Programa de Pós-Graduação em Engenharia Elétrica - PPGEL

Associação ampla UFSJ / CEFET-MG

Análise de Sensibilidade dos Parâmetros do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante

André Luiz Carvalho Ottoni

Orientador: Erivelton Geraldo Nepomuceno

Coorientador: Marcos Santos de Oliveira

São João del-Rei, 25 de novembro de 2016.

Programa de Pós-Graduação em Engenharia Elétrica - PPGEL

Associação ampla UFSJ / CEFET-MG

Análise de Sensibilidade dos Parâmetros do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante

André Luiz Carvalho Ottoni

Dissertação apresentada à banca examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica, associação ampla entre a Universidade Federal de São João del-Rei e o Centro Federal de Educação Tecnológica de Minas Gerais, como parte dos requisitos necessários à obtenção do grau de Mestre em Engenharia Elétrica.

Orientador: Erivelton Geraldo Nepomuceno

Coorientador: Marcos Santos de Oliveira

São João del-Rei, 25 de novembro de 2016.

Ficha catalográfica elaborada pela Divisão de Biblioteca (DIBIB)
e Núcleo de Tecnologia da Informação (NTINF) da UFSJ,
com os dados fornecidos pelo(a) autor(a)

O89a

Ottoni, André Luiz Carvalho .
Análise de Sensibilidade dos Parâmetros do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante / André Luiz Carvalho Ottoni ; orientador Erivelton Geraldo Nepomuceno; coorientador Marcos Santos de Oliveira. -- São João del-Rei, 2016.
68 p.

Dissertação (Mestrado - Mestrado em Engenharia Elétrica) -- Universidade Federal de São João del Rei, 2016.

1. Inteligência Artificial. 2. Aprendizado por Reforço. 3. Problema do Caixeiro Viajante. 4. Análise de Sensibilidade dos Parâmetros. 5. Metodologia de Superfície de Resposta. I. Nepomuceno, Erivelton Geraldo, orient. II. Oliveira, Marcos Santos de , co orient. III. Título.

Dedico este trabalho à família Ottoni, em especial à minha mãe.

Agradecimentos

À Deus.

Um agradecimento especial aos meus pais. Minha mãe Arlinda, por todo amor, dedicação, carinho e orações. Meu pai José Luiz, que sempre está presente no meu coração.

À minha família Ottoni: Gustavo, Marcelo, Heloísa, Ana Clara, Lorenzo e Murilo. Muito obrigado por todo o apoio ao longo desses anos.

Ao meu amor Lara, agradeço por todo o companheirismo, conversas e por estar sempre comigo.

Agradeço à minha Vó Teresinha, tios, tias, primos por todo o carinho.

Agradeço aos professores Erivelton e Marcos, por todos os ensinamentos, orientações, oportunidades e amizade.

Agradeço também à professora Bete da Cia Enlace, por ter me ajudado quando precisei.

Aos meus parceiros de desenvolvimento de artigos durante o mestrado: Rubisson, Heitor, Alípio e Lara.

Aos meus amigos e colegas de São João del-Rei, GCOM e República Tsunami.

Agradeço à Capes, pelo apoio financeiro durante o Mestrado. Agradeço também à UFSJ, CEFET-MG, PPGEL e FAPEMIG.

Muito obrigado à todos!

"As máquinas podem pensar?"

Alan Turing.

Resumo

O Aprendizado por Reforço (AR) é uma técnica de Inteligência Artificial fundamentada nos Processos de Decisão de Markov. Em uma estrutura comum de AR, um agente aprende por meio da experiência de sucessos e fracassos em um ambiente. Além disso, em um sistema de AR é necessária a definição de alguns parâmetros, como taxa de aprendizado (α), fator de desconto (γ) e política $\epsilon - greedy$. Nesse aspecto, a escolha desses parâmetros pode comprometer o desempenho do algoritmo de aprendizado. Dessa forma, pretende-se analisar a sensibilidade dos parâmetros do AR. Para isso, são adotados dois tradicionais algoritmos de AR: Q-learning e SARSA. Optou-se por aplicar o AR no Problema do Caixeiro Viajante (PCV) por se tratar de um problema clássico de otimização combinatória largamente usado na literatura. A metodologia proposta baseia-se em experimentos e métricas estatísticas para analisar a influência desses parâmetros no PCV. A principal técnica de análise abordada é a Metodologia de Superfície de Resposta (RSM). A partir dos modelos de RSM ajustados é possível identificar como o desempenho do AR é influenciado pelos níveis dos parâmetros α e γ. Além disso, a análise de pontos estacionários permite inferir quais os valores dos parâmetros tendem a otimizar a resposta em cada problema. Os resultados computacionais demonstram que a RSM é capaz de produzir melhores soluções para problemas simétricos e assimétricos do PCV.

Palavras-chave: Inteligência Artificial, Aprendizado por Reforço, Problema do Caixeiro Viajante, Análise de Sensibilidade dos Parâmetros, Metodologia de Superfície de Resposta.

Abstract

The Reinforcement Learning (RL) is an Artificial Intelligence technique based on Markov Decision Processes. In an RL common structure an agent learns through experience of successes and failures in an environment. Moreover it is necessary to define some parameters such as learning rate (α), discount factor (γ) and $\epsilon - greedy$. In this respect the choice of these parameters may compromisethe performance of the learning algorithm.Thus it is intended to analyze the sensitivity of the RL parameters. For this, two RL algorithms are adopted: Q-learning and SARSA. This work to implement the RL in the Travelling Salesman Problem (TSP) because it is a classic combinatorial optimization problem widely used in the literature. The proposed methodology is based on experiments and statistical metrics to analyze the influence of these parameters in the TSP. The main approached analysis technique is the Response Surface Methodology (RSM). Thus from the adjusted RSM models can be identified as RL performance is influenced by the levels of parameters α and γ. Furthermore analysis of stationary points which allows inferring the parameter values optimize the response of each problem. Computational results demonstrate that the use of RSM is capable of producing better solutions to both symmetric and asymmetric tests of TSP.

Keywords: Artificial Intelligence, Reinforcement Learning, Traveling Salesman Problem, Parameters Sensitivity Analysis, Response Surface Methodology.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

α: taxa de aprendizado.

α_k: taxa de aprendizado constante.

$\alpha_n(s,a)$: taxa de aprendizado decaindo ao longo do tempo.

γ: fator de desconto.

ϵ: parâmetro da política $\epsilon - greedy$.

λ: traço de elegibilidade.

s_t: estado no instante t.

a_t: ação no instante t.

r_{t+1}: reforço no instante $t + 1$.

S: conjunto de estados possíveis do sistema.

A: conjunto de ações que o agente pode realizar.

T: função de transição de estados.

R: função de recompensa.

$v_n(s,a)$: contador do número de visitas ao par estado-ação.

π: política de decisão para s_t.

a^*: ação mais bem estimada para s_t.

a_a: ação aleatória selecionada com probabilidade ϵ.

Q: matriz de aprendizado.

n_s: número de estados do modelo de aprendizado e número de linhas da matriz Q.

n_a: número de ações do modelo de aprendizado e número de colunas da matriz Q.

$Q_t(s,a)$: valor no instante t na matriz de aprendizado Q para o par estado-ação.

$r(s,a)$: recompensa imediata para a execução da ação a no estado s;

$\max_{a'} Q(s',a')$: valor máximo na matriz Q na linha do novo estado s_{t+1}.

s': novo estado (s_{t+1}).

a': nova ação (a_{t+1}).

ρ: recompensa média para o algoritmo R-learning.

β: taxa de aprendizado para o algoritmo R-learning.

x_i: variável de decisão para problemas modelados com programação inteira.

C: conjunto de cidades para o Problema do Caixeiro Viajante.

c_n: cidade n.

c_{ij}: custo de deslocamento entre duas cidades i e j.

x_{ij} variável de decisão para o Problema do Caixeiro Viajante.

d_{ij}: distâncias entre as cidades i e j.

r_{ij}: reforço recebido por partir de i para j.

D_n: matriz de distâncias para o problema n.

R_n: matriz de reforços para o problema n.

y: variável resposta de um modelo de Superfície de Resposta.

β_0: intercpeto de um modelo de Superfície de Resposta.

β_n: coeficiente de um modelo de Superfície de Resposta.

e: erro (resíduo) de um modelo de Superfície de Resposta.

$\hat{y}$: resposta predita de um modelo de Superfície de Resposta.

H_0: hipótese nula para o teste de normalidade de Kolmogorov-Smirnov.

H_1: hipótese alternativa para o teste de normalidade de Kolmogorov-Smirnov.

p_{ks}: p-valor para o teste de normalidade de Kolmogorov-Smirnov.

R^2: coeficiente de determinação múltipla.

R_a^2: coeficiente de determinação múltipla ajustado.

p: p-valor para o teste de significância dos coeficientes de Superfície de Resposta.

b: representação matricial dos coeficientes lineares Superfície de Resposta.

B: representação matricial dos coeficientes de segunda ordem de Superfície de Resposta.

x_0: ponto estacionário.

Lista de Abreviações

AG: Algoritmos Genéticos.

AR: Aprendizado por Reforço.

IA: Inteligência Artificial.

MDP: Processos de Decisão de Markov (*Markov Decision Processes*).

PCV: Problema do Caixeiro Viajante.

PKS: Problema dos K-Servos.

PRV: Problema de Roteamento de Veículos.

RSM: Metodologia de Superfície de Resposta (*Response Surface Methodology*).

SI: Sistemas Inteligentes.

TSP: *Symmetric Traveling Salesman Problem*.

ATSP: *Asymmetric Traveling Salesman Problem*.

VI: Variável Independente.

KS: Teste de Kolmogorov-Smirnov.

Sumário

CAPÍTULO 1

Introdução

Os Sistemas Inteligentes (SI) estão cada vez mais presentes na resolução de problemas importantes na sociedade (Russell e Norving, 2013; Rezende, 2003). Robótica (Romero et al., 2014), carros autônomos (Ozguner et al., 2007), aplicativos de internet (Atzori et al., 2010), jogos (Mnih et al., 2015) e indústria (Lee et al., 2015) são apenas algumas das possíveis aplicações para os SI. Dessa forma, novas pesquisas e o aprimoramento de técnicas de Inteligência Artificial (IA) se tornam fatores importantes para garantir o contínuo avanço tecnológico em diversas áreas.

Uma importante característica que define os SI é a capacidade de aprender. De acordo com Russell e Norving (2013), o aprendizado permite a um agente operar em ambiente inicialmente desconhecido e se tornar mais competente do que seu conhecimento inicial permitiria. Assim, o Aprendizado de Máquina é um campo da IA que busca desenvolver algoritmos capazes de melhorar o desempenho de um sistema por meio da experiência na tarefa (Mitchell, 1997; Russell e Norving, 2013). De acordo com Russell e Norving (2013), o Aprendizado de Máquina pode ser classificado em: Aprendizado Supervisionado, Aprendizado Não-Supervisionado e Aprendizado por Reforço.

O Aprendizado por Reforço (AR) é uma técnica baseada no aprendizado pelo sucesso e fracasso, e fundamentada nos Processos de Decisão de Markov (Barto et al., 1981; Sutton, 1984, 1988; Watkins, 1989; Watkins e Dayan, 1992; Littman, 1994; Kaelbling et al., 1996; Sutton e Barto, 1998; Ribeiro, 2002; Wiering e van Otterlo, 2012). Em uma estrutura comum de AR, o aprendizado acontece a partir da interação direta de um agente com o ambiente. Assim, no AR o agente usa sensores para identificar o estado (s) atual do ambiente, em seguida executa a

melhor a ação (a), e então recebe um retorno para o par estado-ação (s,a). Geralmente, reforços positivos indicam sucesso na tomada de decisão. Já as reforços negativos são as penalidades. Dessa forma, o agente armazena essas informações de sucesso e fracasso para auxiliar nas tomadas de decisões futuras.

Por exemplo, em um ambiente de futebol de robôs, as ações devem ser recompensadas visando aumentar a chance de vitória do time, como chutar a bola para o gol. Já as ações que levem a equipe a derrota devem ser penalizadas, como tocar a bola para o adversário ou marcar gol contra (Ottoni et al., 2015b). Outro exemplo, é quando um agente deve aprender a traçar a menor trajetória entre dois pontos em um labirinto. Nesse caso, o fracasso do agente aprendiz cresce com o aumento da distância percorrida. Assim, quanto menor o caminho realizado, maior será a recompensa ao finalizar a trajetória (Ottoni et al., 2016d).

O Aprendizado por Reforço possui aplicações diversas na literatura como, robótica (Selvatici e Costa, 2007; Kober et al., 2013; Kormushev et al., 2013), sistemas multiagentes (Stone et al., 2005; Bazzan, 2009; Ottoni et al., 2012; Bianchi et al., 2014) e otimização (Gambardella e Dorigo, 1995; Mariano e Morales, 2000; Berlink et al., 2015). Nesse sentido, o AR também vem sendo aplicado na resolução de um problema clássico de otimização combinatória, o Problema do Caixeiro Viajante (PCV) (Gambardella e Dorigo, 1995; Liu e Zeng, 2009; Bianchi et al., 2009; Lima Júnior et al., 2010; Santos et al., 2014; Ottoni et al., 2015a; Alipour e Razavi, 2015), apresentado em seguida.

1.1 Domínio estudado: Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante (PCV), em inglês *Traveling Salesman Problem*, é frequentemente abordado na literatura, e assim diversos métodos já foram aplicados na sua solução, como Algoritmo de Colônia de Formigas (Dorigo e Gambardella, 1997), Algoritmos Meméticos (Buriol et al., 2004), Algoritmos Genéticos (Deng et al., 2015), Aprendizado por Reforço (Gambardella e Dorigo, 1995), Busca Tabu (Fiechter, 1994), Enxame de Partículas (Marinakis e Marinaki, 2010), Redes Neurais (Siqueira et al., 2007), *Simulated Annealing* (Chen e Chien, 2011), entre outros (Applegate et al., 2007; Cook, 2011; Marinakis et al., 2011; Ouaarab et al., 2014). Seu objetivo é definir a menor rota entre um conjunto de n cidades. Assim, o caixeiro deve passar por todas as localidades uma única vez, exceto aquela na qual se inicia e termina a

jornada.

Geralmente, o PCV é formulado sobre um grafo $G = (N,A)$, em que, N é conjunto de nós (vértices), e A é o conjunto de arcos (i,j) do problema (Goldbarg e Luna, 2005). Além disso, o PCV pode ser abordado sobre dois paradigmas: simétrico e assimétrico. No caso simétrico, o custo associado ao deslocamento de uma cidade i para uma localidade j é equivalente ao custo de ir de j para i. Já no problema assimétrico, o sentido de realização da rota pode alterar o valor da distância total percorrida.

Uma das principais aplicações do PCV é no planejamento de rotas de veículos (Reinelt, 1994; Applegate et al., 2007). Nesse aspecto, encontrar um melhor caminho para uma rota pode representar, por exemplo, uma grande economia para uma transportadora, em termos de gastos com combustível e menor tempo de entrega de um produto. Além disso, o PCV possui outras aplicações práticas, como na confecção de placas de circuito impresso, mapeamento de genoma, sequenciamento de DNA e problemas diversos de sequenciamento de tarefas (Reinelt, 1994; Applegate et al., 2007).

Na próxima seção são levantados trabalhos que abordaram o AR na resolução de problemas de otimização combinatória, como o PCV.

1.2 Aprendizado por Reforço em Problemas de Otimização Combinatória

Gambardella e Dorigo (1995) realizam uma conexão entre a técnica de otimização de Colônia de Formigas, em inglês *Ant System (AS)*, e o Aprendizado por Reforço. Dessa forma, é introduzida a classe de algoritmos Ant-Q. Além disso, o Ant-Q é aplicado na solução do Problema do Caixeiro Simétrico e Assimétrico. Em Chang (2004), o aprendizado acontece adotando um conjunto de formigas que seguem políticas de exploração distintas. Já os trabalhos de Mariano e Morales (1999, 2000) introduzem o algoritmo Ant-Q multiobjetivo, denominado MOAQ.

Uma outra abordagem recorrente na solução de problemas de otimização combinatória é o desenvolvimento de soluções híbridas entre Algoritmos Genéticos (AG) e Aprendizado por Reforço (Miagkikh e Punch, 1999a; Liu e Zeng, 2009; Santos et al., 2009; Lima Júnior et al., 2010). Em Miagkikh e Punch (1999b,a), os autores apresentam resultados para o Problema do Caixeiro Viajante Assimétrico. Já os trabalhos de Lima Júnior (2009); Lima Júnior et al.

(2010), propõem a adoção do algoritmo Q-learning como estratégia de exploração/explotação para as metaheurísticas GRASP e AG. De forma semelhante, as pesquisas de Santos et al. (2009) também abordam resoluções com AR, GRASP e AGs, porém adotando uma implementação paralela. O artigo de Liu e Zeng (2009), por sua vez, propõe o algoritmo RMGA, que também integra Algoritmos Genéticos e Aprendizado por Reforço.

Seguindo a mesma linha de conciliar técnicas para a resolução do PCV, o trabalho de Bianchi et al. (2009) aplica a aceleração por heurísticas no AR. Já em Santos et al. (2014), são integrados os algoritmos de busca local VNS e Aprendizado por Reforço na solução do Problema do Caixeiro Viajante e Problema do Escalonamento de Sondas para Manutenção de Poços. Já o trabalho de Dries e Peterson (2008), reúne conceitos do Aprendizado por Reforço Hierárquico (*Hierarchical Reinforcement Learning - HRL*) com a técnica de *Ant Colony Optimization*, criando o método HRLACO.

Outras abordagens propostas para resolver o PCV por AR são: cooperativa (Qi e Sun, 2005) e multiagente (Alipour e Razavi, 2015). Nessa linha, Sun et al. (2001) propõe dois métodos baseados na *Ant Colony System* para o Aprendizado por Reforço multiagente.

Em Ottoni et al. (2015a), são realizados experimentos para analisar a aplicação do AR na solução do PCV, de acordo com a definição dos parâmetros taxa de aprendizado (α) e política $\epsilon - greedy$. Os resultados de Ottoni et al. (2015a), apontam que a seleção dos valores de α e $\epsilon - greedy$ podem comprometer o desempenho do AR na resolução do PCV. De fato, estudos já demonstraram que o desempenho do AR pode ser influenciado pela definição de parâmetros, como taxa de aprendizado, fator de desconto (γ) e $\epsilon - greedy$ (Sutton e Barto, 1998; Schweighofer e Doya, 2003; Even-Dar e Mansour, 2003; Tokic et al., 2013; Gatti, 2015). Nessa perspectiva, na próxima seção são apresentadas pesquisas que analisaram os efeitos da definição de parâmetros no AR.

1.3 Análise e Definição de Parâmetros no Aprendizado por Reforço

Even-Dar e Mansour (2003) mostraram que a convergência do Q-learning é sensível aos valores de α e γ. Já o trabalho de Schweighofer e Doya (2003), introduz o conceito de meta-parâmetros para o AR. Dessa forma, em Schweighofer e Doya (2003) é proposto um algoritmo para o ajuste de parâmetros do AR de forma dinâmica. Murakoshi e Mizuno (2004), por sua vez,

estendem os resultados de Schweighofer e Doya (2003), implementando um método de controle de parâmetros que responde mais rapidamente às mudanças inesperadas no ambiente. Já Gosavi (2008) apresenta um estudo empírico sobre o efeito da taxa de aprendizado na convergência de algoritmos de AR.

No trabalho de Cline (2004), algoritmos genéticos são adotados para a definição de parâmetros do Q-learning em ambiente simulado de cooperação de agentes. Já os autores de Kobayashi et al. (2009), propõem um método de ajuste dos parâmetros baseado no erro de diferença temporal. Yoshida et al. (2013), por sua vez, apresentam um *framework* para otimizar o fator de desconto, adotando algoritmos evolucionários. O método se baseia em função de adaptação de γ de acordo com estado corrente do ambiente. Já Tokic et al. (2013), investigam a definição dos parâmetros adotando o método "*REINFORCE explotation control*", e observam que a definição da constante de traços de elegibilidade (λ) está intimamente ligada ao valor da taxa de aprendizado.

Em outras pesquisas são realizadas análises para verificar os efeitos dos parâmetros nas aplicações do AR em estudo. Leng e Lim (2011) e Lamperti et al. (2013b) analisaram os efeitos de parâmetros do AR em simulação de futebol de robôs. Já em Prauchner et al. (2014), os parâmetros fator de desconto, taxa de aprendizado e traço de elegibilidade, são analisados em um ambiente de navegação robótica de coleta de lixo. Em uma aplicação semelhante, o trabalho Bal e Mahalik (2014), define os parâmetros α e γ por tentativa e erro para um ambiente de navegação simulada. Também no campo da robótica, os autores de Khamassi et al. (2011) verificam a sensibilidade da aplicação à taxa de aprendizado. Já o artigo de Berlink e Costa (2013), avalia a sensibilidade do Q-learning à variação de γ em um problema de otimização de venda de energia. Leite (2014), por sua vez, realiza experimentos para avaliar os efeitos do fator de desconto e da política $\epsilon - greedy$ na aplicação do AR em sistemas de telecomunicações. Já Medeiros (2014), compara as políticas de exploração $\epsilon - greedy$ e softmax. Além disso, avalia o efeito de oito combinações dos parâmetros α, γ e ϵ para o ambiente *Chain Domain*.

Outros trabalhos implementam algoritmos para taxas de aprendizados adaptativas em ambientes dinâmicos (Noda, 2010; Dabney, 2014; Ryzhov et al., 2015).

No entanto, não foram encontrados na literatura estudos que modelem matematicamente a sensibilidade dos parâmetros do AR na solução do PCV. Assim, torna-se um fator relevante a adoção de uma metodologia estatística para analisar os efeitos dos parâmetros do AR aplicado

ao PCV (Ottoni et al., 2015a). Uma alternativa é a adoção da Metodologia de Superfície de Resposta (RSM) (Myers et al., 2009). A RSM é uma técnica estatística aplicada nos estudos de processos de otimização (Myers et al., 2009). Trabalhos recentes abordaram a Metodologia de Superfície de Reposta em conjunto com técnicas inteligentes, como Redes Neurais (Gonçalves Júnior et al., 2014) e Algoritmos Genéticos (Mendes et al., 2014). Já em Gatti (2015), a RSM é adotada na análise da influência de parâmetros do AR na convergência do algoritmo $TD(\lambda)$ em dois problemas: *Mountain Car Problem* e *Truck Backer-upper Problem*.

1.4 Objetivos

Este trabalho visa analisar o desempenho do Aprendizado por Reforço na solução do Problema do Caixeiro Viajante (PCV). Para isso, serão adotados os algoritmos de AR mais tradicionais na literatura: Q-learning (Watkins e Dayan, 1992) e SARSA (Sutton e Barto, 1998). Pretende-se analisar a sensibilidade da definição de parâmetros do Aprendizado por Reforço nesse tipo de problema de otimização combinatória. Assim, o objetivo é desafiar os algoritmos de AR na solução do PCV, verificando como seus desempenhos são afetados pelos valores de taxa de aprendizado, fator de desconto e política $\epsilon - greedy$. Além disso, pretende-se analisar a influência dos parâmetros no PCV a partir de uma metodologia baseada em experimentos e métricas estatísticas.

1.5 Contribuições

Em seguida, são apresentados os trabalhos publicados, frutos dos estudos ao longo do desenvolvimento desta dissertação:

- Ottoni, A. L. C., Nepomuceno, E. G., Cordeiro, L. T., Lamperti, R. D. e Oliveira, M. S. (2015). Análise do Desempenho do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante. Anais do XII SBAI - Simpósio Brasileiro de Automação Inteligente 2015 (Ottoni et al., 2015a).

- Ottoni, A. L. C., Nepomuceno, E. G., Oliveira, M. S., Cordeiro, L. T. e Lamperti, R. D. Análise da influência da taxa de aprendizado e do fator de desconto sobre o desempenho dos algoritmos Q-learning e SARSA: aplicação do aprendizado por reforço na navegação

autônoma. Revista Brasileira de Computação Aplicada, 8(2), 44–59. http://doi.org/10.5335/rbca.v8i2.5249 (Ottoni et al., 2016d).

- Ottoni, A. L. C., Nepomuceno, E. G. e Oliveira, M. S. Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante Assimétrico: Uma comparação entre os Algoritmos Q-learning e SARSA. Anais do XII SIMMEC - Simpósio de Mecânica Computacional 2016 (Ottoni et al., 2016c).

- Ottoni, A. L. C., Nepomuceno, E. G. e Oliveira, M. S. Análise de Sensibilidade dos Parâmetros do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante: Modelagem via Superfície de Resposta. Anais do XXI CBA - Congresso Brasileiro de Automática 2016 (Ottoni et al., 2016b).

- Ottoni, A. L. C., Nepomuceno, E. G., Oliveira, M. S. e Felix, L. B. Modelo Híbrido de Aprendizado por Reforço e Lógica Fuzzy Aplicado ao Problema do Caixeiro Viajante com Reabastecimento. Anais do XXI CBA - Congresso Brasileiro de Automática 2016 (Ottoni et al., 2016e).

- Ottoni, A. L. C., Barbosa, A. M., Nepomuceno, E. G. e Oliveira, M. S. Controle de Epidemias com Aprendizado por Reforço: Estratégia de Combate ao *Aedes aegypti*. Anais do XIX ENMC - Encontro Nacional de Modelagem Computacional 2016 (Ottoni et al., 2016a).

Em Ottoni et al. (2015a), o desempenho do Q-learning é analisado na resolução de três instâncias simétricas do PCV: berlin52, brazil58 e kroA100. Para isso, são realizados experimentos adotando três valores para política $\epsilon - greedy$ e dois paradigmas para a taxa de aprendizado (constante e decaindo). Nesse mesmo aspecto, (Ottoni et al., 2016c) analisam os efeitos dos parâmetros α e γ em instâncias assimétricas do PCV.

Já em Ottoni et al. (2016d), é analisada a influência da taxa de aprendizado e fator de desconto sobre o desempenho dos algoritmos Q-learning e SARSA. Como estudo de caso, é abordado um ambiente simples de navegação simulada em *gridworld*.

Ottoni et al. (2016b), por sua vez, apresentam a modelagem matemática de análise de sensibilidade dos parâmetros dos parâmetros α e γ, com a adoção da Metodologia de Superfície de Resposta.

Já Ottoni et al. (2016e), apresenta um modelo híbrido de AR e Lógica Fuzzy, aplicado a uma variação do PCV, Problema do Caixeiro Viajante com Reabastecimento.

Finalmente, Ottoni et al. (2016a) aplicam o Aprendizado por Reforço no controle de epidemias. Mais especificamente é apresentada uma estratégia de combate ao *Aedes aegypti*, através do aprendizado de rotas de um veículo de saúde pública.

Além disso, vale ressaltar que, em anos anteriores o autor deste trabalho realizou outras publicações envolvendo o Aprendizado por Reforço, como: (Ottoni et al., 2012; Ottoni e Oliveira, 2014; Ottoni et al., 2015b).

1.6 Organização do trabalho

Este trabalho está organizado em seis capítulos. No Capítulo 2, são apresentados os fundamentos teóricos do Aprendizado por Reforço. Já o Capítulo 3, descreve o Problema do Caixeiro Viajante. Em seguida, os procedimentos metodológicos são levantados no Capítulo 4. O Capítulo 5, por sua vez, apresenta os resultados e devidas análises para os experimentos do PCV Simétrico e Assimétrico. Finalmente, no Capítulo 6 são descritas as conclusões desta dissertação e propostas de trabalhos futuros.

CAPÍTULO 2

Aprendizado por Reforço

Neste capítulo, são apresentados conceitos que fundamentam a técnica de Aprendizado por Reforço (AR). Inicialmente, são descritos os tipos de Aprendizado de Máquina e a teoria dos Processos de Decisão de Markov. Em seguida, são levantados aspectos da estrutura e algoritmos do AR. Por fim, são apresentadas algumas aplicações do Aprendizado por Reforço.

2.1 Aprendizado de Máquina

O Aprendizado de Máquina é um importante campo da Inteligência Artificial (IA) (Russell e Norving, 2013). Os sistemas inteligentes dotados com algoritmos de aprendizado conseguem melhorar seu desempenho em uma tarefa por meio da experiência (Mitchell, 1997).

O Aprendizado de Máquina é multidisciplinar e reuni conceitos tanto da IA quanto de diversas áreas, como: probabilidade e estatística, computação complexa, teoria da informação, filosofia, psicologia e teoria de controle (Mitchell, 1997).

De acordo com (Russell e Norving, 2013), o campo do Aprendizado de Máquina pode ser classificado em três casos:

- Aprendizado Supervisionado: essa classe de métodos envolve aprender uma função a partir de exemplos de entradas e saídas (Russell e Norving, 2013). A tabela de dados (entradas/saídas) se comporta com um "professor"ensinando para o sistema inteligente qual a resposta para cada valor de entrada (Silva et al., 2010).

- Aprendizado Não-Supervisionado: esse tipo de aprendizado não dispõe das saídas desejadas. Assim, o próprio sistema se auto organiza identificando subconjuntos que contenham

similaridades nas amostras (Silva et al., 2010).

- Aprendizado por Reforço: nessa categoria o agente inteligente aprende a partir de sinais de reforços (Sutton e Barto, 1998; Wiering e van Otterlo, 2012). Essas recompensas indicam o sucesso e fracasso na tomada de decisão. Na sequência deste capítulo, o Aprendizado por Reforço será tratado com mais detalhes.

2.2 Processos de Decisão de Markov

A teoria dos Processos de Decisão de Markov, em inglês *Markov Decision Processes (MDP)* (White, 1993; Puterman, 1994; Littman et al., 1995) é a base da formulação do Aprendizado por Reforço (Mitchell, 1997; Sutton e Barto, 1998; Bianchi, 2004). Um MDP modela processos em que as transições entre os estados do sistema são probabilísticas (Pellegrini e Wainer, 2007). Além disso, são chamados de processos de decisão, pois modelam a possibilidade de um agente interferir no sistema executando ações (Pellegrini e Wainer, 2007). Outro ponto chave é que um MDP deve obedecer a Hipótese de Markov (Sutton e Barto, 1998). A Hipótese de Markov estabelece que o próximo estado (s_{t+1}) do sistema depende apenas do estado atual (s_t) e da ação (a_t) tomada pelo agente nesse instante (Sutton e Barto, 1998; Bianchi, 2004).

Em um MDP, uma regra de decisão é o mapeamento de estados em ações. Assim, uma política π é o conjunto de todas as regras de decisão (Pellegrini e Wainer, 2007). Dessa forma, a política que otimiza a recompensa recebida ao longo tempo resolve o MDP (Bianchi, 2004).

Os Processos de Decisão de Markov são definidos pela quádrupla (S, A, T, R) (Puterman, 1994; Mitchell, 1997; Bianchi, 2004; Pellegrini e Wainer, 2007), em que:

- S é o conjunto finito de estados possíveis do sistema;
- A é o conjunto finito de ações que o agente pode realizar;
- T é a função de transição de estados. Estabelece qual a probabilidade do sistema passar do estado s_t para o estado s_{t+1}, quando o agente realiza a ação a_t.
- R é a função que mapeia a recompensa por se tomar a ação a no estado s.

2.3 Estrutura do Aprendizado por Reforço

O AR consiste em aprender mapeando ações para situações (estados), com o objetivo de maximizar ao longo tempo o valor de recompensa. O aprendiz é denominado agente, que toma decisões e interage com o ambiente (Sutton e Barto, 1998). A interação é contínua no ambiente por meio das ações, que dão origem as novas situações e as recompensas para o agente.

Mais especificamente, agente e ambiente interagem em uma sequência de passos de tempo discretos ($t = 0, 1, 2, 3...$). A cada instante de tempo t, o agente recebe uma representação do ambiente, por meio do estado, $s_t \in S$, e seleciona uma ação $a_t \in A$. Um instante de tempo depois, $t + 1$, o agente recebe um reforço, $r_{t+1} \in R$ e observa o novo estado s_{t+1}. Sendo que, S é o conjunto de todos os estados, A é o conjunto das ações e R é a função de reforço (Sutton e Barto, 1998).

Uma representação da interação agente-ambiente é mostrada na Figura 2.1.

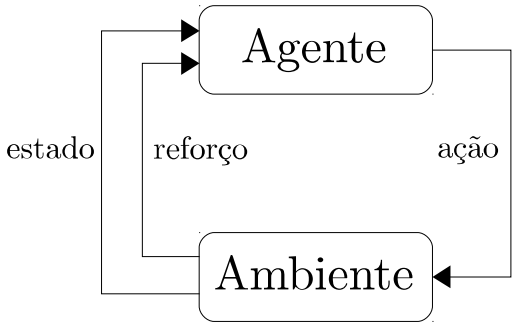


Figura 2.1: Representação da interação do agente com o ambiente no Aprendizado por Reforço. Modificado de Sutton e Barto (1998).

Em seguida, serão levantados outros elementos relativos a estrutura do Aprendizado por Reforço. Posteriormente, os principais parâmetros do AR serão apresentados: taxa de aprendizado (α), fator de desconto (γ) e o parâmetro ϵ da política de seleção de ações ϵ-*greedy*. Além disso, no decorrer do capítulo alguns algoritmos do AR serão apresentados.

2.3.1 Elementos do Aprendizado por Reforço

De acordo com (Sutton e Barto, 1998), além do agente e do ambiente, existem quatros elementos que envolvem o AR:

- Política: define o comportamento direto de estados e ações. Assim, uma política é responsável por determinar a tomada de decisão para cada situação do ambiente.

- Função de Recompensa: a função de recompensa define quais são os eventos bons e ruins para o agente. Dessa forma, para cada par estado e ação (s,a), é estabelecida um valor de recompensa.

- Função de Valor: enquanto uma função de recompensa indica o que é bom em uma resposta imediata, uma função de valor especifica o que é importante ao longo prazo. Ou seja, o valor de um estado é a quantidade total de recompensa que o agente pode esperar acumular para esse estado.

- Modelo do Ambiente: define o comportamento do ambiente.

2.3.2 Parâmetros no Aprendizado por Reforço

Taxa de Aprendizado

Nos algoritmos Q-learning e SARSA, a taxa de aprendizado pode ser definida em qualquer valor no intervalo entre 0 e 1, $0 < \alpha \leq 1$ (Sutton e Barto, 1998).

Uma condição de convergência do Q-learning (Mitchell, 1997), estabelece que cada par estado-ação (s, a) deve ser visitado infinitas vezes, com α atendendo as Equações 2.1 e 2.2:

$$\sum_{n=1}^{\infty} \alpha_n(s,a) = \infty, \tag{2.1}$$

$$\sum_{n=1}^{\infty} [\alpha_n(s,a)]^2 < \infty. \tag{2.2}$$

Uma maneira de satisfazer essas duas condições, é decaindo o valor da taxa de aprendizado de acordo com o número de acessos a cada par (s, a). Para isso, (Mitchell, 1997) estabeleceu a Equação 2.3:

$$\alpha_n(s,a) = \frac{1}{1 + v_n(s,a)}, \tag{2.3}$$

em que, $v_n(s,a)$ é o número de visitas ao par estado-ação (s, a).

No entanto, o método mais comum é a definição da taxa de aprendizado com um valor constante para todo instante de tempo t, ou seja, $\alpha_t = \alpha \in (0,1), \forall t \geq 0$ (Beck e Srikant, 2012; Dabney, 2014). Nesse caso, a Equação 2.2 não é satisfeita. Dessa forma, o sistema pode nunca completar a convergência, pois continua variando em resposta às recompensas recentes (Sutton e Barto, 1998).

Fator de Desconto

O fator de desconto (γ) permite ao agente selecionar as ações na tentativa de maximizar a soma de recompensas no futuro. O retorno R_t, na Equação 2.4, representa o somatório de recompensas descontadas no tempo:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \tag{2.4}$$

em que, $\gamma \in [0,1]$ (Sutton e Barto, 1998).

Assim, o fator de desconto controla o grau de influência de recompensas futuras no instante t sobre a recompensa imediata (r_{t+1}) (Sutton e Barto, 1998):

- $\gamma = 1$: as recompensas futuras são fortemente consideradas;
- $\gamma < 1$: a influência do somatório das recompensas futuras é limitado;
- $\gamma = 0$: maximiza a recompensa imeditada, ou seja, o objetivo é simplesmente aprender uma ação (a_t) em um estado (s_t) para maximizar somente r_{t+1}.

Política de Seleção de Ações $\epsilon - greedy$

Uma importante definição do Aprendizado por Reforço é a política de seleção de ações. Esse método é responsável por definir a tomada de decisão em cada instante de tempo. Uma opção é a adoção de uma política gulosa (*greedy*), que sempre seleciona a ação mais bem estimada até o momento para o estado corrente. No entanto, esse método pode facilmente encontrar uma solução local e ficar preso nesse ponto indefinidamente. Uma alternativa para esse problema é selecionar ações de forma aleatória com pequena probabilidade ϵ. Esse método é denominado $\epsilon - greedy$ (ou, quase-guloso) (Sutton e Barto, 1998). O parâmetro ϵ é responsável pelo controle entre gula (exploitação) e aleatoriedade (exploração) na seleção das ações. Em outras palavras,

com a política $\epsilon - greedy$ o agente de aprendizado altera entre exploitação (*exploitation*) e exploração (*exploration*) de novas ações/estados no ambiente (Sutton e Barto, 1998).

A regra de seleção de ações da política $\epsilon - greedy$ é dada na Equação 2.5:

$$\pi(s) = \begin{cases} a^*, & \text{com probabilidade} \quad 1-\epsilon \\ a_a, & \text{com probabilidade} \quad \epsilon, \end{cases} \tag{2.5}$$

em que:

- $\pi(s)$ é a política de decisão para o estado corrente s;
- a^* é a ação mais bem estimada para o estado s no instante atual;
- a_a é uma ação aleatória selecionada com probabilidade ϵ.

2.4 Algoritmos de Aprendizado por Reforço

Os algoritmos de resolução de Aprendizado por Reforço geralmente envolvem conceitos de métodos de Monte Carlo (MC), Programação Dinâmica (PD) ou Diferença Temporal (DT). O Aprendizado por Reforço baseado em DT não necessita de um modelo dinâmico do ambiente, assim como os métodos de Monte Carlo. Por outro lado, os métodos de Diferença Temporal se assemelham à PD, pois utilizam as estimativas já aprendidas para atualizar as novas estimativas (Sutton e Barto, 1998). Os algoritmos de AR descritos nesta seção são baseados na Diferença Temporal: Q-learning (Watkins, 1989; Watkins e Dayan, 1992), SARSA (Sutton e Barto, 1998) e R-learning (Schwartz, 1993).

2.4.1 Q-learning

O Q-learning proposto por (Watkins, 1989; Watkins e Dayan, 1992) é um dos algoritmos de Aprendizado por Reforço mais conhecidos e aplicados. O método se baseia na atualização da matriz de aprendizado Q, a partir da Equação 2.6.

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r(s,a) + \gamma max_{a'} Q(s',a') - Q_t(s,a)], \tag{2.6}$$

em que:

- $Q_t(s,a)$ é valor no instante t na matriz de aprendizado Q para o par estado (s) $\times$ ação (a);

- $Q_{t+1}(s,a)$ é a atualização da matriz de aprendizado no instante $t+1$ pela execução da ação a no estado s;
- $r(s,a)$ é o recompensa imediata para a execução da ação a no estado s;
- $\max_{a'} Q(s',a')$ é a utilidade de s', ou seja, o valor máximo na matriz de aprendizado na linha do novo estado s'.
- α é a taxa de aprendizado;
- γ é o fator de desconto;
- $s = s_t$, $a = a_t$, $s' = s_{t+1}$ e $a' = a_{t+1}$.

A matriz Q possui dimensões $n_s \times n_a$, em que n_s é o número de estados do modelo e n_a é o número de ações que o agente pode executar. Assim, para cada tomada de decisão (ação) em um determinado estado do ambiente, é retornado ao agente um sinal $r(s,a)$. Dessa forma, em casa instante de tempo t, a matriz é atualizada em um único par estado $\times$ ação.

Para a atualização de Q também é necessário identificar o novo estado do ambiente (s' ou s_{t+1}). Já que, o termo $\max_{a'} Q(s',a')$ é o máximo valor na linha relativa ao novo estado s' na matriz.

O Algoritmo 1 retrata o Q-learning.

Algoritmo 1: Q-learning.

Definir os parâmetros: α, γ e ϵ

Para cada par ***s,a*** inicialize a matriz ***Q(s,a)***=0

Observe o estado ***s***

repita

Selecione a ação ***a*** usando a política $\epsilon - greedy$

Execute a ação ***a***

Receba a recompensa imediata ***r(s, a)***

Observe o novo estado ***s'***

$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r(s,a) + \gamma max_{a'} Q(s',a') - Q_t(s,a)]$

s = s'

até *o critério de parada ser satisfeito*;

2.4.2 SARSA

O algoritmo SARSA (Sutton e Barto, 1998) é uma modificação do Q-learning. O SARSA não adota a maximização das ações do Q-learning, assim a matriz de aprendizado é atualizada como na Equação 2.7:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r(s,a) + \gamma Q_t(s',a') - Q_t(s,a)]. \tag{2.7}$$

O algoritmo SARSA recebeu esse nome pois envolve na sua atualização cinco termos:

- s_t: estado no instante t;
- a_t: ação executada no instante t;
- $r(s,a)$: retorno no instante t para o par $s_t \times a_t$;
- s_{t+1}: estado no instante $t+1$;
- a_{t+1}: ação executada no instante $t+1$.

O SARSA é retratado no Algoritmo 2.

Algoritmo 2: SARSA.

Definir os parâmetros: α, γ e ϵ

Para cada par ***s,a*** inicialize a matriz ***Q(s,a)***=0

Observe o estado ***s***

Selecione a ação ***a*** usando a política ε-gulosa

repita

  Execute a ação ***a***

  Receba a recompensa imediata ***r(s,a)***

  Observe o novo estado ***s'***

  Selecione a nova ação ***a'*** usando a política $\epsilon - greedy$

  $Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r(s,a) + \gamma Q_t(s',a') - Q_t(s,a)]$

  s = s'

  a = a'

até *o critério de parada ser satisfeito*;

2.4.3 R-learning

O algoritmo R-learning (Schwartz, 1993) busca maximizar a recompensa média a cada instante de tempo. A matriz de aprendizado é atualizada de acordo com a Equação 2.8:

$$Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r(s,a) - \rho + max_{a'}Q(s',a') - Q_t(s,a)]. \quad (2.8)$$

A recompensa média ρ é calculada como na Equação 2.9. Vale ressaltar que, ρ somente é atualizada quando $Q(s,a) = max_{a'}Q(s',a')$.

$$\rho = \rho + \beta[r(s,a) - \rho + max_{a'}Q(s',a') - max_aQ(s,a)], \quad (2.9)$$

em que β é uma taxa de aprendizado.

O R-learning é retratado no Algoritmo 3.

Algoritmo 3: R-learning.

Definir os parâmetros: α, β, γ e ϵ

Inicializar ρ

Para cada par ***s,a*** inicialize a matriz ***Q(s,a)***=0

repita

- Observe o estado ***s***
- Selecione a ação ***a*** usando a política $\epsilon - greedy$
- Execute a ação ***a***
- Receba a recompensa imediata ***r(s,a)***
- Observe o novo estado ***s'***
- Selecione a nova ação ***a'*** usando a política $\epsilon - greedy$
- $Q_{t+1}(s,a) = Q_t(s,a) + \alpha[r(s,a) - \rho + max_{a'}Q(s',a') - Q_t(s,a)]$
- **se** $Q(s,a) = max_{a'}Q(s',a')$ **então**
  - $\rho = \rho + \beta[r(s,a) - \rho + max_{a'}Q(s',a') - max_aQ(s,a)]$
- **fim**
- ***s = s'***

até *o critério de parada ser satisfeito*;

2.5 Aplicações do Aprendizado por Reforço

Ao longo dos últimos anos, diversas foram as aplicações para o Aprendizado por Reforço. Nesse aspecto, em seguida são levantados alguns trabalhos que tiveram como atuação primordial a aplicação do AR.

- Robótica: os trabalhos (Faria e Romero, 2002; Monteiro e Ribeiro, 2004; Selvatici e Costa, 2007; Benicasa, 2012; Santos e Nascimento Júnior, 2012; Silva et al., 2015) são exemplos de aplicações do AR na robótica móvel. Já os artigos (Kober et al., 2013; Kormushev et al., 2013) levantam diversos estudos no campo de integração de robótica e aprendizado por reforço.

- Sistemas Multiagentes: a aplicação do AR em sistemas inteligentes com vários agentes também é muito comum na literatura (Stone e Veloso, 2000; Bianchi et al., 2007; Busoniu et al., 2008; Aranibar, 2009; Bazzan, 2009; Riveret et al., 2014; Bianchi et al., 2014). Uma aplicação importante é o Futebol de Robôs (Stone et al., 2005; Celiberto et al., 2007; Ottoni et al., 2012; Lamperti et al., 2013b; Fabro et al., 2014; Ottoni et al., 2015b).

- Otimização: Otimização da Produção de Petróleo (Oliveira, 2010), Conformação de Feixe em Arranjo de Antenas (Almeida et al., 2015) e Otimização de Venda de Energia (Berlink et al., 2015). Na linha de Otimização Combinatória foram propostos alguns métodos híbridos em conjunto com o AR, como Algoritmo Ant-Q (Gambardella e Dorigo, 1995), *Global Search* (Miagkikh e Punch, 1999b,a) Aceleração por Heurísticas (Bianchi et al., 2009), Algoritmos Genéticos (Liu e Zeng, 2009; Lima Júnior et al., 2010), Busca Reativa (Santos et al., 2014) e Algoritmos Distribuídos (Mariano e Morales, 1999, 2000).

- Outros exemplos: Controle de Elevadores (Crites e Barto, 1996), Controle de Tráfego Aéreo (Alves et al., 2008) , Controle Ótimo de Descarregadores de Navios (Scárdua et al., 2003), Controle de Sistemas Não Lineares (Teixeira e Bottura, 2016), Roteamento em Redes de Sensores sem Fio (Campos et al., 2012), Segurança Pública (Almeida, 2014), Helicópteros Autônomos (Ng et al., 2006), Jogos (Mnih et al., 2013), Semáforo Inteligente (Silva et al., 2013), Sistemas Elétricos de Potência (Yu e Zhen, 2009), Sistemas Educacionais (Dorça et al., 2013), Telecomunicações (Leite et al., 2012), e Tecnologias Assistivas (Alves et al., 2014; Andrade et al., 2014).

CAPÍTULO 3

Problema do Caixeiro Viajante

Neste capítulo, são apresentados aspectos teóricos que envolvem o Problema do Caixeiro Viajante (PCV). Inicialmente, são descritos conceitos de otimização combinatória. Em sequência, são levantadas definições da modelagem matemática do PCV. Além disso, o capítulo também explana sobre problemas simétricos e assimétricos do PCV. Finalmente, a biblioteca TSPLIB é apresentada.

3.1 Otimização Combinatória

De acordo com Martínez e Santos (1995), a otimização consiste em encontrar os mínimos ou máximos de uma função de várias variáveis, com valores dentro de uma determinada região do espaço multi-dimensional. Além disso, Martínez e Santos (1995) ressaltam que a otimização é um problema matemático com muitas aplicações no "mundo real". De fato, a otimização pode ser aplicada desde em um simples problema de busca pelo menor trajeto para o robô entre dois pontos em uma sala, até em um complexo problema de logística, envolvendo o roteamento de centenas de veículos de entregas de produtos.

Um problema de otimização é composto principalmente pelas variáveis de decisão, função objetivo e restrições (Almeida, 2012). As variáveis de decisão são os parâmetros que são alterados na busca por solucionar o problema. Já as restrições, são imposições no modelo do problema, a fim de garantir uma solução factível. A função objetivo, por sua vez, é o resultado que se deseja minimizar/maximizar (Almeida, 2012).

Para a resolução de problemas de otimização, constantemente é adotado o campo de Pro-

gramação Matemática (PM) (Goldbarg e Luna, 2005). Goldbarg e Luna (2005) apontam que a Programação Matemática pode examinar inúmeras configurações viáveis para o problema proposto, e selecionar dentro de certos critérios, as melhores soluções. Assim, existem alguns grupos de técnicas de solução baseadas em PM, como:

- Programação Linear: as variáveis do modelo são contínuas. Além disso, apresentam comportamento linear, tanto em relação às restrições como à função objetivo (Goldbarg e Luna, 2005).
- Programação Não-Linear: se um modelo de otimização apresentar não-linearidade, seja nas restrições ou função objetivo (Goldbarg e Luna, 2005).
- Programação Inteira: se as variáveis do modelo são restritas a assumir apenas valores discretos. Assim, a condição das variáveis serem inteiras, implica em uma maior complexidade computacional do que em situações com modelos lineares ou não-lineares (Goldbarg e Luna, 2005).

Na otimização, quando o conjunto de soluções viáveis é discreto, o problema é dito de otimização combinatória (Lima Júnior, 2009). Assim, geralmente os problemas de otimização combinatória podem ser modelados adotando a programação inteira. Para isso, são adotadas variáveis de decisão discretas, como $x_i \in \{0,1\}$ (Silva, 2014). Com isso:

- se $x_i = 1$, então o elemento de índice i compõe a solução, ou,
- se $x_i = 0$, então o elemento de índice i não compõe a solução.

Na literatura, são definidos e estudados diversos problemas de otimização combinatória. Alguns desses problemas são:

- Problema da Mochila (*Knapsack Problem*): retrata o desafio de inserir itens em uma mochila otimizando o valor do produto carregado, respeitando o limite de peso da bolsa (Goldbarg e Luna, 2005). Dessa forma, pode ser aplicado em vários casos práticos como: investimento de capital, problema de corte e empacotamento, carregamento de veículos e orçamento (Goldbarg e Luna, 2005).
- Problema de Roteamento de Veículos (PRV): de acordo com Goldbarg e Luna (2005), um sistema de roteamento pode ser considerado como um conjunto organizado de meios que

visa atender demandas de uma rede de transporte. Assim, um PRV aborda basicamente a definição da sequência de visitas que serão realizadas para atender os nós ou arestas de uma rede (Goldbarg e Luna, 2005).

- Problema dos K-Servos (PKS): tem como objetivo planejar o deslocamento de unidades de atendimento (servos) para suprir uma demanda. O PKS se distigue do PRV, por a posição inicial do servo ser um fator importante no resultado e a unidade permanecer ocupando o último nó da rede atendido (Goldbarg e Luna, 2005).

Outro relevante problema de otimização combinatória é o Problema do Caixeiro Viajante (PCV). O PCV será apresentado com mais detalhes na próxima seção.

3.2 Definições do Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante, no inglês *Traveling Salesman Problem*, consiste em determinar a menor rota entre um conjunto de cidades, $C = (c_1, c_2, c_3, ..., c_n)$ (Lima Júnior, 2009). A cada par de cidades é dada uma distância (ou custo) associado, c_{ij}. Como restrição, cada localidade deve ser visitada uma única vez. Além disso, o caixeiro deve iniciar e finalizar o percurso na mesma cidade.

3.2.1 Modelagem Matemática

Geralmente, o PCV é formulado sobre um grafo $G = (N,A)$, em que, N é o conjunto de nós (vértices) e A é o conjunto de arcos (i,j) do problema (Goldbarg e Luna, 2005).

Em seguida, uma formulação matemática apresentada por Bodin et al. (1983) e adotada em trabalhos recentes (Benevides, 2011; Vitor, 2015):

$$Min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij}, \tag{3.1}$$

sujeito a:

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad (\forall j = 1,...,N), \tag{3.2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad (\forall i = 1,...,N), \tag{3.3}$$

$$x_{ij} \in \{0,1\} \qquad (\forall i,j = 1,...,N), \tag{3.4}$$

$$X = x_{ij} \in S \qquad (\forall i,j = 1,...,N), \tag{3.5}$$

em que, a Equação 3.1 retrata o objetivo de minimizar a distância percorrida total do caixeiro na rota. Assim, o custo de deslocamento entre duas cidades (i e j) é dado por c_{ij}. Já a variável de decisão $x_{i,j}$, assume 1 se o arco (i,j) compor a solução e 0 caso contrário. As restrições 3.2 e 3.3 asseguram que cada localidade será visitada uma única vez. Além disso, Equação 3.4 garante que a variável x_{ij} é binária. Por fim, na restrição 3.5, o conjunto S representa qualquer grupo de restrições que eliminem a formação de sub-rotas.

3.2.2 PCV Simétrico e Assimétrico

Neste trabalho, o Problema do Caixeiro Viajante será abordado sobre dois paradigmas: PCV Simétrico e PCV Assimétrico. No caso Simétrico, o custo associado ao deslocamento de uma cidade i para uma localidade j é equivalente ao custo de ir de j para i. Já no problema Assimétrico, o sentido de realização da rota pode alterar o valor da distância total percorrida. Assim:

- PCV Simétrico: $c_{ij} = c_{ji}$,
- PCV Assimétrico: $c_{ij} \neq c_{ji}$.

Dessa forma, o caso Assimétrico apresenta o dobro de possíveis soluções do que o PCV Simétrico. Mais especificamente, um problema Assimétrico possui $(n-1)!$ soluções e o caso Simétrico $\frac{(n-1)!}{2}$ soluções, em que n é o número de nós (cidades). Consequentemente, o PCV Assimétrico apresenta maior custo de resolução que o PCV Simétrico (Pedro, 2013).

A Figura 3.1 apresenta um exemplo de uma rota gerada para um problema com quatro localidades, adotando simetria e assimetria. Para o caso Simétrico (Figura 3.1 - (a)), o custo total é 17 $(2+5+3+7)$. No entanto, para o exemplo Assimétrico (Figura 3.1 - (b)), a distância total percorrida é 17 apenas no sentido anti-horário e 16 $(3+3+4+6)$ no sentido horário.

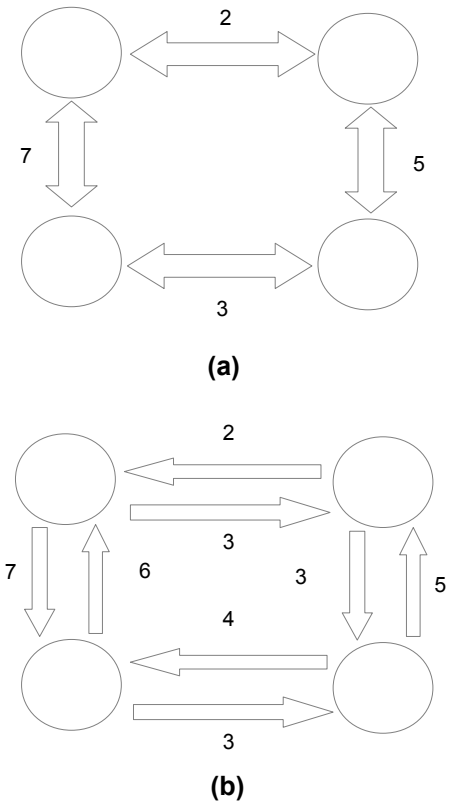


Figura 3.1: Exemplo de rota para o PCV Simétrico (a) e Assimétrico (b) com quatro cidades. Baseado em um exemplo de Pedro (2013).

3.2.3 TSPLIB: Biblioteca de Instâncias

A biblioteca TSPLIB[1] (Reinelt, 1991, 1994), *A Traveling Salesman Problem Library*, é um repositório de dados aberto que reune opções de estudos de caso do Problema do Caixeiro Viajante. Além disso, o repositório fornece o valor da solução ótima para cada instância da biblioteca. Assim, desde que a TSPLIB foi proposta por Reinelt (1991), é adotada como fator de *benchmark* em diversos trabalhos, como (Gambardella e Dorigo, 1995; Miagkikh e Punch, 1999b; Liu e Zeng, 2009; Bianchi et al., 2009; Lima Júnior et al., 2010; Santos et al., 2014; Alipour e Razavi, 2015; Ottoni et al., 2015a; Vitor, 2015).

O repositório TSPLIB apresenta cinco classes de problemas (Reinelt, 1994, 1995):

- *Symmetric traveling salesman problem (TSP)*: contempla instâncias do PCV Simétrico.
- *Asymmetric traveling salesman problem (ATSP)*: contempla instâncias do PCV Assimétrico.

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

- *Hamiltonian cycle problem (HCP)*: esse problema consiste em determinar se um grafo contém um ciclo hamiltoniano.
- *Sequential ordering problem (SOP)*: esse problema é equivalente ao ATSP com restrições adicionais. Essas restrições requerem a precedência de um nó i ser visitado antes de um determinado nó j.
- *Capacitated vehicle routing problem (CVRP)*: esse problema consiste em determinar o roteamento de veículos (caminhões) para atender as demandas de localidades (nós). como restrição, deve-se respeitar a capacidade de transporte de carga de cada caminhão.

Neste trabalho, são abordadas instâncias dos problemas das classes TSP e ATSP da TSPLIB.

CAPÍTULO 4

Metodologia

Neste capítulo são apresentados os procedimentos metodológicos deste trabalho. A seção de modelagem do problema descreve como o Aprendizado por Reforço (AR) foi planejado e implementado para o Problema do Caixeiro Viajante (PCV). Posteriormente, é apresentada uma descrição detalhada dos experimentos formulados para este trabalho. Finalmente, a última seção deste capítulo descreve a modelagem matemática pela Metodologia de Superfície de Resposta proposta neste trabalho para a análise de sensibilidade dos parâmetros do AR.

4.1 Modelo de Aprendizado por Reforço

Para a solução do Problema do Caixeiro Viajante via métodos de Aprendizado por Reforço são necessárias algumas definições iniciais, como ações, estados e reforços do modelo. Assim, a metodologia adotada para o desenvolvimento da estratégia de aprendizagem é dividida em quatro etapas:

1. Definição do conjunto finito de estados do ambiente: nesse caso, os estados são todas as localidades em que o caixeiro viajante (agente) deve acessar. Essa definição garante a resolução do PCV como problema de decisão sequencial (Lima Júnior et al., 2010). Além disso, o conjunto S (estados) sempre tem o mesmo tamanho da instância do problema (Lima Júnior et al., 2010).

2. Definição do conjunto finito de ações que o agente pode realizar: cada ação foi definida como sendo intenção de ir para outra localidade (estado) do problema. Vale ressaltar que,

para evitar a repetição de localidades na rota, as ações que levem aos estados já visitados não devem estar disponíveis (Lima Júnior et al., 2010).

3. Definição dos valores dos reforços, para cada par estado (s) $\times$ ação (a): os reforços foram definidos como as distâncias entre as localidades multiplicada por -1, conforme a Equação 4.1:

$$r_{ij} = -d_{ij}, \tag{4.1}$$

em que, i e j são as localidades, d_{ij} é a distância entre as cidades i e j, e r_{ij} é o reforço recebido por partir de i para j. Assim, quanto maior a distância, mais negativo é o reforço. Dessa forma, espera-se que o agente procure encontrar a distância mais curta entre duas localidades para diminuir a penalidade. Essa abordagem é a mesma adotada por Bianchi et al. (2009).

4. Aplicação do algoritmos de aprendizado por reforço Q-learning e SARSA no simulador desenvolvido: foi desenvolvido um simulador no *software* MATLAB® para realizar os experimentos.

O caixeiro viajante passa então a ser considerado como um agente de aprendizado que deve buscar otimizar os retornos (reforços) pelas tomadas de decisões (ações) na seleção da ordem das localidades (estados) que deve visitar. Vale ressaltar que a estrutura proposta é baseada nas apresentadas por Lima Júnior et al. (2010) e Bianchi et al. (2009).

Para exemplificar a estrutura proposta, a Figura 4.1 mostra uma representação com 4 localidades (PCV Simétrico e Assimétrico), com os valores de distância entre as localidades. Assim, nesse Exemplo, o problema possui 4 estados (S_1, S_2, S_3, S_4).

As matrizes D_1 e D_2 (Equação 4.2) reúnem as distâncias entre as localidades do Exemplo (Figura 4.1), para os casos Simétrico e Assimétrico, respectivamente.

$$D_1 = \begin{bmatrix} 0 & 2 & 7 & 10 \\ 2 & 0 & 4 & 5 \\ 7 & 4 & 0 & 3 \\ 10 & 5 & 3 & 0 \end{bmatrix}, \quad D_2 = \begin{bmatrix} 0 & 2 & 6 & 10 \\ 3 & 0 & 4 & 5 \\ 7 & 9 & 0 & 4 \\ 2 & 3 & 3 & 0 \end{bmatrix} \tag{4.2}$$

Já matrizes R_1 e R_2 (Equação 4.3), representam a estrutura de reforços para o Exemplo (Fi-

gura 4.1). Conforme Equação 4.1, os reforços são as distâncias entre as localidades multiplicada por -1, ou seja, $R_1 = -D_1$ e $R_2 = -D_2$.

$$R_1 = \begin{bmatrix} 0 & -2 & -7 & -10 \\ -2 & 0 & -4 & -5 \\ -7 & -4 & 0 & -3 \\ -10 & -5 & -3 & 0 \end{bmatrix}, \quad R_2 = \begin{bmatrix} 0 & -2 & -6 & -10 \\ -3 & 0 & -4 & -5 \\ -7 & -9 & 0 & -4 \\ -2 & -3 & -3 & 0 \end{bmatrix} \tag{4.3}$$

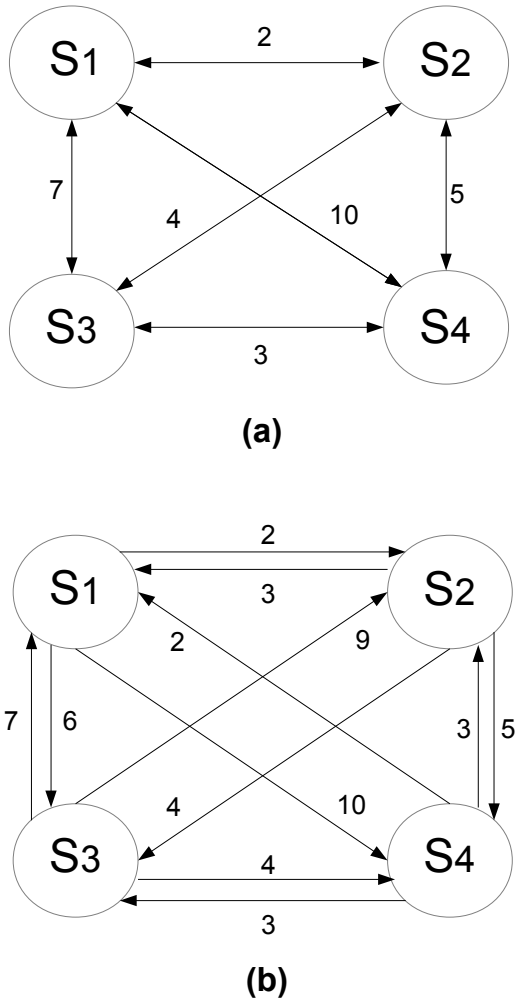


Figura 4.1: Exemplo de representação em estados para um problema com 4 localidades, com os valores de distância entre as localidades. (a) PCV Simétrico. (b) PCV Assimétrico.

4.2 Formulação dos Experimentos

Os experimentos realizados neste trabalho visaram analisar a sensibilidade dos parâmetros do Aprendizado por Reforço na solução do Problema do Caixeiro Viajante. Para isso, foram adotados os algoritmos de AR mais tradicionais na literatura: Q-learning e SARSA. Além disso, o PCV foi analisado sobre duas classes de problemas: Simétricos e Assimétricos.

4.2.1 Instâncias da TSPLIB

Para cada algoritmo, foram realizados testes com o PCV adotando oito instâncias da biblioteca TSPLIB:

- Problemas Simétricos: berlin52, brazil58, kroA100 e kroA200.
- Problemas Assimétricos: br17, ftv33, ftv44 e ftv64.

A Tabela 4.1 especifica o número de localidades e a solução ótima para cada um dos problemas estudados. Vale ressaltar que, a solução ótima representa a menor caminho possível para o caixeiro viajante, ou seja, a rota que minimize a distância percorrida.

Tabela 4.1: Problemas da TSPLIB estudados.

Tipo	**Problema**	**Cidades**	**Solução Ótima**
Simétrico	berlin52	52	7542
	brazil58	58	25395
	kroA100	100	21282
	kroA200	200	29368
Assimétrico	br17	17	39
	ftv33	34	1286
	ftv44	45	1613
	ftv64	65	1839

4.2.2 Experimentos Iniciais

Os experimentos iniciais tiveram como objetivo principal investigar a influência da taxa de aprendizado (α) e a política $\epsilon - greedy$ no desempenho do AR na solução do Problema do Caixeiro Viajante.

Para isso, o AR foi simulado com 3 distintos valores para a política $\epsilon - greedy$: 0, 0,01 e 0,1. Assim, o sistema é:

- 100% guloso para ϵ = 0;
- 99% guloso para ϵ = 0,01;
- 90% guloso para ϵ = 0,1.

Além disso, a taxa de aprendizado foi analisada sobre dois paradigmas:

- α_k = 0,99, taxa de aprendizado constante;
- $\alpha_n(s,a)$, taxa de aprendizado decaindo durante o processo de AR de acordo com a Equação 2.3.

Quanto ao fator de desconto (γ), foi fixado em 0,01 em todas as simulações.

Foram realizados testes com o caixeiro viajante adotando o algoritmo Q-learning e três instâncias da TSPLIB: berlin52, brazil58, kroA100.

Os resultados desta etapa são apresentados na Seção 5.1 (Resultados Iniciais) e também em Ottoni et al. (2015a).

4.2.3 Experimentos para Análise da Taxa de Aprendizado e Fator de Desconto

Nesta etapa de experimentos, para cada um dos oito problemas do PCV abordados (Tabela 4.1), foram realizadas simulações envolvendo um conjunto de 64 combinações dos parâmetros taxa de aprendizado (α) e fator de desconto (γ). Os valores para α e γ são:

- α: 0,01, 0,15, 0,30, 0,45, 0,60, 0,75, 0,90 e 0,99.
- γ: 0,01, 0,15, 0,30, 0,45, 0,60, 0,75, 0,90 e 0,99.

Vale ressaltar que, os valores para α e γ foram selecionados na perspectiva de analisar tanto magnitudes baixas e altas desses parâmetros, na compreensão do espaço de definição possível entre 0 e 1. No entanto, a modelagem com quaisquer outros valores para esses parâmetros também é perfeitamente possível (Ottoni et al., 2016d).

Cada simulação foi padronizada em cinco épocas (repetições) com 1000 (mil) episódios. Sendo que, cada episódio teve como resposta a distância total (custo) percorrida pelo agente na rota da instância.

Já para a definição do parâmetro ϵ da política de seleção de ações $\epsilon - greedy$, foram analisados os resultados obtidos em (Ottoni et al., 2015a) (Experimentos Iniciais). Ottoni et al. (2015a)

analisam os efeitos de ϵ sobre três instâncias do PCV: berlin52, brazil58 e kroA100. A utilização da $\epsilon - greedy$, com $\epsilon = 0{,}01$, mostrou-se o método mais eficiente para todos os estudos de casos, entre os valores analisados ($\epsilon = 0$ e $\epsilon = 0{,}1$) (Ottoni et al., 2015a). Mais detalhes sobre as análises para a definição da $\epsilon - greedy$ são apresentados da Seção 5.1 (Resultados Iniciais).

A Tabela 4.2 resume as definições dos experimentos para cada algoritmo e instância adotados.

Tabela 4.2: Resumo dos experimentos para cada algoritmo e instância adotados.

	Quantidade	**Valores**
α	8	0,01; 0,15; 0,30; 0,45; 0,60; 0;75; 0,90; 0;99
γ	8	0,01; 0,15; 0,30; 0,45; 0,60; 0;75; 0,90; 0;99
ϵ	1	0,01
Combinações	8 × 8 × 1 = 64	-
Épocas por Combinação	5	-
Episódios por Época	1000 (mil)	-
Episódios por Combinação	5 × 1000 5000	-
Total de Épocas	64 × 5 = 320	-
Total de Episódios	320 × 1000 = 320000	-

Os resultados desta etapa de experimentos são abordados na modelagem matemática pela Metodologia de Superfície de Resposta, explicada em seguida.

4.3 Modelagem Matemática pela Metodologia de Superfície de Resposta

4.3.1 Metodologia de Superfície de Resposta

A Metodologia de Superfície de Resposta, em inglês *Response Surface Methodology (RSM)*, reúne um conjunto de técnicas estatísticas para a otimização de processos (Myers et al., 2009). A medida de desempenho é denominada resposta. Já as variáveis de entrada são ditas variáveis independentes (VIs) (Myers et al., 2009).

Os modelos de superfície de resposta apresentam a mesma estrutura dos modelos de regressão linear múltipla (Myers et al., 2009). Assim, as Equações 4.4 e 4.5 apresentam a estrutura dos modelos RSM de 1ª e 2ª ordem, respectivamente, com duas VIs (x_1 e x_2):

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + e, \tag{4.4}$$

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1^2 + \beta_4 x_2^2 + \beta_5 x_1 x_2 + e. \tag{4.5}$$

O efeito do erro (resíduo) na resposta é representado por e. Para a estimação dos coeficientes do modelo (β), normalmente é adotado o método de mínimos quadrados, assumindo distribuição normal com média zero e variância constante (Myers et al., 2009).

De acordo com Myers et al. (2009), os modelos de 2ª ordem são mais adotados por serem mais flexíveis e mais bem ajustáveis em problemas de superfícies reais.

Neste trabalho, a análise dos resultados com a RSM compreende três etapas:

- Análise das medidas de adequação dos modelos: o objetivo é verificar se os modelos atendem algumas medidas de adequação, como normalidade dos resíduos, coeficiente de determinação múltipla e significância dos coeficientes individuais.
- Análise dos gráficos de contornos e superfícies: o objetivo é visualizar graficamente como a variável resposta é influenciada pelos níveis dos parâmetros α e γ.
- Análise e experimentos com os pontos estacionários: verificar os parâmetros que otimizam a resposta em um modelo. Além disso, os valores dos pontos estacionários são adotados em novos experimentos para analisar o desempenho dos parâmetros ajustados pelos modelos.

4.3.2 Modelagem Matemática

Neste trabalho, foram ajustados 16 modelos matemáticos de Superfície de Resposta de 2ª ordem. Esses modelos têm como objetivo representar a sensibilidade aos parâmetros (α e γ) no desempenho do AR na solução de cada instância do Problema do Caixeiro Viajante estudada, conforme Tabela 4.3.

Assim, a estrutura dos modelos propostos é composta por três variáveis: y, α e γ. A variável resposta (y) representa a média da distância percorrida pelo caixeiro na rota. Para cada uma das cinco repetições de cada instância, foi calculada uma média para cada combinação de α e γ. Além disso, as variáveis independentes são $VI_1 = \alpha$ e $VI_2 = \gamma$. Dessa forma, os modelos possuem o formato da Equação 4.6:

$$y = \beta_0 + \beta_1\alpha + \beta_2\gamma + \beta_3\alpha^2 + \beta_4\gamma^2 + \beta_5\alpha\gamma + e. \tag{4.6}$$

A Equação 4.7, por sua vez, representa a estrutura dos modelos ajustados tendo como saída a resposta predita $\hat{y}_i$. Nesse caso, o erro não aparece na Equação 4.7, pois e_i é a diferença entre uma observação (y_i) e a respectiva resposta predita ($\hat{y}_i$), ou seja, $e_i = y_i - \hat{y}_i$.

$$\hat{y} = \beta_0 + \beta_1\alpha + \beta_2\gamma + \beta_3\alpha^2 + \beta_4\gamma^2 + \beta_5\alpha\gamma. \tag{4.7}$$

Tabela 4.3: Definição dos modelos de Superfície de Resposta.

Número	**Modelo**	**Algoritmo**	**Instância**
1	Qberlin52	Q-learning	berlin52
2	Sberlin52	SARSA	
3	Qbrazil58	Q-learning	brazil58
4	Sbrazil58	SARSA	
5	QkroA100	Q-learning	kroA100
6	SkroA100	SARSA	
7	QkroA200	Q-learning	kroA200
8	SkroA200	SARSA	
9	Qbr17	Q-learning	br17
10	Sbr17	SARSA	
11	Qftv33	Q-learning	ftv33
12	Sftv33	SARSA	
13	Qftv44	Q-learning	ftv44
14	Sftv44	SARSA	
15	Qftv64	Q-learning	ftv64
16	Sftv64	SARSA	

Para facilitar a identificação dos modelos, cada estrutura recebeu um código, dado pela união entre a primeira letra do algoritmo e o nome da instância. Por exemplo, para o modelo que representa as simulações do algoritmo Q-learning no problema berlin52, o código é Qberlin52. Os demais códigos de identificação dos modelos são apresentados na Tabela 4.3.

Para o ajuste dos modelos foi adotado o pacote RSM do *software* estatístico R (Lenth, 2009; R Core Team, 2013).

CAPÍTULO 5

Resultados

Neste capítulo, são apresentadas as análises dos resultados deste trabalho. Inicialmente, são apresentados alguns resultados iniciais relativos a análise do desempenho do AR de acordo com a política $\epsilon - greedy$ e o paradigma de taxa de aprendizado (constante ou decaindo). Na sequência, são descritos os resultados da modelagem com a adoção da Metodologia de Superfície de Resposta. Nesse aspecto, a RSM é abordada em etapas: análise de adequação dos modelos ajustados, análise gráfica e análise e experimentos com os pontos estacionários.

5.1 Resultados Iniciais

Nesta seção, são apresentados os resultados referentes aos experimentos iniciais (Seção 4.2.2). O objetivo é investigar a influência da taxa de aprendizado (α) e da política $\epsilon - greedy$ no desempenho do aprendizado por reforço, aplicando o Q-learning na solução das instâncias berlin52, brazil58 e kroA100. Vale ressaltar que, os resultados desta seção também são apresentados em Ottoni et al. (2015a).

5.1.1 Resultados com 52 localidades - berlin52

Para a instância berlin52, a menor distância calculada foi 8009. Esse valor foi encontrado quando adotado $\epsilon = 0{,}01$ e $\alpha_n(s,a)$, e também para $\epsilon = 0{,}1$ e $\alpha_n(s,a)$. A Tabela 5.1 apresenta os resultados mínimos encontrados de acordo com a taxa de aprendizado e a política $\epsilon - greedy$.

As Figuras 5.1 e 5.2 apresentam os gráficos de aprendizado ao longo dos episódios para a instância berlin52, considerando os resultados de distâncias calculadas nas rotas. Nesse sentido,

Tabela 5.1: Menor distância calculada para a instância berlin52.

$\epsilon - greedy$	α_k	$\alpha_n(s,a)$
0	9148	8146,3
0,01	8009,5	8009
0,1	8037	8009

pela Figura 5.1, é possível visualizar a estabilização do sistema adotando α_k e $\epsilon = 0$. Nesse caso, a combinação entre a taxa de aprendizado constante (α_k) e o sistema 100% guloso ($\epsilon = 0$), levou a convergência para o valor 9148, distância muito superior ao valor encontrado pelas demais combinações. Já a Figura 5.2, apresenta e evolução da distância calculada, adotando $\alpha_n(s,a)$ para os três valores da política $\epsilon - greedy$.

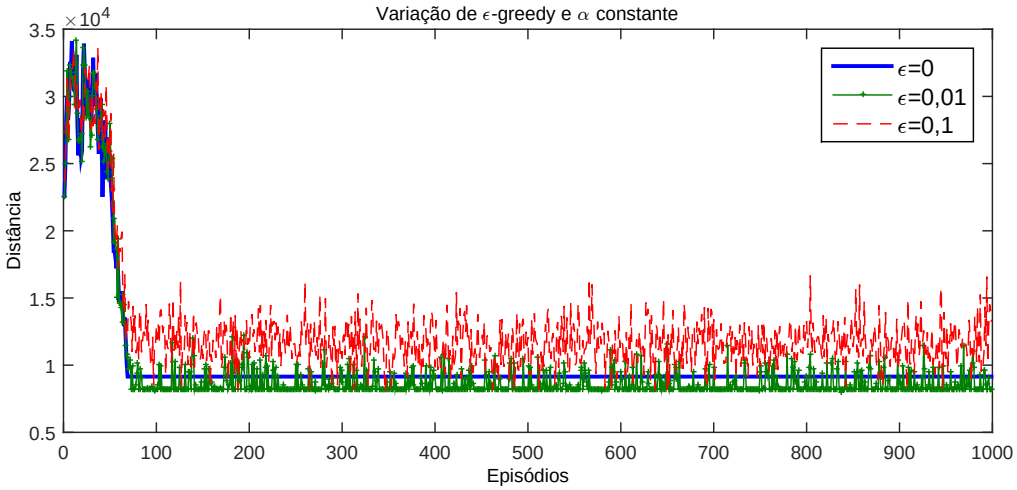


Figura 5.1: Distância calculada versus o episódio de aprendizado para berlin52 com $\alpha_k = 0{,}99$.

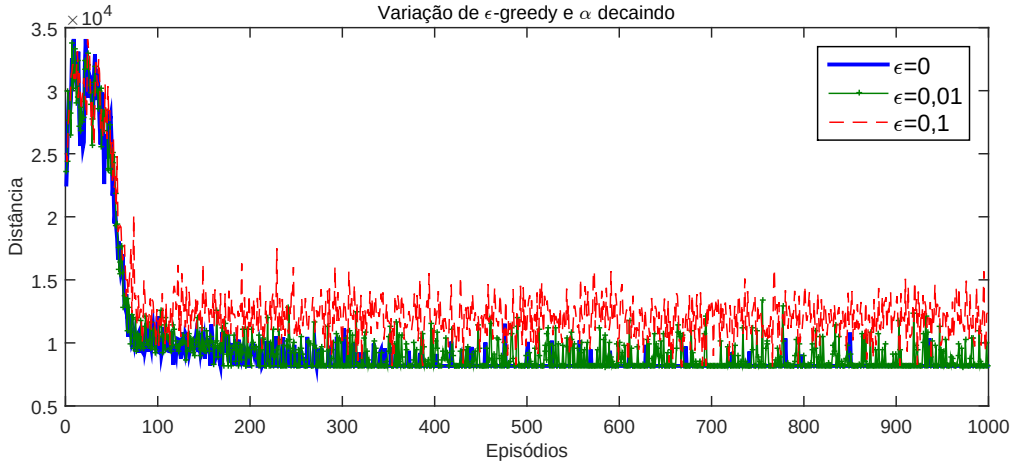


Figura 5.2: Distância calculada versus o episódio de aprendizado para berlin52 com $\alpha_n(s,a)$.

As Figuras 5.3 à 5.4, mostram o comportamento do AR, a partir dos caminhos encontrados para a instância berlin52, considerado o início do aprendizado (Figura 5.3), melhor solução para o sistema 100% guloso e taxa de aprendizado constante (Figura 5.4), e melhor solução para o sistema 100% guloso e taxa de aprendizado decaindo (Figura 5.5).

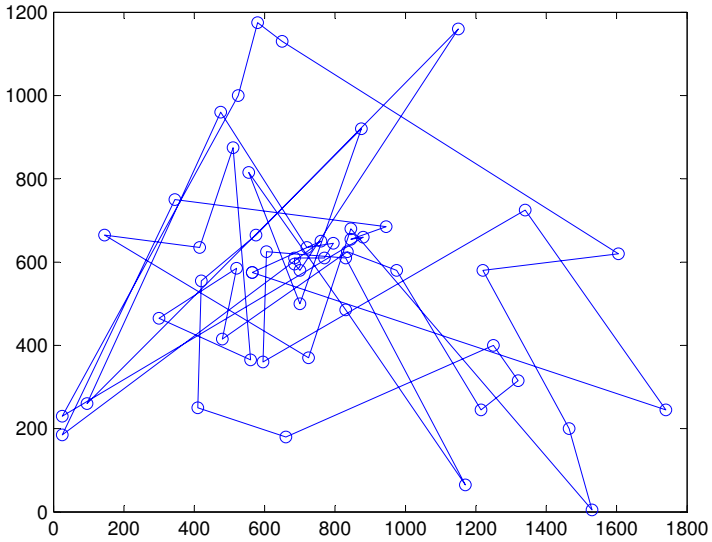
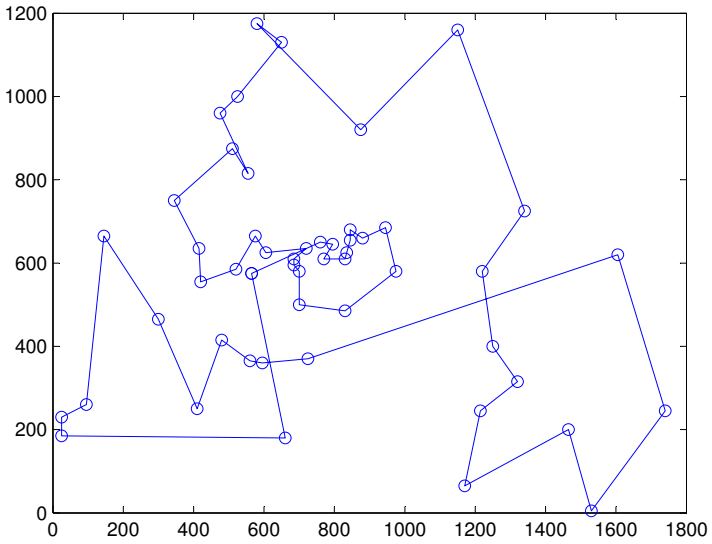


Figura 5.3: Caminho definido com a distância percorrida igual a 22561 para a instância berlin52, para o primeiro episódio do aprendizado com $\epsilon = 0$ e $\alpha_k = 0{,}99$.



,

Figura 5.4: Caminho definido com a distância percorrida igual a 9148 para a instância berlin52 com $\epsilon = 0$ e $\alpha_k = 0{,}99$.

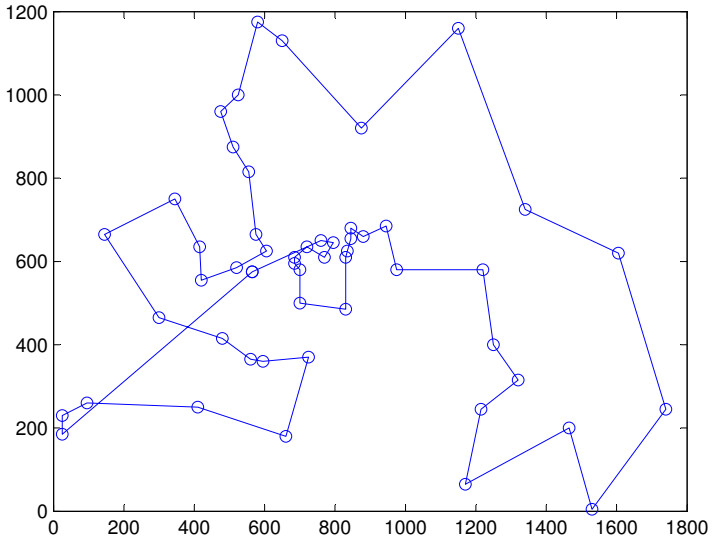


Figura 5.5: Caminho definido com a distância percorrida igual a 8146,3 para a instância berlin52 com $\epsilon = 0$ e $\alpha_n(s,a)$.

5.1.2 Resultados com 58 localidades - brazil58

Para a instância brazil58, a menor distância calculada foi 27753. Esse valor foi encontrado quando adotado $\epsilon = 0{,}01$ e $\alpha_n(s,a)$. A Tabela 5.3 apresenta os resultados mínimos encontrados de acordo com a taxa de aprendizado e a política $\epsilon - greedy$.

Tabela 5.2: Menor distância calculada para a instância brazil58.

$\epsilon - greedy$	α_k	$\alpha_n(s,a)$
0	31223	27922
0,01	27922	27753
0,1	28930	28117

5.1.3 Resultados com 100 localidades - kroA100

Para a instância kroA100, a menor distância calculada foi 25748. Esse valor foi encontrado quando adotado $\epsilon = 0{,}01$ e $\alpha_n(s,a)$. A Tabela 5.3 apresenta os resultados mínimos encontrados de acordo com a taxa de aprendizado e a política $\epsilon - greedy$.

Tabela 5.3: Menor distância calculada para a instância kroA100.

$\epsilon - greedy$	α_k	$\alpha_n(s,a)$
0	28437	26291
0,01	26268	25748
0,1	28832	28944

As Figuras 5.6 à 5.7 mostram o comportamento do AR, a partir dos caminhos encontrados para a instância kroA100, considerando o início do aprendizado (Figura 5.6), melhor solução para o sistema 100% guloso e taxa de aprendizado constante (Figura 5.7), e melhor solução para o sistema 100% guloso e taxa de aprendizado decaindo (Figura 5.8).

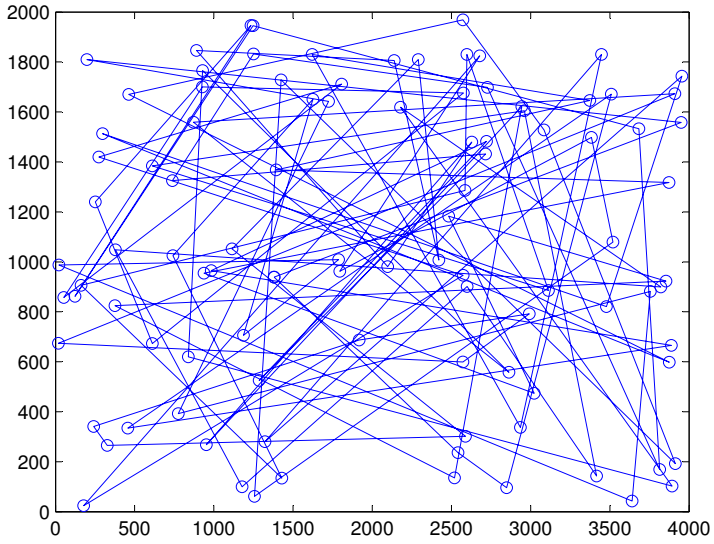


Figura 5.6: Caminho definido com a distância percorrida igual a 189260 para a instância kroA100, para o primeiro episódio do aprendizado com $\epsilon = 0$ e $\alpha_k = 0{,}99$ (constante).

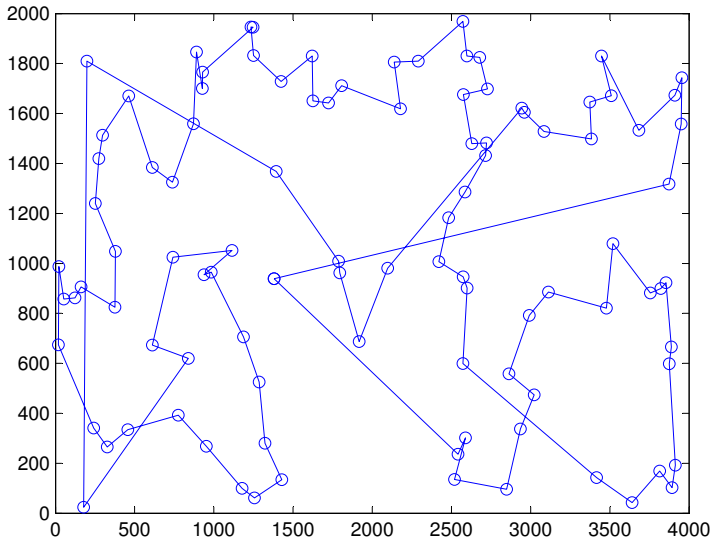


Figura 5.7: Caminho definido com a distância percorrida igual a 28437 para a instância kroA100 com ϵ=0 e $\alpha_k = 0{,}99$ (constante).

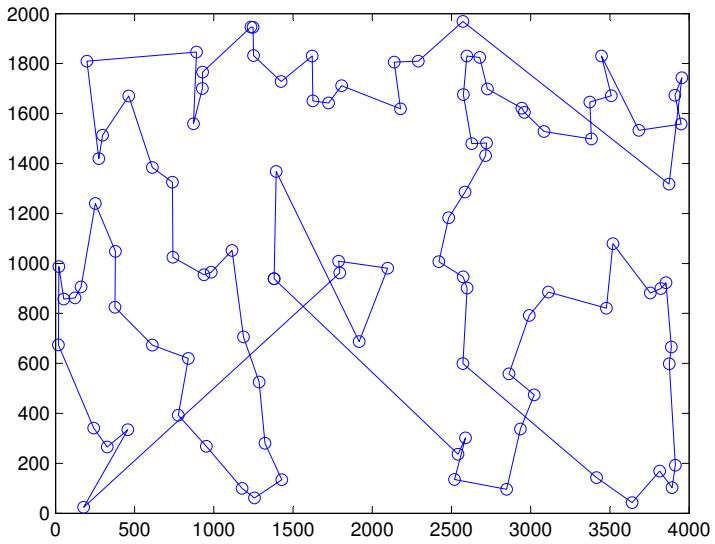


Figura 5.8: Caminho definido com a distância percorrida igual a 26291 para a instância kroA100 com ϵ=0 e $\alpha_n(s,a)$ (decaindo).

5.1.4 Resultados Gerais

A Tabela 5 resume os melhores resultados encontrados nos Experimentos Iniciais para cada problema.

Tabela 5.4: Melhor solução encontrada para cada problema nos experimentos iniciais.

Problema	**Solução Ótima**	**Melhor Solução**	ϵ	α
berlin52	7542	8009	0,01/0,1	$\alpha_n(s,a)$
brazil58	25395	27753	0,01	$\alpha_n(s,a)$
kroA100	21282	25748	0,01	$\alpha_n(s,a)$

Dessa forma, conforme evidenciado por Sutton e Barto (1998), o método guloso ($\epsilon = 0$) pode facilmente encontrar uma solução local e ficar preso nesse ponto. Além disso, método $\epsilon = 0{,}1$ explora mais, mas nunca seleciona a ação mais bem estimada por mais de 91% do tempo (Sutton e Barto, 1998). Já o método $\epsilon = 0{,}01$ teve um melhor desempenho do que os outros dois valores para $\epsilon - greedy$ (Sutton e Barto, 1998). Assim, a utilização da política $\epsilon - greedy$, com $\epsilon = 0{,}01$, mostrou-se o método mais eficiente para os estudos de casos, entre os valores de ϵ analisados.

Quanto a taxa de aprendizado, em quase todos os casos $\alpha_n(s,a)$ (decaindo ao longo do tempo) apresentou rendimento superior a $\alpha_k = 0{,}99$ (constante).

5.2 Resultados da Metodologia de Superfície de Resposta

Os resultados para o ajuste dos modelos de superfície de resposta, referentes aos experimentos da Seção 4.2.3 e modelagem matemática proposta na Seção 4.3, são apresentados em seguida em três etapas:

- Análise das medidas de adequação dos modelos.
- Análise dos gráficos de contornos e superfícies.
- Análise e experimentos com os pontos estacionários.

Vale ressaltar que, parte dos resultados desta seção também são apresentados em Ottoni et al. (2016b).

5.2.1 Medidas de Adequação dos Modelos

Os modelos ajustados devem satisfazer algumas medidas de adequação. Nesse sentido são realizadas três análises:

- Análise da normalidade dos resíduos.
- Análise do coeficiente de determinação múltipla e coeficiente de determinação múltipla ajustado.
- Teste de significância dos coeficientes individuais.

A primeira análise de adequação visa verificar se os resíduos dos modelos estão distribuídos normalmente (Hines et al., 2006). Defini-se os resíduos como $e_i = y_i - \hat{y}_i$, $i = 1, 2, ..., n$, em que y_i é uma observação e $\hat{y}_i$ é o correspondente valor estimado a partir do modelo de regressão (Hines et al., 2006). Utilizou-se o teste de Kolmogorov-Smirnov (KS) (Razali e Wah, 2011) para a verificação dessa suposição. No teste KS, a hipótese nula é que os resíduos seguem uma distribuição normal (H_0), e a hipótese alternativa (H_1) que os resíduos não tem essa distribuição (Razali e Wah, 2011). Os correspondentes p-valores do teste KS para os modelos são apresentados na Tabela 5.5 e denotados por p_{KS}.

Apenas os modelos SkroA100 ($p_{KS} = 0{,}0292$), QkroA200 ($p_{KS} = 1{,}19 \times 10^{-6}$), Qbr17 ($p_{KS} = 0{,}0104$) e Sftv64 ($p_{KS} = 0{,}0025$), não confirmaram a hipótese de normalidade dos resíduos, pois $p_{KS} < 0{,}05$. Assim, para garantir a qualidade das análises subsequentes, esses

modelos foram descartados. Dessa forma, apenas 12 modelos são analisados nas próximas seções.

Outros componentes de análise da adequação de um modelo de superfície de resposta são: coeficiente de determinação múltipla (R^2) e coeficiente de determinação múltipla ajustado (R^2_a) (Myers et al., 2009). Esses coeficientes, definidos entre 0 e 1, indicam o quanto da variabilidade é explicada pelo modelo. Assim, se R^2 e R^2_a se aproximam de 1, apontam um bom ajuste do modelo à amostra (Hines et al., 2006). Não serão feitos estudos aprofundados para verificar se o modelo proposto é ou não adequado para o fim, pois essas análises podem ser bem amplas, conforme Ferreira et al. (2016). A Tabela 5.5 apresenta os valores ajustados para R^2, R^2_a e p_{KS} para cada um dos modelos.

Tabela 5.5: Medidas de adequação dos modelos.

Modelo	p_{KS}	R^2	R^2_a
Qberlin52	0,6801	0,8914	0,8896
Sberlin52	0,2491	0,9037	0,9022
Qbrazil58	0,2769	0,9008	0,8992
Sbrazil58	0,3649	0,9080	0,9065
QkroA100	0,1068	0,8959	0,8942
SkroA100	*0,0292*	0,9029	0,9014
QkroA200	*1,19*$\times 10^{-6}$	0,7897	0,7864
SkroA200	0,1943	0,8965	0,8948
Qbr17	0,0104	0,6959	0,6910
Sbr17	0,3025	0,8377	0,8352
Qftv33	0,4205	0,8712	0,8691
Sftv33	0,3692	0,8872	0,8854
Qftv44	0,4019	0,8817	0,8798
Sftv44	0,1803	0,8932	0,8915
Qftv64	*0,0536*	0,9015	0,8999
Sftv64	*0,0025*	0,9091	0,9076

As Tabelas 5.6 e 5.7 apresentam os coeficientes ajustados para cada modelo em estudo, adotando os algoritmos Q-learning e SARSA, respectivamente. Os testes de significância de coeficientes individuais para este trabalho revelaram que, para os 12 modelos, todos os coeficientes são altamente significantes ($p < 0{,}001$). O termo Intercepto refere-se ao coeficiente linear (β_0) da estrutura do modelo (Equação 4.6) proposto.

5.2.2 Gráficos de Superfície e Contornos

A metodologia de superfície de resposta fornece duas ferramentas gráficas para análise: gráfico de contornos e superfície de resposta (Myers et al., 2009).

Tabela 5.6: Coeficientes para os modelos com algoritmo Q-learning.

Modelo	**Coeficiente**	β	p
Qberlin52	Intercepto	19051	$< 2 \times 10^{-16}$
	α	-23477	$< 2 \times 10^{-16}$
	γ	-9839	$< 2 \times 10^{-16}$
	α^2	15476	$< 2 \times 10^{-16}$
	γ^2	17169	$< 2 \times 10^{-16}$
	$\alpha\gamma$	6262	$9{,}6 \times 10^{-14}$
Qbrazil58	Intercepto	70114	$< 2 \times 10^{-16}$
	α	-89081	$< 2 \times 10^{-16}$
	γ	-39545	$< 2 \times 10^{-16}$
	α^2	56615	$< 2 \times 10^{-16}$
	γ^2	67766	$< 2 \times 10^{-16}$
	$\alpha\gamma$	29422	$< 2 \times 10^{-16}$
QkroA100	Intercepto	92844	$< 2 \times 10^{-16}$
	α	-134710	$< 2 \times 10^{-16}$
	γ	-50035	$1{,}74 \times 10^{-14}$
	α^2	85159	$< 2 \times 10^{-16}$
	γ^2	92922	$< 2 \times 10^{-16}$
	$\alpha\gamma$	32659	$3{,}67 \times 10^{-12}$
Qftv33	Intercepto	3157	$< 2 \times 10^{-16}$
	α	-3927	$< 2 \times 10^{-16}$
	γ	1639	$< 2 \times 10^{-16}$
	α^2	2682	$< 2 \times 10^{-16}$
	γ^2	2648	$< 2 \times 10^{-16}$
	$\alpha\gamma$	728	$3{,}24 \times 10^{-8}$
Qftv44	Intercepto	4380	$< 2 \times 10^{-16}$
	α	-5600	$< 2 \times 10^{-16}$
	γ	-2087	$1{,}48 \times 10^{-15}$
	α^2	3684	$< 2 \times 10^{-16}$
	γ^2	3564	$< 2 \times 10^{-16}$
	$\alpha\gamma$	1243	$3{,}09 \times 10^{-11}$
Qftv64	Intercepto	6097	$< 2 \times 10^{-16}$
	α	-8162	$< 2 \times 10^{-16}$
	γ	-2986	$3{,}41 \times 10^{-16}$
	α^2	5100	$< 2 \times 10^{-16}$
	γ^2	5266	$< 2 \times 10^{-16}$
	$\alpha\gamma$	2072	$4{,}97 \times 10^{-15}$

O gráfico de contornos oferece uma visualização bidimensional entre as VIs (α e γ) e a variável resposta (y) do modelo. Assim, nesse tipo de gráfico, as VIs são dadas nas escalas x e y e os valores de resposta são representados pelas linhas de contornos. Nesse sentido, um gráfico de contornos é semelhante a um mapa topográfico, onde são representados os valores de latitude (eixo x), longitude (eixo y) e elevação (contornos). Assim, a partir de linhas de contornos é possível identificar regiões que se aproximam do mínimo ou máximo da resposta ajustada.

Tabela 5.7: Coeficientes para os modelos com algoritmo SARSA.

Modelo	**Coeficiente**	β	p
Sberlin52	Intercepto	18656	$< 2 \times 10^{-16}$
	α	-22730	$< 2 \times 10^{-16}$
	γ	-7097	$1{,}9 \times 10^{-11}$
	α^2	14985	$< 2 \times 10^{-16}$
	γ^2	14487	$< 2 \times 10^{-16}$
	$\alpha\gamma$	6718	$< 2 \times 10^{-16}$
Sbrazil58	Intercepto	68807	$< 2 \times 10^{-16}$
	α	-85423	$< 2 \times 10^{-16}$
	γ	-30211	$2{,}16 \times 10^{-13}$
	α^2	54619	$< 2 \times 10^{-16}$
	γ^2	58197	$< 2 \times 10^{-16}$
	$\alpha\gamma$	30530	$< 2 \times 10^{-16}$
SkroA200	Intercepto	191276	$< 2 \times 10^{-16}$
	α	-236975	$< 2 \times 10^{-16}$
	γ	-75721	$1{,}07 \times 10^{-11}$
	α^2	147295	$< 2 \times 10^{-16}$
	γ^2	147568	$< 2 \times 10^{-16}$
	$\alpha\gamma$	62817	$1{,}59 \times 10^{-14}$
Sbr17	Intercepto	116,829	$< 2 \times 10^{-16}$
	α	-86,174	$< 2 \times 10^{-16}$
	γ	12,367	0,01779
	α^2	67,467	$< 2 \times 10^{-16}$
	γ^2	26,247	$2{,}26 \times 10^{-8}$
	$\alpha\gamma$	10,321	0,00656
Sftv33	Intercepto	3110	$< 2 \times 10^{-16}$
	α	-3888	$< 2 \times 10^{-16}$
	γ	-1281	$4{,}64 \times 10^{-14}$
	α^2	2658	$< 2 \times 10^{-16}$
	γ^2	2278	$< 2 \times 10^{-16}$
	$\alpha\gamma$	877	$9{,}28 \times 10^{-13}$
Sftv44	Intercepto	4298	$< 2 \times 10^{-16}$
	α	-5442	$< 2 \times 10^{-16}$
	γ	-1593	$3{,}93 \times 10^{-11}$
	α^2	3550	$< 2 \times 10^{-16}$
	γ^2	3079	$< 2 \times 10^{-16}$
	$\alpha\gamma$	1401	$3{,}29 \times 10^{-15}$

Neste trabalho, os gráficos de contornos mostram em duas dimensões como a taxa de aprendizado (α) e o fator de desconto (γ) influenciam na distância percorrida (contornos - variável resposta) pelo caixeiro viajante na rota. As Figuras 5.9 e 5.10 apresentam os gráficos de contornos para os modelos Sberlin52 e Sftv33, referentes aos experimentos com as instâncias berlin52 e ftv33, respectivamente, e adoção do algoritmo SARSA. A região vermelha indica o conjunto de pontos, dados pela relação de α e γ, que produzem valores mais baixos para a variável resposta.

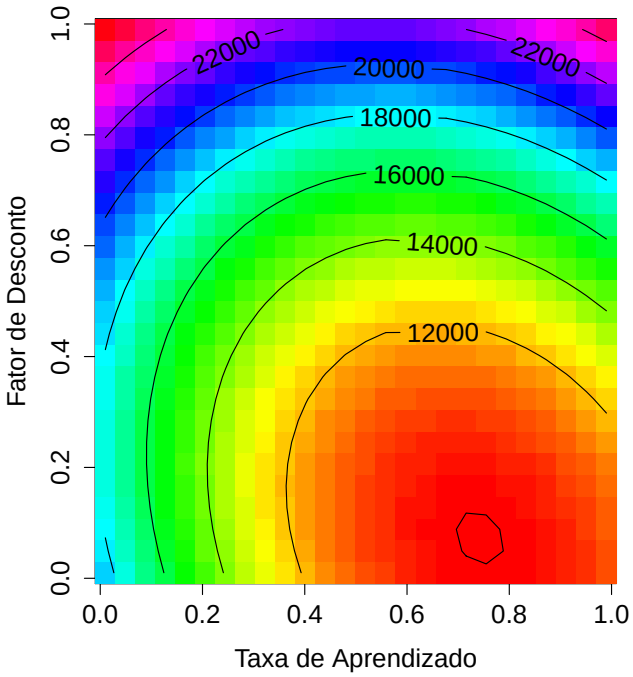


Figura 5.9: Gráficos de contornos para o modelo Sberlin52 (berlin52 - SARSA).

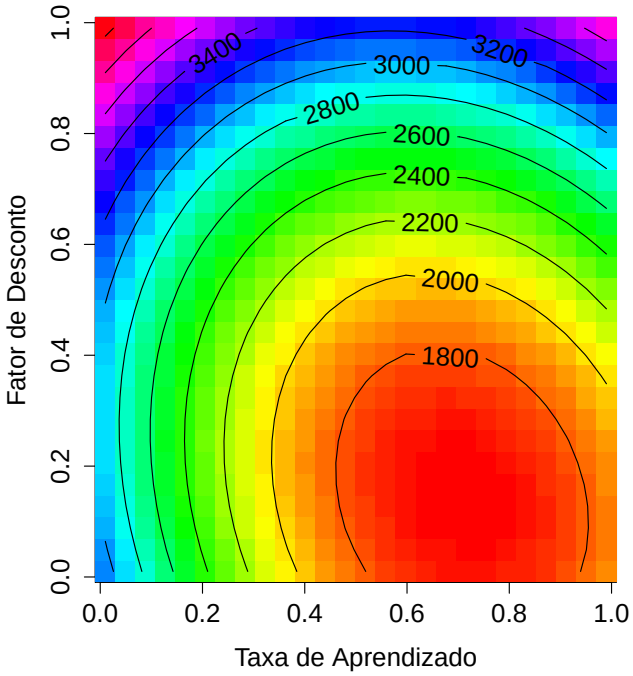


Figura 5.10: Gráficos de contornos para o modelo Sftv33 (ftv33 - SARSA).

Já para a análise em três dimensões (3D), a ferramenta adotada é o gráfico de superfície de resposta. De forma similar ao gráfico de contornos, é possível visualizar a relação entre as variáveis do modelo (α, γ e y). As VIs (α e γ) são exibidas nas escalas x e y do gráfico 3D. Já a variável resposta (distância - eixo z) é representada pela superfície. As Figuras 5.11 e 5.12 apresenta essa visualização em superfície para os modelos Sberlin52 e Sftv33. Nesses gráficos em 3D, a região em vermelho também indica onde a variável resposta (distância) tende a ser minimizada.

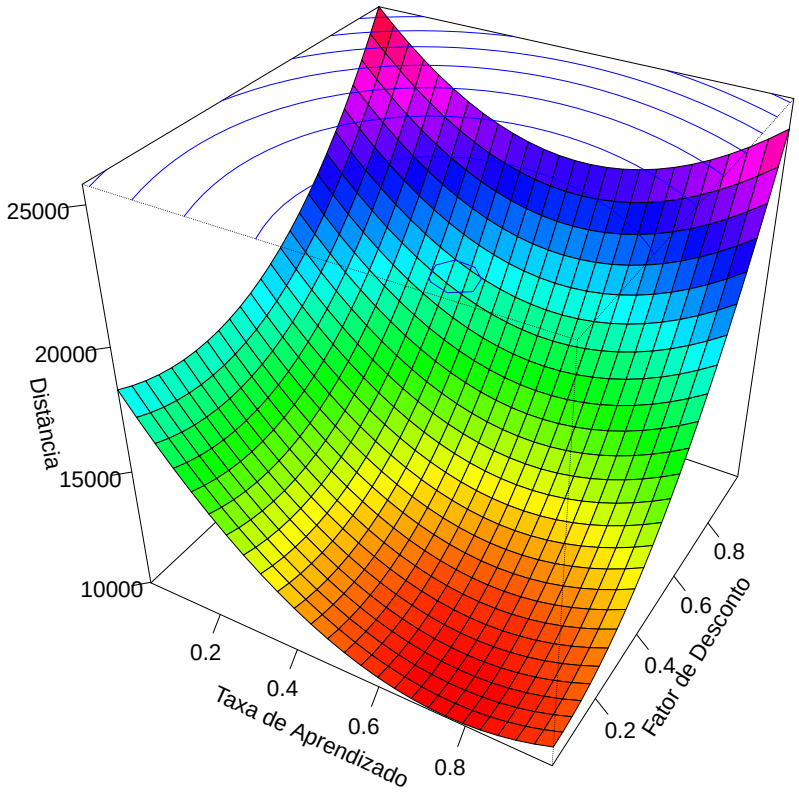


Figura 5.11: Superfície de resposta para o modelo Sberlin52 (Berlin52 - SARSA).

5.2.3 Pontos Estacionários

Na metodologia de superfície de resposta, a identificação dos pontos estacionários (mínimo, máximo, ou ponto de sela) é interessante para verificar os valores que otimizam a resposta predita no modelo (Myers et al., 2009). Para ilustrar como localizar um ponto estacionário (x_0), o modelo proposto, $\hat{y} = \beta_0 + \beta_1\alpha + \beta_2\gamma + \beta_3\alpha^2 + \beta_4\gamma^2 + \beta_5\alpha\gamma$ (Equação 4.7), será representado na forma matricial (Equação 5.1) (Myers et al., 2009):

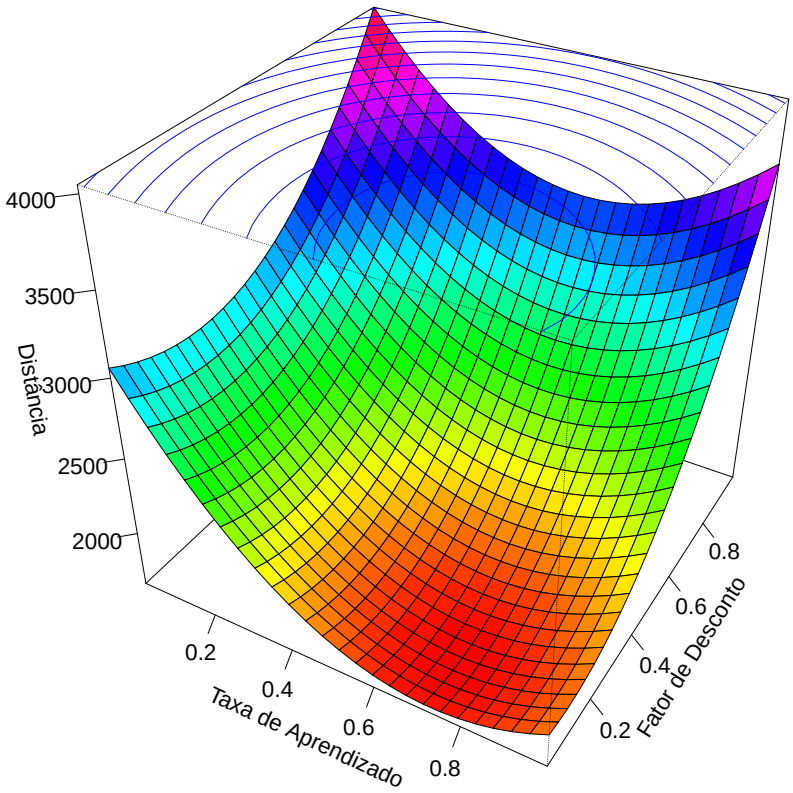


Figura 5.12: Superfície de resposta para o modelo Sftv33 (ftv33- SARSA).

$$\hat{y} = \beta_0 + x'b + x'Bx, \tag{5.1}$$

em que, β_0, b, e B são os valores estimados para intercepto, coeficientes lineares e coeficiente de segunda ordem, respectivamente. Assim, os termos matriciais da Equação 5.1 são definidos nas Equações 5.2, 5.3 e 5.4:

$$x' = [\alpha \quad \gamma], \tag{5.2}$$

$$b = [\beta_1 \quad \beta_2], \tag{5.3}$$

$$B = \begin{bmatrix} \beta_3 & \beta_5/2 \\ \beta_5/2 & \beta_4 \end{bmatrix}. \tag{5.4}$$

Realizando a derivada parcial de $\hat{y}$ em relação a x e igualando a zero, tem-se (Equação 5.5):

$$\frac{\partial \hat{y}}{\partial x} = b + 2Bx = 0. \tag{5.5}$$

Em seguida, ao isolar x encontra-se o ponto estacionário (Equação 5.6):

$$x_0 = -\frac{1}{2}B^{-1}b. \tag{5.6}$$

No Problema do Caixeiro Viajante, o objetivo é minimizar a distância percorrida na rota. Sendo assim, os pontos estacionários desejados são os mínimos nas superfícies modeladas. Vale ressaltar que a definição dos pontos estacionários remete a um segundo problema de otimização neste trabalho, ou seja, encontrar os valores dos parâmetros α e γ que minimizam a resposta predita $\hat{y}$ em cada um dos modelos ajustados. Dessa forma, o fato de x_0 minimizar uma superfície de resposta não garante encontrar a melhor resposta na execução do AR na solução do PCV.

Neste trabalho, os pontos estacionários foram obtidos no *software* R (Myers et al., 2009; Lenth, 2009), e são apresentados nas Tabelas 5.8 e 5.9.

Tabela 5.8: Pontos estacionários para o algoritmo Q-learning.

Modelo	α	γ
Qberlin52	0,7273439	0,1538926
Qbrazil58	0,7534082	0,1282272
QkroA100	0,7650819	0,1347810
Qftv33	0,7031901	0,2128110
Qftv44	0,7320604	0,1652470
Qftv64	0,7735304	0,1312989

Tabela 5.9: Pontos estacionários para o algoritmo SARSA.

Modelo	α	γ
Sberlin52	0,7420503	0,0729088
Sbrazil58	0,7655741	0,0587433
SkroA200	0,7853619	0,0894069
Sbr17	0,6666903	-0,366666*
Sftv33	0,7074018	0,1449799
Sftv44	0,7490532	0,0882221

*Para o modelo Sbr17 foi considerado $\gamma = 0$,
pois o fator de desconto deve ser definido no intervalo $[0,1]$.

Em seguida, foi realizada uma nova fase de experimentos para verificar o desempenho do

AR com a adoção do valores dos pontos estacionários para os parâmetros α e γ. Assim, as combinações foram simuladas em cinco repetições com 10000 (dez mil) episódios. As Tabelas 5.11 e 5.10 apresentam os melhores resultados encontrados para cada instância, com os valores de taxa de aprendizado e fator de desconto ajustados (pontos estacionários) para os algoritmos Q-learning e SARSA, respectivamente.

Além disso, foram realizados experimentos adotando os parâmetros utilizados em outros trabalhos:

- $\alpha = 0{,}1$ e $\gamma = 0{,}3$ (Gambardella e Dorigo, 1995; Bianchi et al., 2009),
- $\alpha = 0{,}9$ e $\gamma = 1$ (Lima Júnior et al., 2010) e
- $\alpha = 0{,}99$ e $\gamma = 0{,}01$ (Ottoni et al., 2015a).

Essas combinações também foram simuladas em cinco repetições com 10000 episódios.

Os resultados para o algoritmo SARSA são apresentados na Tabela 5.10. Nesse caso, os pontos estacionários alcançaram os melhores resultados em quatro instâncias: brazil58, br17, ftv33 e ftv44. Além disso, nas simulações dos outros dois problemas (berlin52 e kroA200), os parâmetros ajustados pela RSM obtiveram o segundo melhor desempenho entre os valores de α e γ analisados.

Tabela 5.10: Melhor solução encontrada com o algoritmo SARSA para cada problema adotando os valores dos pontos estacionários (PE) e parâmetros de outros trabalhos.

Problema	**SO**	**D95**	**L10**	**O15**	**PE**
berlin52	7542	9169	16441	8046	8048
brazil58	25395	28284	49156	27150	26685
kroA200	29368	38468	121879	33125	35311
br17	39	39	42	39	39
ftv33	1286	1533	2245	1458	1381
ftv44	1613	2033	2692	1863	1812

SO: Solução ótima conhecida da TSPLIB.
D95: solução com parâmetros de Gambardella e Dorigo (1995).
L10: solução com parâmetros de Lima Júnior et al. (2010).
O15: solução com parâmetros de Ottoni et al. (2015a).
PE: solução com parâmetros de pontos estacionários.

Já para o algoritmo Q-learning, os parâmetros ajustados pela RSM alcançaram o segundo melhor desempenho nas simulações, conforme Tabela 5.11.

Nesse caso, os pontos estacionários obtiveram rendimento inferior apenas aos parâmetros de Ottoni et al. (2015a) ($\alpha = 0{,}99$ e $\gamma = 0{,}01$). Vale ressaltar que, os parâmetros de Ottoni

Tabela 5.11: Melhor solução encontrada com o algoritmo Q-learning para cada problema adotando os valores dos pontos estacionários (PE) e parâmetros de outros trabalhos.

Problema	SO	D95	L10	O15	PE
berlin52	7542	8871	15126	8029	8619
brazil58	25395	27895	54371	26891	27487
kroA100	21282	25363	64022	24877	24925
ftv33	1286	1525	2245	1458	1464
ftv44	1613	1980	2631	1838	1873
ftv64	1839	2411	4682	2153	2279

SO: Solução ótima conhecida da TSPLIB.
D95: solução com parâmetros de Gambardella e Dorigo (1995).
L10: solução com parâmetros de Lima Júnior et al. (2010).
O15: solução com parâmetros de Ottoni et al. (2015a).
PE: solução com parâmetros de pontos estacionários.

et al. (2015a) são valores próximos aos pontos estacionários ajustados para cada problema. Além disso, pode-se perceber no gráfico de contornos para o modelo Qftv44, Figura 5.13, que o ponto estacionário ($\alpha = 0{,}7320604$ e $\gamma = 0{,}1652470$) está na mesma região que os parâmetros de Ottoni et al. (2015a). Ou seja, o ponto estacionário ajustado pode não garantir a melhor solução para a execução do AR na solução do PCV. Nesse aspecto, o gráfico de contornos oferece um aspecto visual que auxilia a definição dos parâmetros, delimitando regiões de α e γ que tendem a otimizar a solução.

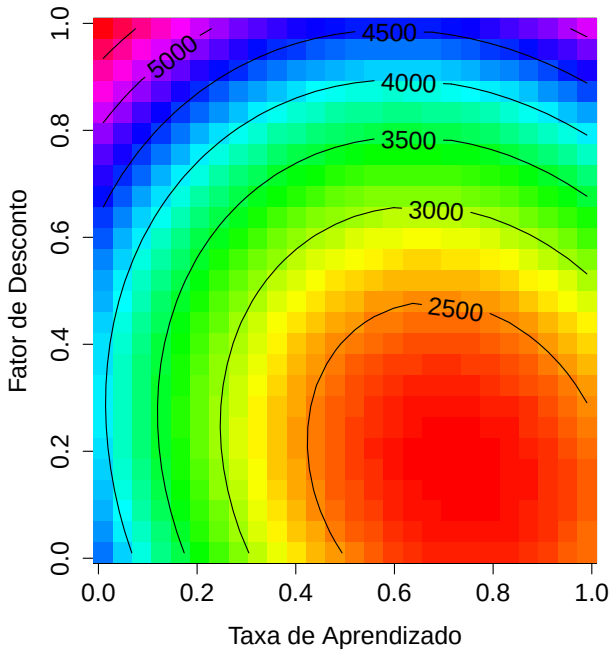


Figura 5.13: Gráficos de contornos para o modelo Qftv44 (ftv44 - Q-learning).

Assim, é importante destacar a capacidade da RSM indicar regiões com bons parâmetros. Tomando como exemplo o ponto definido pelos parâmetros utilizados por Gambardella e Dorigo (1995) ($\alpha = 0{,}1$ e $\gamma = 0{,}3$) está na região de contorno verde na Figura 5.13, longe da área vermelha, na qual é possível encontrar o ponto estacionário ($\alpha = 0{,}7320604$ e $\gamma = 0{,}1652470$). Nesse mesmo sentido, ainda analisando a Figura 5.13, o ponto definido pelos parâmetros de Lima Júnior et al. (2010) ($\alpha = 0{,}9$ e $\gamma = 1$) está na região azul do gráfico.

CAPÍTULO 6

Conclusões

Neste capítulo, são apresentadas as conclusões desta dissertação. Nesse aspecto, a primeira seção descreve as considerações finais sobre as propostas e análises do trabalho. Em seguida, são apontadas algumas possibilidades para trabalhos futuros.

6.1 Considerações Finais

O objetivo deste trabalho foi realizar uma análise de sensibilidade dos parâmetros do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante. Para isso, foram adotados os algoritmos Q-learning e SARSA. Além disso, foram realizados experimentos com instâncias do repositório TSPLIB. A metodologia proposta no trabalho compreendeu algumas análises iniciais e também estudos a partir da modelagem matemática pela Metodologia de Superfície de Resposta. Assim, foi possível avaliar qual o impacto gerado pela definição da taxa de aprendizado, fator de desconto e política $\epsilon - greedy$.

Os experimentos iniciais apontaram que a adoção do parâmetro $\epsilon = 0{,}01$ foi mais eficiente na seleção de ações pela política $\epsilon - greedy$, entre os valores analisados ($\epsilon = 0$ e $\epsilon = 0{,}1$). Além disso, a taxa de aprendizado decaindo ao longo do tempo apresentou desempenho superior à taxa de aprendizado constante.

Já os modelos de Superfície de Reposta ajustados permitem identificar como o desempenho do AR é influenciado pelos níveis dos parâmetros taxa de aprendizado e fator de desconto. Isso é possível graças ao conjunto de ferramentas disponíveis pela RSM. Os gráficos de contornos e superfície de resposta oferecem um importante aspecto visual quanto a sensibilidade do AR

aos valores de α e γ. A análise de pontos estacionários, por sua vez, permite inferir para cada modelo quais os valores dos parâmetros tendem a otimizar a resposta.

Os parâmetros ajustados pela RSM alcançaram o melhor desempenho geral nas simulações do SARSA, entre as combinações de α e γ analisadas. Já para o algoritmo Q-learning, os pontos estacionários obtiveram o segundo desempenho geral. Assim é importante ressaltar que, executar o AR para solucionar o PCV com os valores dos pontos estacionários ajustados não garante diretamente encontrar a melhor solução. Para auxiliar na definição dos parâmetros, a análise dos gráficos de superfície e contornos permite visualizar regiões próximas ao ponto estacionário que podem otimizar a solução do PCV.

Em relação ao descarte dos quatro modelos pela falta da suposição da normalidade dos resíduos, uma possível explicação está no fato de que a estrutura usada não consegue contemplar adequadamente as relações entre as variáveis para essas situações específicas.

6.2 Trabalhos Futuros

Os estudos realizados nesta dissertação apresentam possíveis vertentes para a sua sequência. Dessa forma, em seguida são levantados alguns pontos relevantes para trabalhos futuros:

- Analisar a definição de parâmetros em outras estruturas de AR, como: traços de elegibilidade (Lamperti et al., 2013b), acelerações por heurísticas (Bianchi et al., 2009), generalizações de funções (Lamperti et al., 2013a), R-learning (Schwartz, 1993) e Ant-q (Gambardella e Dorigo, 1995).
- Investigar a sensibilidade dos parâmetros do AR em outros problemas de otimização combinatória e também outros domínios tradicionais de aplicação do AR, como robótica móvel e sistemas multiagentes.
- Aprimorar a modelagem matemática pela RSM para uma função de três parâmetros: α, γ e também ϵ. Assim, adicionando os efeitos da política $\epsilon - greedy$ nos ajustes dos modelos de superfície de resposta.
- Abordar outras ferramentas para identificar a sensibilidade dos parâmetros, como modelos NARMAX (Nepomuceno e Martins, 2016). Com isso, investigar outras estruturas para tentar comtemplar os casos que a suposição de normalidade dos resíduos não foi satisfeita.

- Expandir as análises de Ottoni et al. (2015a) com experimentos com o algoritmo SARSA. Além disso, investigar os efeitos do $\epsilon - greedy$ adaptativo (Lima Júnior et al., 2010).

Referências Bibliográficas

Alipour, M. M. e Razavi, S. N. (2015). A new multiagent reinforcement learning algorithm to solve the symmetric traveling salesman problem. *Multiagent and Grid Systems*, 11(2):107–119.

Almeida, C. P. (2012). *Transgenética Computacional Aplicada a Problemas de Otimização Combinatória com Múltiplos Objetivos*. Tese, Programa de Pós-Graduação em Engenharia Elétrica e Informática, UTFPR.

Almeida, I. I. (2014). *Metaheurística Híbrida Utilizando GRASP Reativo e Aprendizagem por Reforço: Uma Aplicação na Segurança Pública.* Dissertação, Programa de Pós-Graduação em Ciência da Computação, UERN/UFRSA.

Almeida, N., Fernandes, M., e Neto, A. (2015). Beamforming and power control in sensor arrays using reinforcement learning. *Sensors (Switzerland)*, 15(3):6668–6687.

Alves, D. P., Weigang, L., e Souza, B. B. (2008). *Reinforcement Learning Book: Theory and Applications*, captulo Reinforcement learning to support meta-level control in air traffic management, pp. 409–424. I-Tech Education and Publishing.

Alves, F. A. S., Neumann, A. M. M., e Gouvêa Jr, M. M. (2014). Bengala Inteligente Neural Baseada em Aprendizagem por Reforço para Deficientes Visuais. *Brazilian Conference on Intelligent Systems / Encontro Nacional de Inteligência Artificial e Computacional 2014*.

Andrade, K. O., Fernandes, G., Caurin, G. A. P., Siqueira, A. A. G., Romero, R. A. F., e Pereira, R. L. (2014). Dynamic Player Modelling in Serious Games Applied to Rehabilitation

Robotics. In *Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), 2014 Joint Conference on*, pp. 211–216.

Applegate, D., Bixby, R. E., Chvátal, V., e Cook, W. (2007). *The Traveling Salesman Problem: A Computational Study*. Princeton University Press Princeton.

Aranibar, D. B. (2009). *Aprendizado por Reforço com Valores de Influência em Sistemas Multi-Agente*. Tese, Programa de Pós-Graduação em Engenharia Elétrica e de Computação, UFRN.

Atzori, L., Iera, A., e Morabito, G. (2010). The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805.

Bal, S. J. e Mahalik, N, P. (2014). A Simulation Study on Reinforcement Learning for Navigation Application. *Artificial Intelligence and Applications*, 1(2):43–53.

Barto, A. G., Suton, R. S., e Brouwer, P. S. (1981). Associative search network: A reinforcement learning associative memory. *Biological Cybernetics*, 40(3):201–211.

Bazzan, A. (2009). Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multi-Agent Systems*, 18(3):342–375.

Beck, C. e Srikant, R. (2012). Error bounds for constant step-size Q-learning. *Systems and Control Letters*, 61(12):1203–1208.

Benevides, P. F. (2011). *Aplicação de Heurísticas e Metaheurísticas para o Problema do Caixeiro Viajante em um Problema Real de Roteirização de Veículos*. Dissertação, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, UFPR.

Benicasa, A. X. (2012). Navegação autônoma de robôs baseada em técnicas de mapeamento e aprendizagem de máquina. *Revista Brasileira de Computação Aplicada*, 4(1):102–111.

Berlink, H. e Costa, A. H. R. (2013). Aplicação de Aprendizado por Reforço na Otimização da Venda de Energia na Geração Distribuída. In *ENIAC 2013 - X Encontro Nacional de Inteligência Artificial*.

Berlink, H., Kagan, N., e Reali Costa, A. (2015). Intelligent decision-making for smart home energy management. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 80:331–354.

Bianchi, R., Martins, M., Ribeiro, C., e Costa, A. (2014). Heuristically-Accelerated Multiagent Reinforcement Learning. *Cybernetics, IEEE Transactions on*, 44(2):252–265.

Bianchi, R. A., Ribeiro, C. H., e Costa, A. H. R. (2007). Heuristic Selection of Actions in Multiagent Reinforcement Learning. In *International Joint Conference on Artificial Intelligence*, pp. 690–695.

Bianchi, R. A. C. (2004). *Uso de Heurística para a aceleração do aprendizado por reforço.* Tese, Escola Politécnica da Universidade de São Paulo.

Bianchi, R. A. C., Ribeiro, C. H. C., e Costa, A. H. R. (2009). On the relation between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning. *1st International Workshop on Hybrid Control of Autonomous System*, pp. 49–55.

Bodin, L., Golden, B., Assad, A., e Ball, M. (1983). Routing and Scheduling of Vehicles and Crews – The State of the Art. *Computers and Operations Research*, 10(2):63–211.

Buriol, L., França, P., e Moscato, P. (2004). A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506.

Busoniu, L., Babuska, R., e De Schutter, B. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 38(2):156–172.

Campos, L. R. S., Oliveira, R. D., Melo, J. D., e Neto, A. D. D. (2012). Overheard-Controlled Routing in WSNs with Reinforcement Learning. *Intelligent Data Engineering and Automated Learning - IDEAL.*

Celiberto, L. A., Matsuura, J., e Bianchi, R. A. C. (2007). *Heuristic Q-Learning Soccer Players: A New Reinforcement Learning Approach to RoboCup Simulation*, pp. 520–529. Springer Berlin Heidelberg, Berlin, Heidelberg.

Chang, H. S. (2004). An ant system based exploration-exploitation for reinforcement learning. In *Systems, Man and Cybernetics, 2004 IEEE International Conference on*, volume 4, pp. 3805–3810.

Chen, S.-M. e Chien, C.-Y. (2011). Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. *Expert Systems with Applications*, 38(12):14439–14450.

Cline, B. E. (2004). Tuning Q-Learning Parameters with a Genetic Algorithm.

Cook, W. J. (2011). *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton University Press Princeton.

Crites, R. e Barto, A. (1996). Improving Elevator Performance Using Reinforcement Learning. In *Advances in Neural Information Processing Systems 8*, pp. 1017–1023. MIT Press.

Dabney, W. (2014). *Adaptive Step-Sizes For Reinforcement Learning*. Phd Thesis, University of Massachusetts Amherst.

Deng, Y., Liu, Y., e Zhou, D. (2015). An Improved Genetic Algorithm with Initial Population Strategy for Symmetric TSP. *Mathematical Problems in Engineering*, 2015.

Dorça, F., Lima, L.V., a. F. M., e Lopes, C. (2013). Comparing strategies for modeling students learning styles through reinforcement learning in adaptive and intelligent educational systems: An experimental analysis. *Expert Systems with Applications*, 40(6):2092–2101.

Dorigo, M. e Gambardella, L. M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Dries, E. e Peterson, G. (2008). Scaling ant colony optimization with hierarchical reinforcement learning partitioning. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pp. 25–32.

Even-Dar, E. e Mansour, Y. (2003). Learning Rates for Q-learning. *Journal of Machine Learning Research*, 5:1–25.

Fabro, J., Reis, L., e Lau, N. (2014). Using Reinforcement Learning Techniques to Select the Best Action in Setplays with Multiple Possibilities in Robocup Soccer Simulation Teams. In *Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol), 2014 Joint Conference on*, pp. 85–90.

Faria, G. e Romero, R. A. F. (2002). Navegação de Robôs Móveis Utilizando Aprendizado por Reforço e Lógica Fuzzy. *Revista Controle & Automação*, 13(3):219–230.

Ferreira, D. D., Nepomuceno, E. G., Cerqueira, A. S., e Mendes, T. M. (2016). A Model Validation Scale Based on Multiple Indices. *Electrical Engineering*, pp. 1–10.

Fiechter, C.-N. (1994). A parallel tabu search algorithm for large traveling salesman problems. *Discrete Applied Mathematics*, 51(3):243 – 267.

Gambardella, L. M. e Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. *Proceedings of the 12th International Conference on Machine Learning*, pp. 252–260.

Gatti, C. (2015). *Design of Experiments for Reinforcement Learning*. Springer International Publishing.

Goldbarg, M. C. e Luna, H. P. L. (2005). *Otimização Combinatória e Programação Linear*. Elsevier/Campus.

Gonçalves Júnior, A. M., Rocha e Silva, V. V., Baccarini, L. M. R., e Reis, M. L. F. (2014). Three-Phase Induction Motors Faults Recognition and Classification Using Neural Networks and Response Surface Models. *Journal of Control, Automation and Electrical Systems*, 25(3):330–338.

Gosavi, A. (2008). On step sizes, stochastic shortest paths, and survival probabilities in Reinforcement Learning. *Proceedings of the 40th Conference on Winter Simulation*, pp. 525–531.

Hines, W. W., Montgomery, D. C., Goldsman, D. M., e Borror, C. M. (2006). *Probabilidade e Estatística na Engenharia*. LTC.

Kaelbling, L., Littman, M., e Moore, A. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.

Khamassi, M., Lallée, S., Enel, P., Procyk, E., e Dominey, P. F. (2011). Robot cognitive control with a neurophysiologically inspired reinforcement learning model. *Frontiers in Neurorobotics*, 5(1):1–14.

Kobayashi, K., Mizoue, H., Kuremoto, T., e Obayashi, M. (2009). *A Meta-learning Method Based on Temporal Difference Error*, pp. 530–537. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kober, J., Bagnell, J. A., e Peters, J. (2013). Reinforcement Learning in Robotics: A Survey. *International Journal of Robotics Research*, July:1–38.

Kormushev, P., Calinon, S., e Caldwell, D. G. (2013). Reinforcement learning in robotics: Applications and real-world challenges. *Robotics*, 2(3):122.

Lamperti, R. D., Nepomuceno, E. G., e Ottoni, A. L. C. (2013a). Aprendizado por Reforço com Generalização de Funções. *XI SBAI - Simpósio Brasileiro de Automação Inteligente*.

Lamperti, R. D., Nepomuceno, E. G., e Ottoni, A. L. C. (2013b). Aprendizado por Reforço no Dominio do Futebol de Robôs 2D. *XI SBAI - Simpósio Brasileiro de Automação Inteligente*.

Lee, J., Bagheri, B., e Kao, H.-A. (2015). A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manufacturing Letters*, 3:18–23.

Leite, J. P. (2014). *Aplicação de Técnicas de Aprendizado por Reforço à Alocação de Recursos e ao Escalonamento de Usuários em Sistemas de Telecomunicações*. Tese, UnB - Universidade de Brasíia.

Leite, J. P., Carvalho, P. H. P., e Vieira, R. D. (2012). A flexible framework based on reinforcement learning for adaptive modulation and coding in OFDM wireless systems. *IEEE Wireless Communications and Networking Conference*.

Leng, J. e Lim, C. (2011). Reinforcement learning of competitive and cooperative skills in soccer agents. *Applied Soft Computing Journal*, 11(1):1353–1362.

Lenth, R. V. (2009). Response-Surface Methods in R, using rsm. *Journal of Statistical Software*, 32(7):1–17.

Lima Júnior, F. C. (2009). *Algoritmo Q-learning como Estratégia de Exploração e/ou Explotação para as Mataheurísticas GRASP e Algoritmo Genético*. Tese, Programa de Pós-Graduação em Eng. Elétrica e de Computação da UFRN.

Lima Júnior, F. C., Neto, A. D. D., e Melo, J. D. (2010). *Hybrid Metaheuristics Using Reinforcement Learning Applied to Salesman Traveling Problem, Traveling Salesman Problem, Theory and Applications, Prof. Donald Davendra (Ed.)*. InTech.

Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *Proceedings of the Eleventh International Conference on Machine Learning*, pp. 157–163.

Littman, M. L., Dean, T. L., e Kaelbling, L. P. (1995). On the complexity of solving markov decision problems. *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pp. 394–492.

Liu, F. e Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Systems with Applications*, 36(3):6995 – 7001.

Mariano, C. e Morales, E. (1999). MOAQ: An Ant-Q Algorithm for Multiple Objective Optimization Problems. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-99)*, pp. 894–901.

Mariano, C. e Morales, E. (2000). A New Distributed Reinforcement Learning Algorithm for Multiple Objective Optimization Problems. In Monard, M. e Sichman, J., editores, *Advances in Artificial Intelligence*, volume 1952 de *Lecture Notes in Computer Science*, pp. 290–299. Springer Berlin Heidelberg.

Marinakis, Y. e Marinaki, M. (2010). A Hybrid Multi-Swarm Particle Swarm Optimization algorithm for the Probabilistic Traveling Salesman Problem. *Computers & Operations Research*, 37(3):432 – 442.

Marinakis, Y., Marinaki, M., e Dounias, G. (2011). Honey bees mating optimization algorithm for the euclidean traveling salesman problem. *Information Sciences*, 181(20):4684–4698.

Martínez, J. M. e Santos, S. A. (1995). *Métodos Computacionais de Otimização*. IMPA.

Medeiros, T. R. (2014). *Análise e Implementação de Algoritmos para a Aprendizagem por Reforço*. Dissertação, Programa de Pós-Graduação em Informática, Universidade Federal da Paraíba.

Mendes, L. F. S., Baccarini, L. M. R., e Abreu Júnior, L. (2014). Diagnóstico de Falhas em Motores de Indução Utilizando Superfície de Resposta e Algoritmos Genéticos. *XX CBA - Congresso Brasileiro de Automática*, pp. 2946–2953.

Miagkikh, V. e Punch, W.F., I. (1999a). Global search in combinatorial optimization using reinforcement learning algorithms. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1.

Miagkikh, V. e Punch, W. (1999b). An Approach to Solving Combinatorial Optimization Problems Using a Population of Reinforcement Learning Agents. *Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation, Morgan Kaufmann*.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., e Riedmiller, M. A. (2013). Playing Atari with Deep Reinforcement Learning. *Deep Learning, Neural Information Processing Systems Workshop*, abs/1312.5602.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A., Veness, J., Bellemare, M., Graves, A., Riedmiller, M., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., e Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Monteiro, S. T. e Ribeiro, C. H. C. (2004). Desempenho de Algoritmos de Aprendizagem por Reforço sob Condições de Ambiguidade Sensorial em Robótica Móvel. *Revista Controle & Automação*, 15(3):320–338.

Murakoshi, K. e Mizuno, J. (2004). A parameter control method in reinforcement learning to rapidly follow unexpected environmental changes. *Biosystems*, 77(1-3):109 – 117.

Myers, R. H., Montgomery, D. C., e Anderson-Cook, C. M. (2009). *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons, 3 ed.

Nepomuceno, E. G. e Martins, S. A. M. (2016). A lower bound error for free-run simulation of the polynomial NARMAX. *Systems Science & Control Engineering*, 4(1):50–58.

Ng, A., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., e Liang, E. (2006). Autonomous Inverted Helicopter Flight via Reinforcement Learning. In Ang, MarceloH., J. e Khatib, O., editores, *Experimental Robotics IX*, volume 21 de *Springer Tracts in Advanced Robotics*, pp. 363–372. Springer Berlin Heidelberg.

Noda, I. (2010). Recursive Adaptation of Stepsize Parameter for Non-stationary Environments. *Lecture Notes in Computer Science*, 5924:74–90.

Oliveira, G. A. (2010). *Uma aplicação da aprendizagem por reforço na otimização da produção em um campo de petróleo*. Dissertação, Programa de Pós-Graduação em Ciência e Engenharia do Petróleo, UFRN.

Ottoni, A. L. C., Barbosa, A. M., Nepomuceno, E. G., e Oliveira, M. S. (2016a). Controle de Epidemias com Aprendizado por Reforço: Estratégia de Combate ao Aedes aegypti. *Encontro Nacional de Modelagem Computacional 2016*.

Ottoni, A. L. C., Lamperti, R. D., Nepomuceno, E. G., e Oliveira, M. S. (2012). Desenvolvimento de um sistema de aprendizado por reforço para times de robôs - Uma análise de desempenho por meio de testes estatísticos. *XIX Congresso Brasileiro de Automática*, pp. 3557–3564.

Ottoni, A. L. C., Nepomuceno, E. G., Cordeiro, L. T., Lamperti, R. D., e Oliveira, M. S. (2015a). Análise do Desempenho do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante. *XII SBAI - Simpósio Brasileiro de Automação Inteligente*, pp. 43–48.

Ottoni, A. L. C., Nepomuceno, E. G., e Oliveira, M. S. (2016b). Análise de Sensibilidade dos Parâmetros do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante: Modelagem via Superfície de Resposta. *XXI Congresso Brasileiro de Automática - CBA 2016*, pp. 513–518.

Ottoni, A. L. C., Nepomuceno, E. G., e Oliveira, M. S. (2016c). Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante Assimétrico: Uma Compração entre os Algoritmos Q-learning e SARSA. *XII Simpósio de Mecânica Computacional*, 1:87–94.

Ottoni, A. L. C., Nepomuceno, E. G., Oliveira, M. S., Cordeiro, L. T., e Lamperti, R. D. (2016d). Análise da influência da taxa de aprendizado e do fator de desconto sobre o desempenho

dos algoritmos Q-learning e SARSA: aplicação do aprendizado por reforço na navegação autônoma. *Revista Brasileira de Computação Aplicada*, 8(2):44–59.

Ottoni, A. L. C., Nepomuceno, E. G., Oliveira, M. S., e Felix, L. B. (2016e). Modelo Híbrido de Aprendizado por Reforço e Lógica Fuzzy Aplicado ao Problema do Caixeiro Viajante com Reabastecimento. *XXI Congresso Brasileiro de Automática - CBA 2016*, pp. 1–6.

Ottoni, A. L. C. e Oliveira, M. S. (2014). Regressão Logística Aplicada na Análise do Aprendizado por Reforço. *21º SINAPE - Simpósio Nacional de Probabilidade e Estatística*.

Ottoni, A. L. C., Oliveira, M. S., Nepomuceno, E. G., e Lamperti, R. D. (2015b). Análise do Aprendizado por Reforço via Modelos de Regressão Logística: Um Estudo de Caso no Futebol de Robôs. *Revista Junior de Iniciação Científica em Ciências Exatas e Engenharia*, 1(10):44–49.

Ouaarab, A., Ahiod, B., e Yang, X.-S. (2014). Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Computing and Applications*, 24(7-8):1659–1669.

Ozguner, U., Stiller, C., e Redmill, K. (2007). Systems for safety and autonomous behavior in cars: The DARPA grand challenge experience. *Proceedings of the IEEE*, 95(2):397–412.

Pedro, O. R. (2013). *Uma abordagem de Busca Tabu para o Problema do Caixeiro Viajante com Coleta de Prêmios*. Programa de Pós-Graduação em Engenharia Elétrica, UFMG.

Pellegrini, J. e Wainer, J. (2007). Processos de Decisão de Markov: um tutorial. *RITA - Revista de Informática Teórica e Aplicada*, 14(2):133–179.

Prauchner, D. C. K., Martins, R. S. M., e Padoin, E. L. (2014). Aplicação do algoritmo sarsa à coleta de lixo. In *ENIAC 2014 - XI Encontro Nacional de Inteligência Artificial e Computacional*.

Puterman, M. L. (1994). Markov decision processes: Discrete stochastic dynamic programming. *New York, NY: Wiley-Interscience*.

Qi, D. e Sun, R. (2005). Learning to Cooperate in Solving the Traveling Salesman Problem. *International Journal of Neural Systems*, 15(01n02):151–162.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Razali, N. M. e Wah, Y. B. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of Statistical Modeling and Analytics*, 2(1):21–33.

Reinelt, G. (1991). TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4):376–384.

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Springer-Verlag, Berlin, Heidelberg.

Reinelt, G. (1995). *TSPLIB95*. Institut f¨ur Angewandte Mathematik, Universitat Heidelberg.

Rezende, S. O. (2003). *Sistemas Inteligentes: Fundamentos e Aplicações*. Editora Manole Ltda.

Ribeiro, C. (2002). Reinforcement learning agents. *Artificial Intelligence Review*, 17(3):223–250.

Riveret, R., Artikis, A., Pitt, J., e Nepomuceno, E. G. (2014). Self-Governance by Transfiguration: From Learning to Prescription Changes. In *Self-Adaptive and Self-Organizing Systems (SASO), 2014 IEEE Eighth International Conference on*, pp. 70–79.

Romero, R. A. F., Prestes, E., Osório, F., e Wolf, D. (2014). *Robótica Móvel*. LTC.

Russell, S. J. e Norving, P. (2013). *Inteligência Artificial*. Campus, 3st ed.

Ryzhov, I. O., Frazier, P. I., e B., P. W. (2015). A New Optimal Stepsize for Approximate Dynamic Programming. *IEEE Transactions on Automatic Control*, 60:743–757.

Santos, J. L. e Nascimento Júnior, C. L. (2012). Coordenação dos atuadores das pernas de robôs móveis usando aprendizado por reforçoo: simulação e implementação. *Revista Controle & Automação*, 23(1):78 – 93.

Santos, J. P. Q., Melo, J. D., Duarte Neto, A. D., e Aloise, D. (2014). Reactive Search strategies using Reinforcement Learning, local search algorithms and Variable Neighborhood Search. *Expert Systems with Applications*, 41(10):4939–4949.

Santos, J. Q., Lima Junior, F., Magalhaes, R., de Melo, J., e Neto, A. (2009). A parallel hybrid implementation using genetic algorithm, GRASP and reinforcement learning. In *Neural Networks, 2009. IJCNN 2009. International Joint Conference on*, pp. 2798–2803.

Schwartz, A. (1993). A reinforcement learning method for maximizing undiscounted rewards. *Proceedings of the Tenth International Conference on Machine Learning*, pp. 298–305.

Schweighofer, N. e Doya, K. (2003). Meta-learning in reinforcement learning. *Neural Networks*, 16(1):5–9.

Scárdua, L. A., Cruz, J. J., e Costa, A. H. R. (2003). Controle Ótimo de Descarregadores de Navios Utilizando Aprendizado por Reforço. *Revista Controle & Automação*, 14(4):368–376.

Selvatici, A. H. P. e Costa, A. H. R. (2007). Aprendizado da coordenação de comportamentos primitivos para robôs móveis. *Revista Controle & Automação*, 18(2):173–186.

Silva, A. L. M. (2014). *Algoritmo Baseado em Evolução Diferencial Para Solução de Problemas de Otimização Combinatória*. Programa de Pós-Graduação em Engenharia Elétrica, UFMG.

Silva, F. L. S., Costa, G. M., e Bastos, G. S. (2013). Semáforo Inteligente - Integração de Aprendizagem por Reforço e a Técnica CMAC. *XI SBAI - Simpósio Brasileiro de Automação Inteligente*.

Silva, I. J., Perico, D. H., Homem, T. P. D., Vilao, C. O., Tonidandel, F., e Bianchi, R. A. C. (2015). Using Reinforcement Learning to Improve the Stability of a Humanoid Robot: Walking on Sloped Terrain. In *Robotics Symposium (LARS) and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR), 2015 12th Latin American*, pp. 210–215.

Silva, I. N., Spatti, D. H., e Flauzino, R. A. (2010). *Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas*. ArtLiber.

Siqueira, P., Steiner, M., e Scheer, S. (2007). A new approach to solve the traveling salesman problem. *Neurocomputing*, 70(4-6):1013–1021.

Stone, P., Sutton, R. S., e Kuhlmann, G. (2005). Reinforcement Learning for RoboCup-Soccer Keepaway. *Adaptive Behavior*, 13(3):165–188.

Stone, P. e Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383.

Sun, R., Tatsumi, S., e Zhao, G. (2001). Multiagent reinforcement learning method with an improved ant colony system. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, volume 3, pp. 1612–1617.

Sutton, R. e Barto, A. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1st edio.

Sutton, R. S. (1984). *Temporal Credit Assignment in Reinforcement Learning*. Tese, University of Massachusetts, Amherst.

Sutton, R. S. (1988). Learning to predict by the method of temporal ddifferences. *Machine Learning*, 3:9–44.

Teixeira, H. T. e Bottura, C. P. (2016). Aplicação de máquinas de vetor de suporte on-line em algoritmos de interação de política para aprendizado por reforço. *XXI Congresso Brasileiro de Automática - CBA 2016*, pp. 3091–3096.

Tokic, M., Schwenker, F., e Palm, G. (2013). Meta-learning of exploration and exploitation parameters with replacing eligibility traces. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8183 LNAI:68–79.

Vitor, A. (2015). *Uma Proposta de Algoritmo Genético Híbrido para o Problema do Caixeiro Via*. Tese, Programa de Pós-Graduação em Métodos Numéricos em Engenharia, UFPR.

Watkins, C. J. (1989). *Models of Delayed Reinforcement Learning*. Tese, Psychology Department, Cambridge University, Cambridge, United Kingdom.

Watkins, C. J. e Dayan, P. (1992). Technical note Q-learning. *Machine Learning*, 8(3):279–292.

White, D. J. (1993). A survey of applications of Markov decision processes. *The Journal of the Operational Research Society*, 44(11):1073–1096.

Wiering, M. e van Otterlo, M. (2012). *Reinforcement Learning: State-of-the-art*. Springer-Verlag Berlin Heidelberg.

Yoshida, N., Uchibe, E., e Doya, K. (2013). Reinforcement learning with state-dependent discount factor. In *Development and Learning and Epigenetic Robotics (ICDL), 2013 IEEE Third Joint International Conference on*, pp. 1–6.

Yu, T. e Zhen, W.-G. (2009). A reinforcement learning approach to power system stabilizer. In *2009 IEEE Power and Energy Society General Meeting, PES '09*.