6.3.4 Algoritmo *AdBandit*

O algoritmo *AdBandit* foi inspirado no Thompson Sampling combinando-o com uma estratégia de exploração frequentista. Este algoritmo considera a distribuição a priori para cada uma das ações dadas, assim como o tamanho do horizonte do problema, e possui um

parâmetro chamado fator de exploração ϵ , que pode ser modificado conforme a necessidade do problema.

A sua ideia básica é modelar cada uma das ações do MAB como uma distribuição Beta, assim como o algoritmo *Thompson Sampling*. O Algoritmo se divide em dois diferentes estágios: o estágio de exploração e o estágio de exploração, que ocorrem de acordo com uma probabilidade calculada utilizando o fator de exploração ϵ , o tempo e o número de épocas de decisão, sendo que a probabilidade de se explorar cai ao longo do tempo. No final de cada estágio, o algoritmo observa a recompensa e atualiza o modelo da ação escolhida.

O estágio de exploração é baseado no *Thompson Sampling*, neste caso o algoritmo explora utilizando como base a probabilidade das ações serem a melhor ação possível.

No estágio de exploração o algoritmo se baseia em uma estratégia frequentista, utilizando o valor esperado da distribuição Beta como índice para a seleção da ação. O motivo por trás desta regra de seleção é tentar acelerar a convergência das distribuições, ignorando nestes casos as caudas da distribuição. Com esta ideia, o algoritmo no estágio de exploração age baseando-se no valor esperado da recompensa de cada ação $\hat{\mu}_a$:

$$\hat{\mu}_a = \frac{S_a}{S_a + F_a}, \quad (6.12)$$

onde S_a é o número de vezes que a ação a escolhida obteve um resultado positivo (clique), e F_a é o número de vezes que a ação a escolhida não obteve um resultado positivo. O algoritmo, portanto, neste estágio utiliza a seguinte regra de seleção:

$$\hat{a} = \arg \max_a (\hat{\mu}_a) \quad (6.13)$$

onde $\hat{a}$ é a ação escolhida.

Em cada instante de tempo o algoritmo decide se irá fazer um estágio de exploração ou de exploração, com probabilidade igual a $\frac{\epsilon}{\epsilon + t}$, o que faz com que o algoritmo explore mais no início dos tempos e aos poucos passe a explorar mais. O algoritmo pode ser visto no Algoritmo 9.

6.3.5 Comparação entre os algoritmos de MAB

Como existem diversos algoritmos para MABs, e nenhum ainda se provou teoricamente melhor que os outros, é necessário fazer experimentos como forma de avaliação entre os diversos algoritmos.

Algoritmo 9: Algoritmo Adbandit

Entradas: τ (horizonte), $\mathcal{A}$ (ações), (α, β) (parâmetros iniciais das distribuições que modelam as ações), ϵ (fator de exploração)

```

1  $S_a = 0, F_a = 0, \forall a \in \mathcal{A}$ 
2 para  $t = 1$  to  $\tau$  faça
3    $g =$  número aleatório uniformemente distribuído entre  $]0, 1]$ 
4   se  $g > \frac{t}{\epsilon\tau}$  então
5     // Exploração
6     para cada ação  $a = 1, \dots, |\mathcal{A}|$  faça
7       Amostre  $\theta_a$  da distribuição  $Beta(S_a + \alpha_a, F_a + \beta_a)$ 
8       Utilize a ação  $\hat{a} = \arg \max_a \theta_a$ 
9   senão
10    // Exploração
11    Utilize a ação  $\hat{a} = \arg \max_a \mu_a$ 
12  Observe a recompensa  $r$ 
13  se  $r = 1$  então
14     $S_{\hat{a}} = S_{\hat{a}} + 1$ 
15  senão
16     $F_{\hat{a}} = F_{\hat{a}} + 1$ 
17   $\hat{\mu}_{\hat{a}} = \frac{S_{\hat{a}}}{S_{\hat{a}} + F_{\hat{a}}}$ 

```

Desta forma, realizou-se uma análise comparativa entre os algoritmos para MABs apresentados, seguindo os experimentos descritos por Kaufmann e outros (KAUFMANN; KORDA; MUNOS, 2012) que consideram um *toy problem*, em que existem 10 diferentes ações possíveis, com suas probabilidades definidas na Tabela 6.1, em que a ação ótima encontra-se destacada em negrito.

Tabela 6.1 – Probabilidades utilizadas no experimento.

Ações	Probabilidade de sucesso
1 ^a	0.02
2 ^a	0.02
3 ^a	0.02
4 ^a	0.10
5 ^a	0.05
6 ^a	0.05
7 ^a	0.05
8 ^a	0.01
9 ^a	0.01
10 ^a	0.01

Todos os algoritmos foram escritos em Python, e para os algoritmos UCB1, Bayes-UCB e *Thompson Sampling* utilizou-se a implementação disponível no projeto *pymaBandits* 1.0 escrito por Olivier Cappé et. al².

Utilizou-se como parâmetros iniciais do *Thompson Sampling* e do *AdBandit* distribuições Beta com $\alpha_a = 1$ e $\beta_a = 1$ para todas as ações a . Para o algoritmo Bayes-UCB utilizou-se o parâmetro $c = 0$, como descrito pelos autores como forma de se obter o melhor desempenho. Utilizou-se $\epsilon = 0.5$ para o algoritmo *AdBandit*.

O desempenho de cada um dos algoritmos foi comparado utilizando-se a média do valor esperado acumulado do arrependimento sobre 1.000 simulações.

A Figura 6.3 compara os quatro algoritmos, onde se pode observar que o algoritmo UCB1 tem um desempenho sofrível quando comparado aos outros algoritmos, com um arrependimento médio em torno de 700.

Figura 6.3 – Arrependimento médio para um problema com horizonte igual a 15.000 tomadas de decisão para os algoritmos: *AdBandit*, *Thompson Sampling*, Bayes-UCB e UCB1, médias calculadas sobre 1.000 simulações.

Na Figura 6.4 apresentam-se apenas os algoritmos: *AdBandit*, *Thompson Sampling* e Bayes-UCB. Dentre estes, o algoritmo Bayes-UCB apresentou o pior desempenho com um arrependimento médio em torno de 110, seguido pelo *Thompson Sampling* com um

² Disponível em: <http://mloss.org/software/view/415/> acessado pela última vez em 21 de Março de 2014.

arrependimento médio em torno de 85, e pelo *AdBandit* com um arrependimento médio perto de 65, obtendo o melhor desempenho entre os algoritmos.

Figura 6.4 – Arrependimento médio para um problema com horizonte igual a 15.000 tomadas de decisão para os algoritmos: *AdBandit*, *Thompson Sampling* e *Bayes-UCB*; médias calculadas sobre 1.000 simulações.

A Figura 6.5 mostra a distribuição do arrependimento acumulado sobre as 1.000 simulações para os três melhores algoritmos. O Algoritmo *AdBandit* provou ser o melhor com a menor mediana, e com 75% dos seus resultados abaixo da mediana do segundo melhor algoritmo (*Thompson Sampling*). O algoritmo Bayes-UCB foi o algoritmo com menor número de discrepâncias, no entanto com o maior arrependimento dentre todos, com uma mediana perto de 110, enquanto o algoritmo *Thompson Sampling* possui uma mediana perto de 90.

Figura 6.5 – Distribuição do arrependimento considerando um experimento com horizonte de 15.000 tomadas de decisão, valores calculados sobre 1.000 simulações.

6.4 Redes de Anúncios como um MAB com informação de contexto

Assim como o MAB sem contexto, o MAB com contexto foi extensivamente estudado na literatura, e referenciado com nomes diferentes (AGRAWAL; GOYAL, 2013), tais como: *bandit problems with co-variates* (WOODROOFE, 1979; SARKAR et al., 1991), *associative reinforcement learning* (KAELBLING, 1994), *associative bandit problems with expert advice* (AUER et al., 2002) e *linear bandits* (DANI; HAYES; KAKADE, 2008; BUBECK et al., 2012).

A modelagem apresentada anteriormente para o MAB considera que a probabilidade de clique depende apenas do anúncio escolhido, o que, em geral, não ocorre na prática. Poder-se-ia considerar um modelo um pouco mais complexo e utilizar um conjunto de *bandits*, mantendo ainda a possibilidade de aplicação direta dos algoritmos apresentados na Seção anterior de forma simples, como ilustra a Figura 6.6, isto é, para cada perfil de usuário construir um *bandit* em que as ações possíveis dependem do perfil do usuário que fez uma requisição, aprendendo assim o CTR para cada par perfil de usuário e campanha $(i, k) \in \mathcal{G} \times \mathcal{C}$.

Nesta abordagem, tem-se diversos *bandits* sendo considerados, um para cada perfil de usuário $i \in \mathcal{G}$, e a cada requisição a rede de anúncios, deve-se verificar qual o perfil de usuário e acionar o *bandit* que está aprendendo o comportamento daquele perfil específico.

Uma limitação importante desta abordagem é que, apesar de se aprender a probabilidade de clique para cada campanha a dado um perfil de usuário i ($P(a|i)$), ela não utiliza a informação agregada das probabilidades de clique das campanhas ($P(a)$), o que pode

Figura 6.6 – Conjunto de *Bandit*. A cada requisição, verifica-se o perfil do usuário, e utiliza-se um *bandit* específico.

piorar o desempenho da rede de anúncios. Torna-se mais adequado, neste caso, modelar o problema como um problema de *Multi-Armed Bandit* com contexto, como descrito na Subseção 6.1.1, considerando que um usuário é descrito por um vetor de atributos $\vec{x}$, sendo este vetor a informação de contexto apresentada para o algoritmo de MAB com contexto.

Diversos algoritmos foram propostos para atacar este problema, como por exemplo: o algoritmo EXP4 (LANGFORD; ZHANG, 2007), o *Linear Bayes Policy* (LBP) (MARTÍN; VARGAS, 2013), LINUCB (CHU et al., 2011) e o *Thompson Sampling for Contextual bandits* (TSCb) (AGRAWAL; GOYAL, 2013).

Dentre estes algoritmos, consideramos dois os mais relevantes: o LBP, por sua simplicidade de modelagem e extensivos experimentos, e o TSCb por seu rigor matemático ao provar o seu limite de convergência, resolvendo um problema aberto proposto por Chappelle e Li (CHAPELLE; LI, 2012) sobre a convergência de MAB em um ambiente com contexto. Os algoritmos TSCb e LBP são detalhados respectivamente nas Subseções 6.4.1 e 6.4.2, e na Subseção 6.4.3 é descrita uma proposta de algoritmo para este problema.

6.4.1 *Thompson Sampling for Contextual Bandits*

O algoritmo TSCb proposto por Agrawal e Goyal (AGRAWAL; GOYAL, 2013) é extremamente interessante, sendo o único que provou garantias de convergência, resolvendo um problema em aberto proposto por Chappelle e Li (CHAPELLE; LI, 2012). Ele utiliza a ideia do *Thompson Sampling*, no entanto, modela as recompensas como uma função linear

dos atributos de contexto, utilizando uma distribuição Gaussiana multidimensional para modelar cada uma das ações possíveis. Com esta modelagem as recompensas dependem apenas de como os atributos de contexto se relacionam às ações possíveis.

Embora este algoritmo apresente limites de convergência próximos à solução ótima do problema, os atributos são modelados como distribuições Gaussianas multidimensionais necessitando calcular a inversa da matriz de covariância em cada instante de tempo, tornando esta solução computacionalmente custosa. Este algoritmo pode ser visto no Algoritmo 10.

Algoritmo 10: *Thompson Sampling for Contextual Bandits*

```

1  Seja  $B = I_d$ 
2  Seja  $\hat{\mu} = 0_d$ 
3  Seja  $f = 0_d$ 
4  Seja  $t = 1$ 
5  para cada instante de tempo faça
6      Amostre  $\tilde{\mu}(t)$  da distribuição  $\mathcal{N}(\hat{\mu}, v^2 B^{-1})$ 
7      Utilize a ação  $a(t) = \arg \max_a b_a(t)^T \tilde{\mu}(t)$ 
8      Observe a recompensa  $r_t$ 
9      Atualize  $B = B + b_{a(t)}(t) b_{a(t)}(t)^T$ 
10     Atualize  $f = f + b_{a(t)} r_t$ 
11     Atualize  $\hat{\mu} = B^{-1} f$ 
12      $t = t + 1$ 

```

No Algoritmo 10, I_d corresponde à matriz identidade com dimensão $d \times d$ e 0_d corresponde ao vetor zero de dimensão d .

A cada instante de tempo o algoritmo gera uma amostra de dimensão d $\tilde{\mu}(t)$ da distribuição gaussiana $\mathcal{N}(\hat{\mu}, v^2 B^{-1})$, onde d corresponde ao número de atributos do vetor de contexto, v^2 é um fator de normalização da matriz de covariância, dado por:

$$v = R \sqrt{\frac{24}{\epsilon} d \ln\left(\frac{1}{\delta}\right)},$$

onde R , ϵ e δ são parâmetros do algoritmo.

Escolhe-se a ação a que maximiza $b_a(t)^T \tilde{\mu}(t)$, onde b_a corresponde à distribuição gaussiana que modela o contexto dado a ação a .

6.4.2 Linear Bayes Policy

O algoritmo LBP, proposto por Martín e Vargas (MARTÍN; VARGAS, 2013) tem como pontos fortes a simplicidade de modelagem e eficácia experimental.

O algoritmo inicia modelando cada uma das ações a através de sua probabilidades de click $P(a)$, e para cada um dos atributos de contexto modela a sua probabilidade de click

$x_i \in \vec{x}$ dada a ação a ($P(x_i|a)$), e também a probabilidade de ocorrência de um atributo de contexto $P(x_i)$. O algoritmo considera que todas as probabilidades anteriormente citadas iniciam com 100% de probabilidade de sucesso.

A cada instante de tempo o algoritmo decide qual ação utilizar através da seguinte regra de decisão:

$$a(t) = \arg \max_a \left(\prod_i P(x_i|a) \right) \sum_i P(a) \frac{P(x_i|a)}{P(x_i)}. \quad (6.14)$$

Esta regra de decisão combina a probabilidade conjunta de todas as probabilidades $P(x_i|a)$ assumindo independência (primeiro termo do produto) com uma aproximação da regra de Bayes (segundo termo do produto).

Neste algoritmo utiliza-se uma visão frequentista das probabilidades, considerando apenas simples contagens do número de utilizações e do número de sucessos. A cada tomada de decisão o algoritmo atualiza estas contagens e, conseqüentemente, as probabilidades.

Desta forma, já que o algoritmo considera que as probabilidades iniciam-se com 100% de chance de sucesso, a exploração ocorre de forma otimista considerando que cada nova ação inserida no problema sempre é a melhor ação, mantendo esta opção até que a sua probabilidade se torne pior que a de alguma outra ação.

Este algoritmo participou da competição *Exploration and Exploitation Challenge 3*, em 2012 promovida pelo ICML, obtendo o segundo lugar na competição³.

6.4.3 *Extended Thompson Sampling* (ETS)

Neste contexto, propomos o algoritmo *Extended Thompson Sampling* (ETS), que foi construído para agir em ambientes dinâmicos, onde diferentes ações podem surgir em cada instante de tempo.

O algoritmo pode ser visto no Algoritmo 11. Ele considera a existência de um conjunto de ações conhecidas inicialmente vazio, e um conjunto de atributos conhecidos também inicialmente vazio. Em cada instante de tempo o algoritmo verifica quais são as ações possíveis e quais os atributos que modelam o usuário que fez uma requisição naquele instante de tempo, i.e. o agente (controlado pelo algoritmo) observa seu ambiente para reconhecer as possíveis ações e o contexto em que ele está atuando.

Mais formalmente, em cada instante de tempo t o agente verifica se as ações disponíveis já estão em sua base de conhecimento $\mathcal{F}$, e caso alguma delas ainda não seja conhecida,

³ Para mais informações sobre a competição visite <https://explochallenge.inria.fr/category/challenge/>

modela-se a ação como uma distribuição Beta ψ_a^{t-1} com parâmetros $\alpha = 1$ e $\beta = 1$. De forma análoga, para cada atributo disponível no vetor de contexto, o agente verifica se o atributo já é conhecido em sua base de conhecimento $\mathcal{X}$, e caso o atributo ainda não seja conhecido, modela-se o atributo como uma distribuição Beta ρ_{a,x_i}^{t-1} com parâmetros $\alpha = 1$ e $\beta = 1$ para cada uma das ações possíveis.

Para tomar uma decisão, e decidir qual ação deve ser escolhida dentre as ações possíveis em um instante de tempo em particular, o agente amostra $\hat{\theta}_a$ do modelo ψ_a^{t-1} para cada uma das ações disponíveis a e, de forma similar, amostra ζ_{a,x_i} do modelo ρ_{a,x_i}^{t-1} para cada um dos atributos x_i dado uma ação a .

A decisão é feita calculando-se o valor $k_a = \hat{\theta}_a \prod_{a,x_i \in \vec{x}_t} \zeta_{a,x_i}$. O agente então escolhe a ação que obtiver o maior valor k_a .

Considera-se que a ação retorna imediatamente uma recompensa $\hat{r}$ que pode estar entre 0 e 1. O agente então faz uma amostragem em uma distribuição de Bernoulli com probabilidade de sucesso igual a $\hat{r}$, e observa a recompensa r . Finalmente, atualiza o seu conhecimento de acordo com a Expressão 6.9, tanto para o modelo da ação escolhida, quanto para o modelo dos atributos disponíveis no vetor de contexto dado a ação. Note que este teste de Bernoulli só é necessário caso as recompensas não sejam modeladas como uma distribuição de Bernoulli.

O algoritmo desta forma está aprendendo a probabilidade de sucesso das ações e como cada um dos atributos se relaciona com o valor esperado de uma ação. Em última análise, o valor esperado de uma ação em um instante de tempo t , ignorando o contexto, é dado por:

$$\mathbf{E}[r(a)] = \mathbf{E}[\psi_a^t] = \frac{\alpha_a^t}{\alpha_a^t + \beta_a^t}. \quad (6.15)$$

O valor esperado de uma ação em um instante de tempo t considerando o contexto é dado por:

$$\mathbf{E}[r(a|\vec{x})] \propto \mathbf{E}[\psi_a^t] \times \mathbf{E}[\rho_{a,x_i}^{t-1}] \quad (6.16)$$

$$\propto \frac{\alpha_a^t}{\alpha_a^t + \beta_a^t} \prod_{x_i \in \vec{x}} \frac{\alpha_{a,x_i}^t}{\alpha_{a,x_i}^t + \beta_{a,x_i}^t} \quad (6.17)$$

A Expressão 6.16 é apenas uma regra de Bayes ignorando o fator de normalização, já que ele não é importante para a tomada de decisão, evitando uma computação desnecessária. No entanto o algoritmo ETS não utiliza diretamente estes valores esperados, mas

sim aprende um modelo para a ação e como os atributos de contexto se relacionam com a ação, considerando a distribuição completa do modelo e não apenas valores esperados durante a tomada de decisão.

Implicitamente, assim como o *Thompson Sampling*, o agente está tomando as decisões baseadas na probabilidade destas decisões serem as melhores possíveis, dado o seu conhecimento das ações disponíveis e do contexto.

Este fator é extremamente importante em cenários que mudam rapidamente, onde as possíveis ações podem ser diferentes em cada instante de tempo, e em que os atributos que modelam o contexto também têm sua relação com o sucesso das ações modificado ao longo do tempo. A distribuição Beta irá se adaptar às mudanças, diminuindo a possibilidade de escolha de ações ruins que eventualmente tiveram um início tendencioso, i.e. ações ruins que pareciam ser boas no início.

O agente fica trabalhando em um ciclo de observação, ação e atualização até que algum critério de parada especificado seja encontrado, como por exemplo o esgotamento das tomadas de decisão em ambientes com horizonte finito.

Algoritmo 11: *Extended Thompson Sampling* para MABs modelados por distribuições estocásticas (ETS).

```

1  Seja  $t = 0$ 
2  Seja  $\mathcal{F}$  um conjunto de ações conhecidas
3  Seja  $\mathcal{X}$  um conjunto de atributos conhecidos modelados para cada ação
4  repita
5      Observe as possíveis ações  $\mathcal{A}$ 
6      Observe o vetor de contexto  $\vec{x}_t$ 
7      para cada ação  $a \in \mathcal{A}$  faça
8          se  $a \notin \mathcal{F}$  então
9              Inicialize  $\psi_a^{t-1}$  com uma distribuição Beta com parâmetros  $\alpha_a^{t-1} = 1$  e
10              $\beta_a^{t-1} = 1$ 
10             Adicione  $a$  em  $\mathcal{F}$ 
11             para cada atributo disponível  $x_i \in \vec{x}_t$  e  $x_i \notin \mathcal{X}_a$  faça
12                 Inicialize  $\rho_{a,x_i}^{t-1}$  com uma distribuição Beta com parâmetros  $\alpha_{a,x_i}^{t-1} = 1$  e
13                  $\beta_{a,x_i}^{t-1} = 1$ 
13                 Adicione  $x_i$  em  $\mathcal{X}_a$ 
14             Amostre  $\hat{\theta}_a$  de acordo com a distribuição  $Beta(\alpha_a^{t-1}, \beta_a^{t-1}) \forall a \in \mathcal{A}$ 
15             Amostre  $\zeta_{a,x_i}$  de acordo com a distribuição  $Beta(\alpha_{a,x_i}^{t-1}, \beta_{a,x_i}^{t-1}) \forall (a, x_i) \in \mathcal{A} \times \mathcal{X}_a$ ;
16             Escolha a ação  $\hat{a} = \arg \max_a \hat{\theta}_a \prod_{x_i \in \vec{x}_t} \zeta_{a,x_i}$  e observe sua recompensa  $\hat{r}$ 
17             Faça um teste de Bernoulli com probabilidade de sucesso  $\hat{r}$  e observe a
             recompensa  $r$ 
18             Atualize  $\psi_a^t$  de acordo com a Expressão 6.9
19             Atualize  $\rho_{a,x_i}^t \forall (a, x_i) \in \mathcal{A} \times \mathcal{X}_a$  de acordo com a Expressão 6.9
20              $t = t + 1$ 
21 até algum critério de parada ser encontrado

```

6.5 Comparação entre algoritmos de MAB com contexto

Como forma de comparação entre os algoritmos para MAB com contexto utilizou-se a base de dados gentilmente fornecida pela empresa Yahoo para esta pesquisa chamada *User Click Log Dataset*. Esta base de dados corresponde a uma coleção de registros das interações de usuário com o módulo *Today Module*⁴, em que o Yahoo escolhe quais notícias ele considera mais relevante para cada usuário, o *Today Module* pode ser visto na Figura 6.7.

⁴ Juntamente com a base de dados a empresa Yahoo também disponibiliza uma ferramenta de visualização e exploração de dados disponível em: <http://visualize.yahoo.com/core/>.

Figura 6.7 – *Today Module* extraído de <http://us.yahoo.com>, visita em: 18 de fevereiro de 2014.

Esta base de dados foi escolhida por representar um problema clássico de MAB com contexto, e pela não existência de bases de dados abertas do problema de alocação de anúncios. É importante ressaltar que a comparação entre estes algoritmos utilizando esta base de dados é válida para se analisar o problema de alocação de anúncios, e que a única diferença de modelagem apresentada entre o problema de alocação de anúncios e o problema de alocação de notícias, é que no caso do problema de alocação de notícias, considera-se que o custo por clique seja sempre igual a 1.

Cada linha da base de dados consiste em um registro da interação de um usuário com o *Today Module*. Este registro é composto pela notícia recomendada, um valor indicando se o usuário clicou ou não na notícia recomendada, um vetor de contexto composto por atributos do usuário que viu essa recomendação, e um conjunto das notícias que poderiam ser recomendadas no instante em que a notícia foi exibida para o usuário.

O vetor de contexto do usuário é composto por 136 atributos binários que correspondem a informações do usuário, tais como: idade, sexo, região onde o usuário habita,

comportamento do usuário frente a diferentes tipos de notícias, entre outras. Estes atributos podem estar presentes ou não, dependendo se o usuário estava logado ou não na página do Yahoo quando via o módulo. É importante ressaltar que, apesar destes atributos representarem informações do usuário, não é fornecido o significado de cada atributo.

Uma característica importante desta base de dados é que as notícias foram exibidas aos usuários de forma uniforme, ou seja, durante a coleta destes dados as notícias eram escolhidas uniformemente dentre as notícias disponíveis naquele instante de tempo. Esta propriedade permite que uma avaliação sem viés seja conduzida, conforme o algoritmo descrito por Li e outros (LI et al., 2011), descrito na Seção 6.2.

A primeira linha da base de dados está descrita na Tabela 6.2. Na base de dados, em média, existem pelo menos 30 diferentes notícias a cada instante de tempo, e suas probabilidades de clique não são constantes ao longo do tempo. Em nossos experimentos, descritos a seguir, foram utilizados as 9,310 milhões de registros.

Tabela 6.2 – Informação do primeiro registro da base de dados do *Today Module*.

Informação	Valor
<i>Timestamp</i>	1317513291
Notícia exibida	id-560620
Clique	Falso
Atributos	1, 9, 11, 13, ..., 43, 14, 39, 30, 66, 50, 27, 104, 20
Notícias disponíveis	id-552077, id-555224, id-555528, ... ,id-565822

A Tabela 6.3 mostra alguns algoritmos simples avaliados para esta base de dados, a saber:

- Notícia mais nova: apresenta sempre a notícia mais recente dentre as notícias disponíveis;
- Notícia aleatória: apresenta uma notícia escolhida aleatoriamente dentre as notícias disponíveis;
- Notícia mais velha: apresenta sempre a notícia mais antiga dentre as notícias disponíveis.

Pode-se notar que escolher notícias mais novas sempre é uma opção melhor que escolher notícias aleatoriamente ou a notícia mais antiga.

Os experimentos foram conduzidos utilizando a biblioteca disponibilizada pela organização da competição *Exploration and Exploitation Challenge 3*⁵. Foram utilizados os algoritmos TSCb, LBP e o ETS.

⁵ Disponível em <https://explochallenge.inria.fr/category/challenge/>, acesso em 18 de fevereiro de 2014

Tabela 6.3 – Avaliação de algoritmos simples para a base de dados sobre os 9,310 milhões de registros.

Algoritmo	Probabilidade de Clique $\times 10.000$
Notícia mais nova	512,6
Notícia aleatória	366,7
Notícia mais velha	266,7

A Figura 6.8 apresenta o desempenho dos algoritmos TSCb, LBP, ETS e os algoritmos simples: notícia mais nova, notícia mais velha e notícias aleatórias.

Figura 6.8 – Comparação de desempenho medido em *CTR* entre os algoritmos LBP, TSCb, e ETS para os primeiros 9,31 milhões de registros da base de dados *R6B - Yahoo! Front Page Today Module User Click Log Dataset, version 2.0*.

Nota-se que o algoritmo LBP apresentou o melhor desempenho no final da avaliação, sendo seguido pelo ETS e pelo TSCb. O motivo da diferença entre os algoritmos ETS e o LBP, já que ambos baseiam-se na regra de Bayes, pode ser atribuído ao fato de que o algoritmo LBP escolhe sempre as notícias mais novas primeiro, enquanto o algoritmo ETS atua de forma probabilística, criando modelos através das distribuições Beta, não explorando necessariamente as notícias mais novas de forma mais agressiva.

Desta forma o algoritmo ETS mostrou sua capacidade de aprender em problemas de MABs com um grande vetor de contexto, em que as probabilidades de cada ação variam ao longo do tempo, assim como os atributos se relacionam com as ações. O ambiente testado é altamente aderente ao problema proposto, o qual esta dissertação se propõe a tratar,

podendo os algoritmos serem utilizados sem modificações para a solução de problemas de alocação de anúncios em *websites*.

Os resultados apresentados indicam que tanto o algoritmo ETS quanto o algoritmo LBP são as melhores opções dentre os algoritmos estudados. No entanto, apesar do algoritmo LBP ter um resultado melhor que o algoritmo ETS, este ganho de desempenho é pequeno, e advém da estrutura do problema em que as notícias mais novas sempre possuem uma maior probabilidade de clique, o que não necessariamente é verdade em um problema de alocação de anúncios. Conclui-se, portanto, que o algoritmo ETS deve ser utilizado, por sua maior generalidade até que exista um estudo que indique qual o comportamento da probabilidade de clique das campanhas de anúncios de acordo com o tempo.

Quanto a complexidade computacional, os dois algoritmos possuem a mesma complexidade assintótica, no entanto o algoritmo ETS faz amostragem sobre as distribuições, sendo, portanto, pouco um mais custoso em número de operações.

7 CONCLUSÃO E TRABALHOS FUTUROS

Este trabalho de mestrado investigou a modelagem e soluções para o problema de alocação de anúncios em *websites*. Explicamos o problema real, formalizando-o como um processo estocástico (Capítulo 2).

Inicialmente, abordamos o problema como um MDP, criando uma modelagem ótima para o problema, assumindo que as probabilidades de clique e os perfis de usuário fossem dados do problema (Seção 4.2 do Capítulo 4).

A partir desta modelagem inicial, avaliou-se como a diferença da solução ótima, a partir da modelagem através de MDPs, se comporta frente a soluções baseadas em aproximações lineares (revisadas no Capítulo 3). Como conclusão desta análise, mostrou-se, no Capítulo 5 que, apesar de uma solução baseada na modelagem ótima poder ser até 58.2% melhor que soluções baseadas em aproximações lineares, em problemas reais esta diferença diminui chegando em nossos experimentos a até 0.4%. Além disso, dado o tamanho do espaço de estados necessário para a solução através de MDPs, fica evidente que as aproximações lineares podem ser consideradas uma boa solução para este problema.

Em seguida, no Capítulo 6, argumentamos que tanto a solução através de MDPs, como as soluções baseadas em aproximações lineares possuem um sério problema: a necessidade de um modelo que capture o comportamento das probabilidades de clique, o que não é factível em situações reais, dado que o comportamento dos usuários frente aos anúncios pode se modificar rapidamente. E com isto, formalizamos o problema como um problema de Aprendizado por Reforço, utilizando uma modelagem baseada em *Multi-Armed Bandits*.

Aqui utilizamos duas formas de modelagem, a primeira assumindo um *Multi-Armed Bandit* sem contexto, baseando-se apenas nas probabilidades de clique associadas a um perfil de usuário e uma campanha de anúncios, propondo um algoritmo chamado *Ad-Bandit*, e comparando-o com as principais técnicas de solução desta categoria de MAB. Na segunda modelagem utilizou-se uma abordagem através de MABs com contexto (Seção 6.4), propondo uma extensão ao algoritmo *Thompson Sampling* chamado *Extended Thompson Sampling*, que foi comparado com outros dois algoritmos utilizados para a solução deste tipo de problema, o TSCb e o LBP, utilizando uma base de dados reais, disponibilizada pela empresa Yahoo.

Durante este trabalho de mestrado foram publicados os seguintes artigos:

- “*Markov Decision Processes for Ad Network Optimization*” (TRUZZI et al., 2012), que apresentou a modelagem inicial através de MDPs;

- “*AdBandit: A New Algorithm For Multi-Armed Bandits*” (TRUZZI et al., 2013), que apresentou a modelagem baseada em MAB e o algoritmo *AdBandit*;
- “*Ad Network Optimization: Evaluating Linear Relaxations*” (TRUZZI et al., 2013), que apresentou a comparação entre a solução ótima através de MDPs e aproximações lineares do problema, obtendo o prêmio de terceiro melhor artigo da conferência BRACIS 2013.

Além destes trabalhos, encontra-se em finalização um outro artigo a ser enviado para publicação em periódico, apresentando a modelagem através de MABs com contexto, e o algoritmo proposto neste trabalho o *Extended Thompson Sampling*.

Como trabalhos futuros, pode-se fazer uma análise do comportamento temporal das probabilidades de clique para se verificar experimentalmente, em um ambiente real, como estas probabilidades se comportam, de forma semelhante ao que foi feito para a base de dados de notícias utilizadas neste trabalho. Tal análise ajudaria a entender qual a melhor forma de proceder ao se escolher um algoritmo para o problema de alocação de anúncios em *websites*, e a partir desta análise rever a opção entre os algoritmos ETS e LBP, ou até mesmo propor um novo algoritmo que se adeque melhor a este comportamento.

Também é necessário uma análise teórica e experimental do comportamento do algoritmo ETS contra soluções baseadas em aproximações lineares, adequando estas soluções ao ambiente mais dinâmico que o algoritmo ETS se propõe a resolver. Pode-se utilizar, por exemplo, uma solução baseada em aproximação linear com replanejamento, analisando qual seria o horizonte ótimo a ser considerado para se estimar valores de CTR, ou até mesmo, a utilização da solução por aproximação linear aliada a algum algoritmo de MAB, utilizando a saída da programação linear como entrada do MAB. Além disso, seria interessante uma análise comparativa detalhada entre as heurísticas de utilização HLP e SLP propostas por Girgin e outros (GIRGIN et al., 2010a).

Ainda no contexto do problema de alocação de anúncios, deve-se estudar a possibilidade de utilizar o algoritmo ETS em problemas de RTB utilizando a estimativa de CTR dada pelo algoritmo para criar um lance para o leilão de Vickrey.

Pode-se também estudar qual a aplicabilidade do algoritmo ETS em outros problemas de aprendizado por reforço como, por exemplo, a navegação robótica, onde se poderia analisar a aplicabilidade do algoritmo ETS como forma de descobrir qual a melhor ação a ser executada em um problema de navegação de acordo com um vetor de atributos de contexto que indique qual o ambiente em que o robô está atuando.

REFERÊNCIAS BIBLIOGRÁFICAS

AGRAWAL, S.; GOYAL, N. Analysis of thompson sampling for the multi-armed bandit problem. In: *Proceedings of the 25th Annual Conference On Learning Theory (COLT)*. Edinburgh, Schotland: JMLR W&CP, 2012. p. 127–135.

AGRAWAL, S.; GOYAL, N. Thompson sampling for contextual bandits with linear payoffs. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. Atlanta, United States of America: JMLR W&CP, 2013. p. 127–135.

AUER, P.; CESA-BIANCHI, N.; FISCHER, P. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, Kluwer Academic Publishers, v. 47, n. 2-3, p. 235–256, 2002. ISSN 0885-6125.

AUER, P.; CESA-BIANCHI, N.; FREUND, Y.; SCHAPIRE, R. E. The nonstochastic multiarmed bandit problem. *SIAM Journal on Computing*, SIAM, v. 32, n. 1, p. 48–77, 2002.

AUSUBEL, L. M.; MILGROM, P. The lovely but lonely vickrey auction. *Combinatorial auctions*, p. 17–40, 2006.

BANK, P.; FÖLLMER, H. American options, multi-armed bandits, and optimal consumption plans: A unifying view. In: *Paris-Princeton Lectures on Mathematical Finance 2002*. Springer Berlin Heidelberg, 2003, (Lecture Notes in Mathematics, v. 1814). p. 1–42. ISBN 978-3-540-40193-3. Disponível em: <http://dx.doi.org/10.1007/978-3-540-44859-4_1>.

BERGEMANN, D.; VÄLIMÄKI, J. *Bandit problems*. New Haven, United States of America, 2006.

BUBECK, S.; CESA-BIANCHI, N.; KAKADE, S. et al. Towards minimax policies for online linear optimization with bandit feedback. In: *Proceedings of the 25th Conference on Learning Theory (COLT)*. [S.l.: s.n.], 2012. p. 1–14.

CHAKRABARTI, D.; KUMAR, R.; RADLINSKI, F.; UPFAL, E. Mortal multi-armed bandits. In: KOLLER, D.; SCHUURMANS, D.; BENGIO, Y.; BOTTOU, L. (Ed.). *Advances in Neural Information Processing Systems*. Vancouver, Canada: Curran Associates, Inc., 2009. p. 273–280.

CHAPELLE, O.; LI, L. Open problem: Regret bounds for thompson sampling. In: COLT. *Proceedings of the 25th Annual Conference on Learning Theory*. Edinburgh, Scotland: JMLR W&CP, 2012. p. 43.1–43.3.

CHATTERJEE, P.; HOFFMAN, D. L.; NOVAK, T. P. Modeling the clickstream: Implications for web-based advertising efforts. *Marketing Science*, INFORMS, v. 22, n. 4, p. 520–541, 2003.

- CHEN, Y.; BERKHIN, P.; ANDERSON, B.; DEVANUR, N. R. Real-time bidding algorithms for performance-based display ad allocation. *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '11*, ACM Press, New York, USA, p. 1307–1315, 2011. Disponível em: <<http://dl.acm.org/citation.cfm?doid=2020408.2020604>>.
- CHICKERING, D. M.; HECKERMAN, D. Targeted advertising with inventory management. In: *Proceedings of the 2nd ACM conference on Electronic commerce*. New York, NY, USA: ACM, 2000. (EC '00), p. 145–149. ISBN 1-58113-272-7. Disponível em: <<http://doi.acm.org/10.1145/352871.352888>>.
- CHU, W.; LI, L.; REYZIN, L.; SCHAPIRE, R. E. Contextual bandits with linear payoff functions. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. Ft. Lauderdale, United States of America: JMLR W&CP, 2011. p. 208–214.
- DANI, V.; HAYES, T. P.; KAKADE, S. M. Stochastic linear optimization under bandit feedback. In: *COLT*. [S.l.: s.n.], 2008. p. 355–366.
- DANTZIG, G. B. *Linear programming and extensions*. Princeton, New Jersey: Princeton University Press, 1998.
- DANTZIG, G. B. Maximization of a linear function of variables subject to linear inequalities. In: COTTLE, R. W. (Ed.). *The Basic George B. Dantzig*. Stanford: Stanford University Press, 2003. p. 24–32.
- DEVANUR, N. R.; KAKADE, S. M. The price of truthfulness for pay-per-click auctions. *Proceedings of the tenth ACM conference on Electronic commerce - EC '09*, ACM Press, New York, New York, USA, p. 99–106, 2009. Disponível em: <<http://portal.acm.org/citation.cfm?doid=1566374.1566388>>.
- GIRGIN, S.; MARY, J.; PREUX, P.; NICOL, O. Advertising campaigns management: Should we be greedy? In: IEEE. *2010 IEEE 10th International Conference on Data Mining (ICDM 2010)*. Sydney, Australia, 2010. p. 821–826.
- GIRGIN, S.; MARY, J.; PREUX, P.; NICOL, O. Planning-based approach for optimizing the display of online advertising campaigns. In: *NIPS workshop on Machine Learning in Online Advertising*. Vancouver, Canada: [s.n.], 2010. p. 20–25.
- GIRGIN, S.; MARY, J.; PREUX, P.; NICOL, O. Managing advertising campaigns - an approximate planning approach. *Frontiers of Computer Science*, SP Higher Education Press, v. 6, n. 2, p. 209–229, 2012. ISSN 2095-2228. Disponível em: <<http://dx.doi.org/10.1007/s11704-012-2873-5>>.
- GITTINS, J. C. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, JSTOR, p. 148–177, 1979.
- HOEFFDING, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, Taylor & Francis Group, v. 58, n. 301, p. 13–30, 1963.

- Internet Advertising Bureau. *Q3 '13 Internet Advertising Revenue Report*. [S.l.], 2013. Disponível em: <http://www.iab.net/insights_research/530422/1357>.
- KAELBLING, L. P. Associative reinforcement learning: Functions ink-dnf. *Machine Learning*, Springer, v. 15, n. 3, p. 279–298, 1994.
- KAUFMANN, E.; CAPPÉ, O.; GARIVIER, A. On bayesian upper confidence bounds for bandit problems. In: *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*. La Palma, Canary Islands: JMLR W&CP, 2012. p. 592–600.
- KAUFMANN, E.; KORDA, N.; MUNOS, R. Thompson sampling: An asymptotically optimal finite-time analysis. In: BSHOUTY, N.; STOLTZ, G.; VAYATIS, N.; ZEUGMANN, T. (Ed.). *Algorithmic Learning Theory*. Lyon, France: Springer Berlin Heidelberg, 2012, (Lecture Notes in Computer Science, v. 7568). p. 199–213. ISBN 978-3-642-34105-2.
- KEASAR, T.; RASHKOVICH, E.; COHEN, D.; SHMIDA, A. Bees in two-armed bandit situations: foraging choices and possible decision mechanisms. *Behavioral Ecology*, Oxford University Press, v. 13, n. 6, p. 757–765, 2002.
- LAI, T. L.; ROBBINS, H. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, Academic Press, v. 6, n. 1, p. 4–22, 1985.
- LANCZOS, C. A precision approximation of the gamma function. *Journal of the Society for Industrial & Applied Mathematics, Series B: Numerical Analysis*, SIAM, v. 1, n. 1, p. 86–96, 1964.
- LANGFORD, J.; ZHANG, T. The epoch-greedy algorithm for multi-armed bandits with side information. In: *Advances in neural information processing systems*. Vancouver, Canada: Curran Associates, Inc., 2007. p. 817–824.
- LANGHEINRICH, M.; NAKAMURA, A.; ABE, N.; KAMBA, T.; KOSEKI, Y. Unintrusive customization techniques for web advertising. *Computer Networks*, Elsevier, v. 31, n. 11, p. 1259–1272, 1999.
- LI, L.; CHU, W.; LANGFORD, J.; SCHAPIRE, R. E. A contextual-bandit approach to personalized news article recommendation. In: ACM. *Proceedings of the 19th international conference on World wide web*. [S.l.], 2010. p. 661–670.
- LI, L.; CHU, W.; LANGFORD, J.; WANG, X. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In: ACM. *Proceedings of the fourth ACM international conference on Web search and data mining*. [S.l.], 2011. p. 297–306.
- MARTÍN, J.; VARGAS, A. M. Linear bayes policy for learning in contextual-bandits. *Expert Systems with Applications*, Elsevier, v. 40, n. 18, p. 7400–7406, 2013.
- MAUSAM; KOLOBOV, A. *Planning with Markov Decision Processes: An AI Perspective*. San Rafael, USA: Morgan & Claypool Publishers, 2012.
- MUTHUKRISHNAN, S. Ad Exchanges : Research Issues. p. 1–12, 2009.

NOVAK, T. P.; HOFFMAN, D. L. New metrics for new media: toward the development of web measurement standards. *World Wide Web Journal*, Faculty of Business Administration, National University of Singapore, v. 2, n. 1, p. 213–246, 1997.

PANDEY, S.; CHAKRABARTI, D.; AGARWAL, D. Multi-armed bandit problems with dependent arms. In: *Proceedings of the 24th International Conference on Machine Learning*. New York, NY, USA: ACM, 2007. (ICML '07), p. 721–728. ISBN 978-1-59593-793-3. Disponível em: <<http://doi.acm.org/10.1145/1273496.1273587>>.

PANDEY, S.; OLSTON, C. Handling advertisements of unknown quality in search advertising. In: *Advances in Neural Information Processing Systems*. Vancouver, Canada: MIT Press, 2006. p. 1065–1072.

POLONIECKI, J. The two-armed bandit and the controlled clinical trial. *The Statistician*, JSTOR, p. 97–102, 1978.

PUTERMAN, M. L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, USA: Wiley-Interscience, 1994. Hardcover. ISBN 0471619779.

SARKAR, J. et al. One-armed bandit problems with covariates. *The Annals of Statistics*, Institute of Mathematical Statistics, v. 19, n. 4, p. 1978–2002, 1991.

SCHRIJVER, A. *Theory of Linear and Integer Programming*. Hoboken, USA: Wiley-Interscience, 1998. ISBN 0471908541.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press, 1998.

TARDOS, E. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, INFORMS, v. 34, n. 2, p. 250–256, 1986.

THOMPSON, W. R. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, JSTOR, v. 25, n. 3/4, p. 285–294, 1933.

TRUZZI, F. S.; SILVA, V. d. Freire da; COSTA, A. H. R.; COZMAN, F. G. AdBandit: A New Algorithm for Multi-Armed Bandits. In: *Encontro Nacional de Inteligência Artificial (ENIA)*. Fortaleza, Brasil: [s.n.], 2013. p. 1–12.

TRUZZI, F. S.; SILVA, V. Freire da; COSTA, A. H. R.; COZMAN, F. G. Markov Decision Processes for Ad Network Optimization. In: *Encontro Nacional de Inteligência Artificial (ENIA)*. Curitiba, Brasil: [s.n.], 2012. p. 1–12.

TRUZZI, F. S.; SILVA, V. Freire da; COSTA, A. R.; COZMAN, F. G. Ad network optimization: Evaluating linear relaxations. In: *Brazilian Conference on Intelligent Systems (BRACIS)*. Fortaleza, Brasil: CPS Publishing, 2013. p. 219–224.

WANG, Y.-G. Sequential allocation in clinical trials. *Communications in Statistics-Theory and Methods*, Taylor & Francis, v. 20, n. 3, p. 791–805, 1991.

WATKINS, C. J.; DAYAN, P. Q-learning. *Machine learning*, Springer, v. 8, n. 3-4, p. 279–292, 1992.

WOODROOFE, M. A one-armed bandit problem with a concomitant variable. *Journal of the American Statistical Association*, Taylor & Francis Group, v. 74, n. 368, p. 799–806, 1979.