UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RICARDO GRUNITZKI

Aprendizado por Reforço Multiagente: Uma Avaliação de Diferentes Mecanismos de Recompensa para o Problema de Aprendizado de Rotas

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Orientador: Prof[a]. Dra. Ana L. C. Bazzan

Porto Alegre
Julho de 2014

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Grunitzki, Ricardo

Aprendizado por Reforço Multiagente: Uma Avaliação de Diferentes Mecanismos de Recompensa para o Problema de Aprendizado de Rotas / Ricardo Grunitzki. – Porto Alegre: PPGC da UFRGS, 2014.

77 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2014. Orientador: Ana L. C. Bazzan.

1. Aprendizado por reforço multiagente. 2. Função de recompensa. 3. Difference rewards. 4. Sistemas multiagente. I. Bazzan, Ana L. C.. II. Título.

"Quanto mais aumenta nosso conhecimento,
mais evidente fica nossa ignorância."
— John F. Kennedy

AGRADECIMENTOS

Agradeço primeiramente a minha família, especialmente aos meus pais Marco Antônio Grunitzki e Claudete Dutra Grunitzki, por terem me fornecido todo o apoio necessário e também por compreenderem a minha ausência durante o desenvolvimento deste trabalho. Saibam que sem vocês esta dissertação não seria possível.

Agradeço imensamente a minha orientadora Ana L. C. Bazzan, por ter acreditado no meu potencial. Além dos ensinamentos passados, nunca mediu esforços para estimular a mim e ao grupo a sempre dar o nosso melhor no trabalho. Além de excelente orientadora, é um grande exemplo de paixão pela ciência.

Não poderia deixar de agradecer meus colegas de trabalho do MASLAB, especialmente ao Gabriel, Jorge e Mariana, os quais me auxiliaram em importantes decisões desta dissertação. O ambiente de trabalho proporcionado pelo grupo, com constante troca de conhecimento e experiência, foi fundamental para o meu crescimento pessoal e profissional. Muito obrigado, MASLAB.

Aos meus colegas da UFRGS, os quais evito citar nomes para evitar que, por ventura, esqueça de alguém. São pessoas com as quais tive a oportunidade de conhecer e admirar durante o curso do mestrado. Como é de se esperar, sempre tivemos pensamentos e opiniões distintas sobre os mais diversos assuntos. Entretanto, tal divergência de opiniões, por muitas vezes, foi o combustível de muitos cafés ou almoços nos RU. Obrigado a todos vocês.

Meu muito obrigado aos meus amigos de república Anderson, Sérgio, Pedro, Alex e Marcelo. Vocês foram responsáveis por tornar meus dias em Porto Alegre muito mais agradáveis. Talvez por estarem, assim como eu, ligados ao meio acadêmico, sempre apresentaram contribuições significantes a este trabalho. Ainda neste grupo de amigos, não poderia deixar de agradecer a outros dois grandes amigos: Mattew e Eric. Juntos, proporcionaram momentos e recordações que jamais esquecerei.

Aos meus amigos de Ibirama, em especial ao Fernando e Jarbas, da UDESC. Pessoas que me motivaram e tornaram possível meu ingresso na vida acadêmica.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, pelo apoio financeiro. Também não poderia deixar de agradecer ao Instituto de Informática da Universidade Federal do Rio Grande do Sul, pela oportunidade de estudar em um programa de pós-graduação que é referência internacional em qualidade.

Por último, mas não menos importante, eu gostaria de fazer um agradecimento especial a Silvia Roberta Geherke. Agradeço por sua imensa paciência e compreensão ao longo dos últimos anos.

Multiagent Reinforcement Learning: An Evaluation of Different Reward Mechanisms for the Route Learning Problem

ABSTRACT

This dissertation presents a study on the effects of different reward functions applyed to multiagent reinforcement learning, for the vehicles routing problem, in traffic networks. Two reward functions that differ in the alignment of the numerical signal sent from the environment to the agent are addressed. The first function, called individual function is aligned with the agent's (vehicle or driver) utility and seeks to minimize their travel time. The second function, is called difference rewards and is aligned to the system's utility and aims to minimize the average travel time on the network (average travel time of all drivers). Both approaches are applied to two routing vehicles' problems, which differ in the number of learning drivers, network topology and therefore, level of complexity. These approaches are compared with three traffic assignment techniques from the literature. Results show that reinforcement learning-based methods yield superior results than traffic assignment methods. Furthermore, the reward function alignment to the global utility, provides a significant improvement in results when compared with the individual function. However, for scenarios with many agents learning simultaneously, both approaches yield equivalent solutions.

Keywords: Multiagent reinforcement learning, reward function, difference rewards, multiagent systems.

RESUMO

Esta dissertação de mestrado apresenta um estudo sobre os efeitos de diferentes funções de recompensa, aplicadas em aprendizado por reforço multiagente, para o problema de roteamento de veículos, em redes de tráfego. São abordadas duas funções de recompensas que diferem no alinhamento do sinal numérico enviado do ambiente ao agente. A primeira função, chamada função individual, é alinhada à utilidade individual do agente (veículo ou motorista) e busca minimizar seu tempo de viagem. Já a segunda função, por sua vez, é a chamada *difference rewards*, essa é alinhada à utilidade global do sistema e tem por objetivo minimizar o tempo médio de viagem na rede (tempo médio de viagem de todos os motoristas). Ambas as abordagens são aplicadas em dois cenários de roteamento de veículos que diferem em: quantidade de motoristas aprendendo, topologia e, consequentemente, nível de complexidade. As abordagens são comparadas com três técnicas de alocação de tráfego presentes na literatura. Resultados apontam que os métodos baseados em aprendizado por reforço apresentam desempenho superior aos métodos de alocação de rotas. Além disso, o alinhamento da função de recompensa à utilidade global proporciona uma melhora significativa nos resultados quando comparados com a função individual. Porém, para o cenário com maior quantidade de agentes aprendendo simultaneamente, ambas as abordagens apresentam soluções equivalentes.

Palavras-chave: Aprendizado por reforço multiagente, função de recompensa, difference rewards, sistemas multiagente.

LISTA DE ABREVIATURAS E SIGLAS

DP	Programação Dinâmica
JA	Ação Conjunta
JAL	*Joint-Action Learner*
MARL	Aprendizado por Reforço Multiagente
SMA	Sistema Multiagente
MDP	Processo de Decisão de Markov
MMDP	Processo de Decisão de Markov Multiagente
POMDP	Processo de Decisão de Markov Parcialmente Observável
OD	Origem-Destino
RL	Aprendizado por Reforço
TD	Diferença Temporal
TA	Alocação de Tráfego
VDF	*Volume-Delay Function*

LISTA DE SÍMBOLOS

A	Conjunto de ações do agente
a	Determinada ação de um agente
a'	Determinada ação futura de um agente
C_j	Capacidade do link j para a VDF
D_i	Recompensa da função *difference rewards* para o agente i
D	Conjunto de agentes/motoristas
$D^{\bullet}$	Conjunto de agentes/motoristas que ainda não chegaram ao destino
$G(z)$	Total de recompensa que os agentes do sistema ao executar as as ações em z
$G(z_{-d})$	Total de recompensa que os agentes receberiam do sistema ao executar as ações em z_{-d}, sem a presença do agente d no sistema
I	Zonas de origem da matriz OD
J	Zonas de destino da matriz OD
L	Conjunto de links da rede viária
$n_d^{\uparrow}$	nó origem do veículo d
$n_d^{\downarrow}$	nó destino do veículo d
O	Probabilidade de observação do ambiente
P	Conjunto de agentes
Q	Valor de um determinado par estado-ação
R	Função de recompensa
r_d	Rota do agente d
S	Conjunto de estados do ambiente
s	Determinado estado do ambiente
s'	Determinado estado futuro do ambiente
T	Matriz origem destino
t_j	Tempo de viagem no link j
t_{j_0}	Tempo de viagem sem fluxo no link j
V_k	Volume no link j

y_d	Custo de viagem do agente d
z	Representa os pares estado-ação de todos os agentes do sistema
z_{-d}	Representa os pares estado-ação de todos os agentes do sistema menos o do agente d
$\mathcal{A}_j$	Parâmetro da VDF para o link j
$\mathcal{B}_j$	Parâmetro da VDF para o link j
α	Taxa de aprendizagem
γ	Fator de desconto
ϵ	Taxa de exploração
ϵ_0	Taxa de exploração inicial
ϵ_f	Fator de desconto da exploração
Ω	Espaço de observações
Π	Conjunto de políticas
π	Determinada política de transição de estados
π^*	Política ótima de transição de estados
Φ	Função potencial do *potential-based reward shaping*
τ	Número máximo de passos de tempo da simulação
μ	Número máximo de episódios da simulação
V^π	Expectativa de recompensa a ser recebida por seguir determinada política

LISTA DE FIGURAS

2.1 Estrutura de um sistema multiagente. As esferas de influência são representadas por linhas pontilhadas. 17
2.2 Interação agente-ambiente. 19
2.3 Um exemplo de paradoxo de Braess. 31
2.4 Um exemplo de rede viária e sua correspondente representação em forma de grafo. 32
2.5 Comportamento da VDF para o link 22-23 da rede Sioux Falls. 34

4.1 Mapeamento dos estados (S) e ações (A) do agente 47

5.1 Topologia da rede OW adaptada 55
5.2 Tempo de convergência e desvio padrão ao longo dos episódios (principal) e ao final dos episódios (interno) das abordagens propostas na rede OW. 60
5.3 Intervalo de confiança da curva de diferença dos métodos ao longo dos episódios (principal) e nos episódios finais (interno) para a rede OW. . 61
5.4 Frequência de tempo percentual para grupos de 5 episódios em que DQ-learning é estatisticamente melhor do que o IQ-learning na rede OW. 62
5.5 Topologia da rede sioux falls. 63
5.6 Tempo de convergência e desvio padrão ao longo dos episódios (principal) e ao final dos episódios (interno) das abordagens propostas na rede sioux falls. 65
5.7 Intervalo de confiança da curva de diferença dos métodos ao longo dos episódios (principal) e nos episódios finais (interno) para a rede OW. . 66
5.8 Frequência de tempo percentual para grupos de 5 episódios em que IQ-learning é estatisticamente melhor que o DQ-learning na rede sioux falls. 66

LISTA DE TABELAS

3.1 Aspectos desta dissertação e trabalhos relacionados a RL para o problema de escolha de rotas. 42

5.1 Pares OD da rede OW. 56
5.2 Menor caminho e custo sem fluxo para os 4 pares OD da rede OW (versão original e modificada). 56
5.3 Tempo Médio de Viagem (TMV) e tempo médio por par OD na rede OW. Valores entre parênteses representam o desvio padrão. 57
5.4 Tempo de viagem e fluxo em cada link para os métodos DQ-learning, IQ-learning, *incremental assingment*, *successive averages* e *all-or-nothing*, na rede OW. O tempo de viagem sem fluxo é representado por c.s.f. e o percentual de links utilizados (com fluxo > 0) é representado por p.l.u. 59
5.5 Tempo médio de viagem na rede (TMV) e tempo médio por par OD em minutos na rede sioux falls. Valores entre parênteses representam o desvio padrão. 64

6.1 Matriz OD da rede sioux falls. As linhas representam as zonas de origem e as colunas as zonas de destino. 77

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Organização dos Capítulos 16

2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 Agentes Autônomos e Sistemas Multiagente 17
2.2 Aprendizado por Reforço Monoagente 18
2.2.1 Processo de Decisão de Markov 20
2.2.2 Principais Métodos de RL 22
2.3 Aprendizado por Reforço Multiagente 24
2.3.1 Processo de Decisão de Markov Multiagente 25
2.3.2 Equilíbrio de Nash 26
2.3.3 Ótimo de Pareto 27
2.3.4 Principais Métodos de MARL 27
2.3.5 Funções de Recompensa 29
2.3.6 Difference Rewards 30
2.4 Oferta e Demanda em Sistemas de Transporte 32
2.4.1 Alocação de Tráfego 33
2.4.2 Métodos de Alocação de Tráfego 34
2.5 Simulação de Tráfego 37

3 TRABALHOS RELACIONADOS 40
3.1 Abordagens de Aprendizado por Reforço para Escolha de Rotas . . 40
3.2 Abordagens de Aprendizado por Reforço para Outros Problemas . 42
3.3 Abordagens Alternativas para Escolha de Rotas 44

4 ABORDAGEM PROPOSTA 45
4.1 Modelagem dos Agentes 46
4.1.1 Funções de Recompensa 48
4.2 Simulação de Aprendizado de Rotas 50

5 EXPERIMENTOS 54
5.1 Parâmetros e Métodos Utilizados na Comparação 54
5.2 Cenário OW: 1700 Agentes 55
5.2.1 Resultados Experimentais 56
5.3 Cenário Sioux Falls: 360600 Agentes 61
5.3.1 Resultados Experimentais 62

5.4 Resumo dos Experimentos 67

6 CONSIDERAÇÕES FINAIS 69
6.1 Contribuições 70
6.2 Perspectivas de Continuação 70

REFERÊNCIAS 72

APÊNDICE A - MATRIZ OD DA REDE SIOUX FALLS 77

1 INTRODUÇÃO

Os sistemas multiagente formam uma subárea da inteligência artificial distribuída que, ao contrário da inteligência artificial tradicional, busca dividir um problema em problemas menores e mais simples de serem resolvidos. Embora exista uma grande variedade de aplicações para sistemas multiagente, pode-se destacar os problemas em que se faz necessário aprender um comportamento de forma *online*, para se atingir os objetivos de projeto. Neste caso, denominado problema de aprendizado por reforço (RL, em inglês: *reinforcement learning*), o agente aprende um comportamento através de sua interação repetida com o ambiente.

No RL, quando o problema possui múltiplos agentes aprendendo e interagindo em um ambiente em comum, a complexidade da tarefa de aprendizagem aumenta consideravelmente. Nestas condições, o desempenho individual do agente é influenciado por suas ações individuais e pelas consequências das ações dos demais agentes. Além disso, características do ambiente como: observável ou não observável, estático ou dinâmico, discreto ou contínuo e assim por diante contribuem para o aumento dessa complexidade.

Apesar de existir uma série de algoritmos desenvolvidos para realizar tarefas de aprendizado por reforço multiagente (MARL, do inglês *multiagent reinforcement learning*), técnicas como as que consideram o uso de ações conjuntas no processo de tomada de decisão individual, necessitam de um número ainda maior de experiências para aprender uma política de ação, devido à explosão do espaço de busca. Além disso, as garantias de convergência para solução ótima, existentes em problemas monoagente, não se aplicam na presença de múltiplos agentes. Para evitar estes problemas de escalabilidade, os agentes geralmente são implementados como aprendizes independentes cujo processo decisório desconsidera a presença dos demais agentes.

Tanto em problemas mono quanto multiagente o processo de aprendizagem é guiado por uma função de recompensa cuja responsabilidade é incentivar as ações desejáveis e acanhar as indesejáveis. Independente da técnica de RL a ser utilizada, a qualidade da solução emergente está diretamente relacionada à função de recompensa. A modelagem de funções de recompensa é um campo de pesquisa que tem atraído o interesse de muitos pesquisadores. Da forma em que estas funções geralmente são modeladas, o RL busca maximizar a utilidade individual dos agentes. Contudo, trabalhos envolvendo conceitos de inteligência coletiva (TUMER; WOLPERT, 2004, 2000) evidenciam que, mesmo em problemas onde não há uma cooperação implícita entre os agentes, o desempenho de um coletivo pode ser maximizado, caso a função de recompensa esteja alinhada à utilidade do sistema.

Problemas como o de roteamento de veículos em redes de tráfego podem ser modelados por meio de RL. Nestes cenários, um número de veículos (agentes) deve aprender a melhor rota entre sua origem e destino. A qualidade da rota é avaliada por meio de uma função

de custo que, em geral, pode ser o tempo de viagem. Nesta tarefa, os agentes se deparam com problemas como: número crescente de estados e ações devido às dimensões da rede de tráfego, ambiente parcialmente observável (os agentes desconhecem a utilização atual dos demais links da rede) e dinamicidade do ambiente, devido à presença de múltiplos agentes agindo de forma autônoma.

Muitas técnicas alternativas são propostas para resolver o problema de roteamento de veículos. Entretanto, técnicas como as de otimização (BURIOL et al., 2010; CAGARA; BAZZAN; SCHEUERMANN, 2014) desconsideram fatores como os mencionados anteriormente. Sendo assim, a modelagem e simulação baseada em agentes são ferramentas robustas para desenvolver soluções que se aproximem do mundo real. O conceito de agente frisa uma característica muito importante a qual está presente no contexto de roteamento de veículos do mundo real, a autonomia: embora os motoristas possam ser auxiliados por sistemas de navegação, a decisão do caminho a ser seguido é tomada individualmente. Esta característica do cenário real é muito bem reproduzida em simulação baseada em agentes, já que o processo de tomada de decisão dos agentes é realizado de forma autônoma e individual. Além disso, a granularidade - e, consequentemente, a precisão - das informações presentes na simulação são variáveis em função das necessidades do problema.

Nesta dissertação, o problema de roteamento de veículos é abordado como um problema de aprendizado por reforço multiagente. Duas funções de recompensa são utilizadas para guiar o processo de aprendizagem. A primeira, chamada "função individual", está alinhada ao objetivo individual do agente (i.e., minimizar o seu tempo de viagem). Já a segunda, chamada "*difference rewards*", utiliza a abordagem para modelagem de função de recompensa apresentada por (TUMER; AGOGINO, 2007), e está alinhada ao objetivo global do sistema (i.e. minimizar o tempo médio de viagem na rede). O objetivo desta dissertação é verificar se o alinhamento da função de recompensa influencia no desempenho global do sistema, para diferentes quantidades de agentes aprendendo, em cenários de roteamento de veículos. São utilizados dois problemas de roteamento que diferem em nível de complexidade. O primeiro cenário é representado por uma rede abstrata, com 1700 agentes e poucos estados e ações. Já o segundo, possui 360600 agentes aprendendo, além de um número superior de estados e ações. Os resultados obtidos são comparados entre si e com métodos conhecidos de alocação de tráfego presentes na literatura, como *successive averages assignment* e *incremental assignment*.

Os resultados experimentais demonstram que as abordagens baseadas em agentes apresentam resultados superiores aos métodos de alocação de tráfego utilizados na comparação. Em função dos agentes realizarem a escolha de rotas de forma inteligênte, os agentes são capazes de adaptar suas rotas durante a viagem, no intuito de utilizar caminhos alternativos e menos congestionadas que, consequentemente, podem resultar em um menor custo de viagem. Além disso, o alinhamento da função de recompensa à utilidade do sistema apresentou resultados superiores no primeiro cenário, porém equivalentes para o

segundo cenário, quando comparado ao alinhamento da função de recompensa à utilidade do agente.

Embora esta dissertação aplique uma série de técnicas já existentes na literatura para resolver o problema de aprendizado de rotas, ela trás algumas contribuições muito importantes para o avanço da ciência nesta linha. Primeiramente, é o trabalho que se tem conhecimento que aplica a maior quantidade de agentes aprendendo por reforço de forma simultânea em um cenário complexo e dinâmico como o de aprendizado de rotas. Em segundo lugar, pode-se destacar a descoberta de que os benefícios da função *difference rewards* podem ser sensíveis a quantidade de agentes aprendendo.

1.1 Organização dos Capítulos

Os próximos capítulos desta dissertação estão organizados da seguinte maneira:

- Capítulo 2: Apresenta a fundamentação teórica necessária para dar entendimento a este trabalho. São abordados assuntos relacionados aos conceitos básicos de agente e sistemas multiagente (Seção 2.1), aprendizado por reforço (Seção 2.2 e Seção 2.3), oferta e demanda em redes de tráfego (Seção 2.4) e modelos de simulação de tráfego (Seção 2.5).
- Capítulo 3: Apresenta os principais trabalhos relacionados ao problema de roteamento de veículos.
- Capítulo 4: Apresenta a modelagem dos agentes e função de recompensa, bem como o modelo de simulação de tráfego utilizado.
- Capítulo 5: Apresenta os experimentos realizados e cenários utilizados para avaliar o desempenho dos métodos propostos.
- Capítulo 6: Apresenta as conclusões, contribuições e perspectivas de continuação deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Agentes Autônomos e Sistemas Multiagente

Embora não exista uma definição formal para o termo "agente" aceita por toda a comunidade, existem muitos debates e controvérsias a respeito das diversas definições apresentadas na literatura. Apesar disso, as diferentes definições têm o termo "autonomia" como uma noção central de agente. (WOOLDRIDGE, 2009) atribui esta dificuldade de definição ao fato de que os vários atributos do agente são de diferentes importâncias e mudam em função do domínio. Fato este que é totalmente compreensível, já que, em algumas aplicações, a habilidade de aprender um comportamento a partir de experiências é de fundamental importância, porém, para outras, aprendizado é indesejável. Diante deste cenário, este trabalho adota o seguinte conceito de agentes:

> Agente é um sistema computacional, que está situado em algum ambiente, e é capaz de agir de forma autônoma neste ambiente para atingir os objetivos para o qual foi projetado (WOOLDRIDGE, 2009).

Um sistema multiagente (SMA), de acordo com (WOOLDRIDGE, 2009), é composto por múltiplos agentes que interagem entre si. Cada agente atua sobre o ambiente e possui, portanto, uma "esfera" de influência sobre o mesmo. As esferas de influência de diferentes agentes podem se sobrepor e isso pode gerar relações entre os agentes.

A estrutura típica de um sistema multiagente é ilustrada pela Fig. 2.1. Como pode ser observado, existe uma intersecção entre as esferas de influências dos agentes $a2$ e $a3$. Esta intersecção pode ser analogamente comparada a uma tarefa onde, por exemplo, dois robôs precisam passar por uma mesma porta, cujas dimensões só permitem a passagem de um robô por vez. Para que esta tarefa seja realizada com sucesso, deve haver uma coordenação entre os agentes, caso contrário os agentes poderão interferir no ambiente de modo que ambos não consigam realizar sua tarefa.

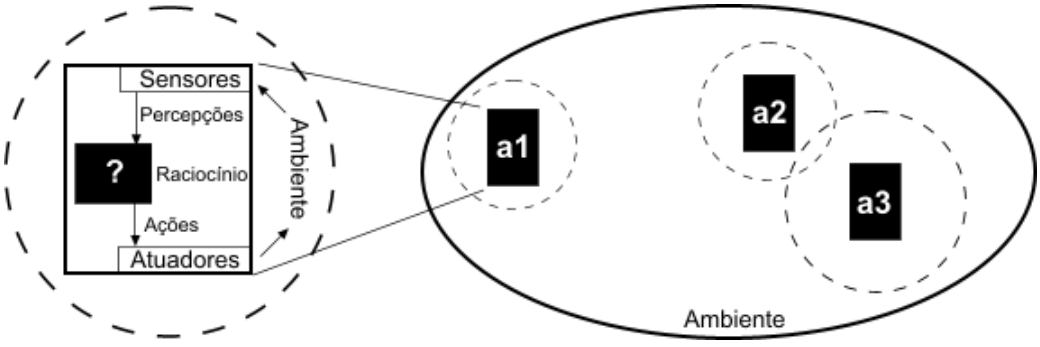


Figura 2.1: Estrutura de um sistema multiagente. As esferas de influência são representadas por linhas pontilhadas (TAVARES, 2013).

Para uma leitura mais aprofundada sobre agentes autônomos e sistemas multiagente recomenda-se a leitura de (WOOLDRIDGE, 2009).

2.2 Aprendizado por Reforço Monoagente

O aprendizado por reforço tem suas raízes no aprendizado animal. É muito comum fazer um cão responder com um comportamento desejado punindo-o ou recompensando-o apropriadamente. Um exemplo bem conhecido disto é o treinamento de cães para encontrar vítimas de crimes ou desastres naturais como terremotos. Para que isto seja possível, os cães são treinados em simulações para encontrar objetos com odores específicos. Sempre que eles encontram um objeto são recompensados com comida. Com base nesta recompensa, o cão adapta seu aprendizado gradativamente para encontrar o objeto (vítima, drogas e etc.) que retorna a maior recompensa. Para garantir que a solução seja encontrada, o RL recompensa os comportamentos desejados e pune os indesejados, provocando assim uma mudança comportamental. De modo geral, o objetivo do RL é descobrir uma política de ação, através do mapeamento de estados e ações que maximize o reforço total recebido (TUYLS; WEISS, 2012).

Do ponto de vista da ciência da computação, o RL é um tipo de aprendizagem de máquina. Aprendizagem de máquina é o processo pelo qual o computador aprende a aperfeiçoar o seu desempenho a partir de exemplos ou observações. Dois outros tipos muito comuns de aprendizagem de máquina são: supervisionado e não supervisionado (MITCHELL, 1997).

No aprendizado supervisionado existe a necessidade de um especialista de domínio para rotular tipos de exemplos, dos quais os algoritmos de aprendizado supervisionado podem identificar padrões e aprender como identificar novos exemplos. O aprendizado não supervisionado identifica padrões em dados e os agrupa em grupos de dados semelhantes, sem a necessidade de um especialista. Já RL encontra-se entre o aprendizado supervisionado e não supervisionado. No RL não há a necessidade de um especialista de domínio para explicitar se uma ação tomada é certa ou errada. Entretanto, utiliza-se uma entrada recebida do ambiente como reforço para o que acredita ser a melhor forma de agir (MITCHELL, 1997).

Por meio do RL, o agente aprende uma política de ação para realizar uma tarefa, através das suas interações com o ambiente. No RL monoagente, isto é, onde só existe um agente aprendendo, considera-se ambiente tudo além do agente (SUTTON; BARTO, 1998). O agente é a entidade que aprende e toma decisões no ambiente de forma autônoma. Em outras palavras, é um programa capaz de tomar decisões, com base em suas próprias motivações de como se comportar em determinada situação. Além disso, um agente que aprende por reforço, aprende como satisfazer suas próprias motivações, através de suas experiências passadas no ambiente.

Para ganhar experiência o agente interage repetidas vezes com o ambiente, como ilustrado pela Fig. 2.2. A interação é um ciclo, iniciando com o ambiente que apresenta a sua situação atual ao agente, na forma de uma representação de estado. O agente, então, escolhe, a partir de um conjunto de ações dadas, o que fazer neste estado. A ação que o agente irá realizar causará algum efeito no ambiente e, portanto, pode alterar o estado atual. Em seguida, o ambiente retorna o novo estado e uma recompensa numérica para o agente. Esta recompensa é uma entrada quantificada que reforça o que o agente acredita ser o correto modo de agir. O ciclo é repetido pelo agente ao tomar uma nova ação (SUTTON; BARTO, 1998).

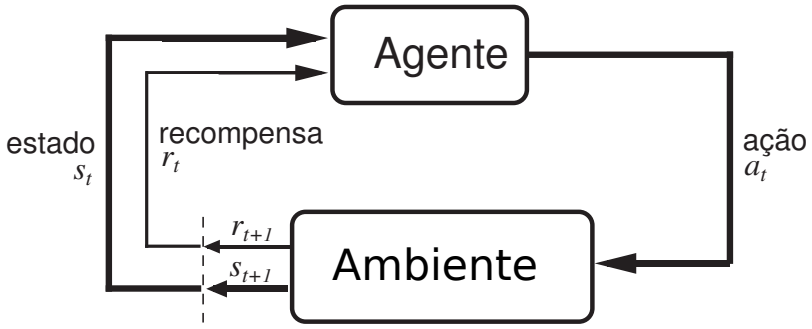


Figura 2.2: Interação agente-ambiente. Adaptado de (SUTTON; BARTO, 1998)

O agente usa a recompensa recebida do ambiente para gerar sua política de ação. A política do agente é um mapeamento de estados para ações numa tentativa de maximizar a recompensa que irá receber no futuro. Uma função valor (em inglês: *value function*) mapeia o estado s e ação a para a recompensa que será recebida ao longo do tempo em que o agente está no estado s, realiza a ação a e continua a seguir a mesma política durante o restante das interações (SUTTON; BARTO, 1998).

As funções valor podem ser inicializadas pessimistamente, otimistamente ou aleatoriamente (SUTTON; BARTO, 1998). Uma inicialização pessimista define todos os pares estado-ação o valor mínimo possível (isto é, se todas as recompensas dadas são maiores ou iguais a 0, o valor inicial 0 é pessimista). Alternativamente, uma inicialização otimista define para todos os pares estado-ação o máximo valor possível. A inicialização pessimista assegura que todas as ações são testadas em todos os estados antes que a convergência para um comportamento fixo ocorra. Enquanto isso, a otimização otimista permite que políticas se tornem favorecidas pelo agente mais rápido, mas não irá descobrir a política ótima até que esta seja encontrada pelo processo de exploração. Já a inicialização aleatória se beneficia dos benefícios de ambos os métodos.

Para maximizar a recompensa recebida pelas interações, o agente deve balancear cuidadosamente a necessidade de explorar com o desejo de tirar proveito do conhecimento adquirido. Especificamente, em cada estado, o agente deve escolher quando vale a pena aproveitar uma ação já conhecida ou explorar novas opções e, potencialmente, descobrir um par estado-ação mais benéfico. Com a exploração o agente deve ocasionalmente esco-

lher ações aleatórias para aprender suas recompensas e descobrir se elas levam a estados com recompensas superiores. Isto impede o agente de agir de forma ótima. Porém, sem exploração, o agente nunca irá ter conhecimento para agir de forma ótima. O equilíbrio entre exploração e aproveitamento é uma chave para o sucesso quando se projeta soluções com RL (SUTTON; BARTO, 1998).

Uma técnica muito utilizada para balancear as ações do agente é agir de forma gulosa na maior parte do tempo e, as vezes, agir de forma aleatória. Desta forma, se garante que uma boa ação não seja perdida ao longo do tempo. Este método de exploração é conhecido como ϵ-guloso (em inglês: *ϵ-greedy*), no qual a opção gulosa é escolhida com uma probabilidade alta $1-\epsilon$, e as ações aleatórias são realizadas com uma pequena probabilidade ϵ.

Embora a escolha de ação proporcionada pelo ϵ-guloso seja efetiva, enquanto explora, esta técnica escolhe um ação com probabilidade igual a todas as demais. Desta forma, o processo de exploração pode selecionar tanto a pior quanto a melhor ação em um determinado momento e, como relatado por (SUTTON; BARTO, 1998), em tarefas onde a pior ação é muito ruim, isto pode levar a resultados insatisfatório. Uma alternativa para este problema é utilizar a abordagem *softmax* ou exploração de Boltzmann (SUTTON, 1990), onde as ações boas têm uma probabilidade exponencialmente alta de serem selecionadas e as ações ruins, baixa. Esta técnica seleciona uma ação a no passo de tempo t com probabilidade:

$$P_a = \frac{e^{\frac{Q_t(a)}{\tau}}}{\sum_{i=1}^{n} e^{\frac{Q_t(i)}{\tau}}} \tag{2.1}$$

onde, e é o expoente do logaritmo natural (aproximadamente 2.718), $Q(a)$ é a função valor da ação a e τ é um parâmetro positivo chamado de temperatura. O parâmetro de temperatura é utilizado para fazer o balanço entre exploração e aproveitamento. Quando $\tau \to 0$ é equivalente ao aproveitamento, já quando $\tau \to \infty$ as ações são puramente aleatórias.

2.2.1 Processo de Decisão de Markov

Grande parte das pesquisas envolvendo aprendizado por reforço monoagente é baseada no processo de decisão de Markov (MDP - *Markov decision process*) (PUTERMAN, 2005). Um MDP é uma tupla $< S, A, T, R >$, onde S representa o conjunto finito de estados do ambiente e A é o conjunto finito de ações do agente. De acordo com a **propriedade de Markov**, a dinâmica do futuro, transições e recompensas são completamente dependentes do estado atual, isto é, uma ação $a \in A$ em um estado $s \in S$ resulta em um estado s', baseado na função de transição $T : S \times A \times S \to [0, 1]$. A probabilidade de transitar para em um estado s' após a realização de uma ação a é denotada como $T(s, a, s')$. A função de recompensa $R : S \to \Re$ retorna a recompensa $R(s, a)$ após o agente tomar uma ação a

a partir de um estado s.

A função de transição T e a função de recompensa R, juntas, são definidas como o modelo que o agente tem do ambiente. A tarefa de aprendizagem em um MDP é encontrar uma política $\pi : S \rightarrow A$ para selecionar ações de maior retorno futuro. A qualidade de uma política é indicada por uma função valor V^{π}. O valor de $V^{\pi}(s)$ específica a quantidade total de recompensa que um agente pode acumular no futuro, iniciando no estado s e então seguindo a política π. Em um MDP com horizonte infinito descontado, a expectativa de recompensa acumulada é denotada por:

$$V^{\pi}(s) = E\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t) \mid s_0 = s\right] \tag{2.2}$$

Um fator de desconto $\gamma \in [0,1)$ é introduzido para assegurar que as recompensas retornadas estejam limitadas em um valor finito. Esta variável determina a relevância da recompensa futura na atualização.

O valor para uma dada política π, expressa pela Eq. 2.2, pode ser iterativamente computada pela equação de Bellman (BELLMAN, 1957). Esta função, tipicamente, é inicializada com valores de função escolhidos arbitrariamente e, a cada iteração, para todo $s \in S$, o valor das funções são atualizados com base na recompensa imediata e na estimativa atual de V_{π}, definida por:

$$V_{t+1}^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} T\left(s, \pi(s), s'\right) V_{t}^{\pi}\left(s'\right) \tag{2.3}$$

O objetivo do MDP é encontrar uma política de estados e ações ótima que maximize a recompensa recebida. A política ótima $\pi^{*}(s)$ é tal que $V^{\pi^{*}}(s) \geq V^{\pi}(s)$ para todo $s \in S$ e todas as políticas π.

2.2.1.1 Processo de Decisão Semi-Markoviano

Assume-se que toda ação em um MDP leva a mesma quantidade de tempo para ser realizada. No entanto, em muitas aplicações práticas esta afirmação não é verdadeira. Por exemplo, considerando um cenário onde um robô, que está aprendendo a se movimentar, possui as seguintes ações: avançar um passo e girar no eixo. Para manter a estabilidade é provável que a ação de girar precise ser mais lenta do que a de avançar um passo.

O processo de decisão semi-Markoviano (SMDP) é uma generalização do MDP que permite abstrair ações que podem levar múltiplos passos de tempo e que podem ter diferentes durações para as outras ações disponíveis ao agente em um mesmo estado. Esta técnica é comumente utilizada em abordagens de aprendizado por reforço hierárquico.

2.2.1.2 Processo de Decisão de Markov Parcialmente Observável

No MDP presume-se que o agente pode observar com precisão o ambiente todo e a qualquer tempo. Quando consideramos aplicações para problemas do mundo real esta afirmação frequentemente não é verdadeira. Por exemplo, em cenários onde robôs recebem informações ruidosas de sensores que podem apenas perceber o ambiente local.

Para modelar este tipo de aplicação deve-se considerar a extensão do MDP para processo de decisão de Markov parcialmente observável (POMDPs) (KAELBLING; LITTMAN; CASSANDRA, 1998). Um POMDP é uma tupla $< S, A, T, R, \Omega, O >$, onde:

- S é o espaço de estados: um conjunto de todos os possíveis estados;
- A é o espaço de ações: um conjunto de todas as possíveis ações;
- T é a função probabilidade de transição $T(s, a, s') = Pr(s'|s, a)$: a probabilidade de que uma ação a em um estado s irá levar ao estado s';
- R é a função de recompensa $R(s, a, s') \in \Re$: a recompensa recebida quando a ação a leva o agente de um estado s para s';
- Ω é o espaço de observação, ou seja, o conjunto de todas as possíveis observações;
- O é a função probabilidade de observação $O(s', a, o) = Pr(o|s', a)$: a probabilidade de receber a observação o quando a ação a gera uma transição para o estado s';

2.2.2 Principais Métodos de RL

A maior parte dos algoritmos de RL para resolver um MDP pode ser agrupada em três categorias gerais: programação dinâmica, aprendizado por diferença temporal e métodos de Monte Carlo. Esta seção, mesmo que brevemente, discute todas estas categorias. Programação dinâmica pode ser utilizada quando as funções de recompensa e transição são conhecidas. Se elas não são conhecidas, mas a propriedade de Markov é consistente para um determinado domínio de problema, o aprendizado por diferença temporal pode ser utilizado neste caso. Se as funções de recompensa e transição são desconhecidas e o domínio do problema não sustenta a propriedade de Markov, métodos de Monte Carlo são mais apropriados.

Para uma leitura mais aprofundada sobre algoritmos de aprendizado por reforço monoagente, recomenda-se a leitura de (SUTTON; BARTO, 1998).

2.2.2.1 Programação Dinâmica

O termo programação dinâmica (PD, em inglês: *dynamic programming*) refere-se à coleção de algoritmos que podem ser utilizados para computar a política ótima de um MDP conhecido e perfeito. Algoritmos clássicos de PD são bastante limitados em RL,

devido à sua suposição de modelo de ambiente perfeito e custo computacional. Porém, ainda assim são importantes teoricamente. De acordo com (SUTTON; BARTO, 1998), os métodos de PD fornecem fundamentos essenciais para compreender os demais métodos de RL existentes.

Um conhecido algoritmo de programação dinâmica é o *policy iteration*. Este algoritmo é inicializado com uma política aleatória e, para cada estado, é verificado se uma melhor ação pode ser realizada. Desta forma, alterando apenas um par estado-ação por vez para um com recompensa maior, o algoritmo pode garantir monotonicamente melhorar a política geral do agente (PUTERMAN, 2005). Outros algoritmos conhecidos de programação dinâmica são: *policy improvement*, *value interation*, *asynchronous dynamic programming* e *generalized policy iteration*. Uma leitura abrangente sobre estes algoritmos pode ser encontrada no Capítulo 4 de (SUTTON; BARTO, 1998).

2.2.2.2 Diferença Temporal

Algoritmos de aprendizado por reforço por diferença temporal (TD, em inglês: *temporal difference*) são métodos iterativos, atualizados *online*, durante as interações do agente com o ambiente. A cada interação com o ambiente o algoritmo reduz gradativamente a diferença entre a expectativa de recompensa e a recompensa recebida, através da atualização das funções valor. Por isso, converge para solução ótima.

O algoritmo Q-learning (WATKINS; DAYAN, 1992) é o mais conhecido exemplo de algoritmo de aprendizado por reforço por TD. Além disso, este algoritmo é livre de modelo. Fato que implica no modelo de mundo do agente, que pode ser incrementado (expandido) de acordo com as interações agente-ambiente. Outra característica importante deste algoritmo é a sua independência de política. Isto permite ao agente atualizar sua função valor, enquanto segue uma política qualquer. De forma mais específica, após qualquer transição de estados, o Q-learning atualiza a função valor através da seguinte regra:

$$Q(s,a) \rightarrow (1-\alpha)\,Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q\left(s', a'\right)\right] \tag{2.4}$$

onde α é a taxa de aprendizagem, e γ o fator de desconto, s é o estado inicial, a é a ação tomada e s' é o estado resultante.

Tanto a taxa de aprendizagem quanto o fator de desconto são parâmetros individuais de cada problema. O fator de aprendizagem implica nos efeitos de quão grande será a atualização de um valor Q. Já o fator de desconto determina a importância das recompensas futuras.

Segundo (SUTTON; BARTO, 1998), o Q-learning tem provado convergir para política ótima em um MDP, se as seguintes restrições forem satisfeitas:

- Todos os pares estado-ação são experienciados por um infinito número de vezes;

- A exploração seja reduzida a zero;
- A taxa de aprendizagem (α) seja reduzida a zero;
- A propriedade de Markov seja satisfeita.

2.2.2.3 Métodos de Monte Carlo

Métodos de aprendizado por reforço puramente baseados em TD, atualizam apenas o valor do último par estado-ação. No entanto, se a propriedade de Markov não é satisfeita, o histórico de estados e ações pode ser responsável pela recompensa recebida. Os métodos de Monte Carlo consideram todos os pares estado-ação durante a interação com o ambiente e atualiza-os apenas uma vez, ao fim da interação. Além disso, para problemas onde a propriedade de Markov não é satisfeita, métodos de Monte Carlo são mais adequados para a tarefa de aprendizado (SUTTON; BARTO, 1998).

Os métodos de Monte Carlo aprendem funções valor e políticas ótimas a partir de experiências em exemplos - na forma de episódios. Segundo (SUTTON; BARTO, 1998), em relação aos métodos de PD, este aprendizado episódico tem pelo menos três vantagens. Primeira, eles podem ser utilizados para aprender um comportamento ótimo diretamente da interação com o ambiente, ou seja, sem um modelo da dinâmica do ambiente. Segundo, eles podem ser utilizados com a simulação ou modelos de exemplo. Terceiro, sua aplicação em um pequeno subconjunto de estados é simples e eficiente.

O Capítulo 5 de (SUTTON; BARTO, 1998) apresenta uma leitura abrangente sobre esta categoria de algoritmos de RL.

2.3 Aprendizado por Reforço Multiagente

No problema de aprendizagem por reforço multiagente, um agente precisa aprender seu comportamento ótimo na presença de outros agentes que também são capazes de aprender. Neste problema, todos os agentes estão situados em um ambiente em comum, o qual, em função da adição de mais agentes, tem um aumento significativo na quantidade de variáveis observáveis (BUŞONIU; BABUSKA; DE SCHUTTER, 2008).

Existem muitas técnicas propostas para realizar a tarefa de aprendizagem por reforço neste problema. Entretanto, como será apresentado na Seção 2.3.4, estas abordagens podem aumentar significativamente a complexidade do problema, em função do número de agentes aprendendo e da quantidade de estados e ações disponíveis. Embora apresente problemas como este, os benefícios proporcionados pelo MARL são imensos, dos quais (BUŞONIU; BABUSKA; DE SCHUTTER, 2008) destacam:

- Aceleração do processo de aprendizagem: possível graças à computação paralela proporcionada pela estrutura descentralizada das tarefas. (GUESTRIN; LAGOU-

DAKIS; PARR, 2002; BUŞONIU; BABUSKA; DE SCHUTTER, 2008) apresentam um estudo detalhado nesta direção;

- Compartilhamento de experiência: permite que agentes com tarefas similares aprendam melhor e mais rapidamente. Assim, através da comunicação, agentes mais habilidosos podem servir como professores para o aprendiz, ou o aprendiz pode assistir e imitar agentes habilidosos;
- Robustez: quando um ou mais agentes falham em um sistema multiagente, os agentes restantes podem assumir algumas de suas tarefas.

As seções subsequentes apresentam a extensão do MDP monoagente para o caso de múltiplos agentes (Seção 2.3.1), suas implicações na qualidade da solução encontrada (Seções 2.3.2 e 2.3.3) e principais categorias de algoritmos de MARL, como: múltiplos aprendizes independentes (Seção 2.3.4.1), *joint action learner* (Seção 2.3.4.2), jogos competitivos (Seção 2.3.4.3), jogos cooperativos (Seção 2.3.4.4) e jogos de soma geral (Seção 2.3.4.5). Uma maior discussão sobre o assunto pode ser encontrada em (BUŞONIU; BABUSKA; DE SCHUTTER, 2008). Neste trabalho são apresentadas as principais técnicas de MARL, bem como os principais desafios e aplicações da área. Para um panorama geral e atualizado sobre MARL recomenda-se o trabalho de (TUYLS; WEISS, 2012).

2.3.1 Processo de Decisão de Markov Multiagente

Quando muitos agentes interagem através de seus processos de aprendizagem, o modelo básico do MDP não é robusto o bastante para representar a tarefa de aprendizado. Jogos estocásticos, também conhecidos como processo de decisão de Markov multiagente (MMDP, em inglês: *multiagent Markov decision process*) generalizam jogos repetidos (sem estado) e MDP (monoagente) para um caso geral de múltiplos estados e múltiplos agentes. Nesta abordagem, a cada etapa, o jogo está em um estado específico, com uma determinada função de recompensa e um conjunto admissível de ações para cada jogador. Os jogadores, por sua vez, tomam ações simultaneamente e em seguida recebem uma recompensa imediata em função de suas ações conjuntas. Uma função de transição mapeia o espaço de ações conjuntas (JA, em inglês: *joint actions*) para uma distribuição de probabilidades sobre todos os estados que, por sua vez, determina a probabilidade de mudança de estado. Portanto, assim como no MDP em MMDP as ações influenciam diretamente nas transições de estado.

Um MMDP para n agentes é formalmente definido por uma tupla $2n+2\,\langle S, A_1 \ldots, A_n, T, R_1, \ldots R_n\rangle$ onde:

- S é o espaço de estados, um conjunto de todos os estados possíveis;
- A_i é o espaço de ações do agente i, um conjunto de todas as ações possíveis do agente i;

- T é a função probabilidade de transição: $T(s, a, s') = Pr(s' \mid s, a)$, a probabilidade de que a ação conjunta a, no estado s, levará ao estado s';
- R_i é a função de recompensa do agente i: $R_i(s, a, s') \in \Re$, a recompensa recebida quando a ação conjunta a transita um agente do estado s para s'.

O primeiro trabalho a utilizar MMDP como estrutura para MARL foi (LITTMAN, 1994). No entanto, sua abordagem apresentou problemas de escalabilidade, já que o espaço de estado-ações cresce rapidamente em função do número de agentes e ações a eles disponíveis. Para resolver este problema, trabalhos subsequentes surgem com formas mais compactas (KOK; VLASSIS, 2004, 2006; OLIVEIRA; BAZZAN, 2009) e descentralizadas (BERNSTEIN et al., 2002) de implementação de MARL.

Existem três formas de MMDPs: completamente cooperativos (em inglês: *fully cooperative*), completamente competitivos (em inglês: *fully competitive*) ou uma mistura de ambos. Em MMDPs completamente cooperativos a função de recompensa de todos os agentes é a mesma. Ao contrário disso, em jogos de dois jogadores, onde a soma das recompensas recebidas para cada par de estados e ações conjuntas é zero, denomina-se MMDP completamente competitivo. Já os jogos com elementos competitivos e cooperativos são conhecidos como jogos de soma geral (em inglês: *general-sum games*).

Ao contrário do que acontece em um MDP, em MMDPs não há um conceito claro de política ótima. No entanto, a teoria dos jogos apresenta muitos conceitos alternativos que podem ser utilizados para solucionar este problema. Teoria dos jogos e MARL estão muito ligadas, já que os algoritmos de MARL combinam aspectos de programação dinâmica e/ou aprendizado por diferença temporal com teorias do campo (BUŞONIU; BABUSKA; DE SCHUTTER, 2008).

Tipicamente, no MARL os agentes aprendem uma política de ação conjunta a qual representa um equilíbrio de Nash. Porém, em alguns casos pode ser preferível aos agentes aprender uma política conjunta Pareto ótima. Tanto o equilíbrio de Nash quanto Pareto ótima são discutidos nas próximas seções.

2.3.2 Equilíbrio de Nash

O equilíbrio de Nash (*Nash Equilibrium*) é uma política conjunta onde nenhum agente se beneficia ao mudar sua própria política, assumindo que todos os outros agentes irão manter sua política inalterada. Existem dois tipos de políticas onde o equilíbrio de Nash está presente: pura e mista. Na política pura cada agente sempre seleciona a mesma ação; já na mista, para cada ação atribui-se uma probabilidade de escolha.

Este conceito foi proposto por John Nash (1951), para jogos não cooperativos e é largamente utilizado em MARL. O autor afirma que em jogos com finito número de agentes e com finito número de ações existe pelo menos um equilíbrio de Nash presente.

A definição formal de uma política conjunta π^{NE} é dada por:

$$\forall_{i \in 1...n}, \pi_i \in \prod \mid R_i\left(\pi_i^{NE} \bigcup \pi_{-i}^{NE}\right) \geqslant R_i\left(\pi_i \bigcup \pi_{-i}^{NE}\right) \tag{2.5}$$

onde n é o número de agentes, $\prod_i$ é o conjunto de todas as possíveis políticas para o agente i, R_i é a função de recompensa para o agente i, $\pi_i^N E$ é uma política específica para o agente i e π_{-i}^{NE} é uma política conjunta de todos os agentes menos o agente i, seguindo suas próprias políticas específicas e fixas.

2.3.3 Ótimo de Pareto

Uma política conjunta π^a é dita Pareto dominante de uma outra política conjunta π^b se, ao usar π^a, um ou mais agentes recebem uma recompensa maior e todos os demais agentes recebem a mesma recompensa. Qualquer outra política conjunta que não é Pareto dominada por outra é Pareto ótima. Assim, se uma política conjunta é Pareto ótima, não há política conjunta que possa melhorar a recompensa de algum agente sem que a recompensa de outro agente seja reduzida (FUDENBERG; TIROLE, 1991).

2.3.4 Principais Métodos de MARL

2.3.4.1 Múltiplos Aprendizes Independentes

Uma forma muito utilizada para contornar o problema de escalabilidade em MMDPs é dada através da modelagem individual do processo decisório dos agentes. Desta forma, cada agente possui seu MDP descentralizado e aprende sem considerar as ações dos demais agentes. Neste caso, um agente entende o aprendizado e mudança de comportamento dos demais agentes como uma mudança na dinâmica do ambiente. Nesta abordagem, os agentes são chamados aprendizes independentes (em inglês: *multiple independent learners*). A grande desvantagem desta técnica é que algoritmos como Q-learning não são tão robustos quanto em cenários monoagente (CLAUS; BOUTILIER, 1998). Além disso, o aprendizado de agentes sem considerar a adaptação dos outros agentes no processo de decisão, pode convergir para mínimos locais, onde os agentes param de aprender e, portanto, não é matematicamente justificável (LITTMAN; DEAN; KAELBLING, 1995). Apesar disso, trabalhos como (LITTMAN; DEAN; KAELBLING, 1995) atingem resultados satisfatórios com essa abordagem.

2.3.4.2 Joint Action Learner

Joint action learner (JAL) é uma extensão do algoritmo Q-learning para um único agente. A diferença entre os métodos está na tabela Q, onde o JAL aprende os valores Q para ações conjuntas em vez de ações individuais. As experiências dos agentes são obtidas através dos resultados destas ações conjuntas. Isto implica que cada agente pode observar as ações de outros agentes. Para determinar os valores das ações conjuntas em relação de suas ações individuais, cada agente mantém crenças sobre as estratégias de outros agentes.

Embora este algoritmo se beneficie da completa observação do ambiente e de sua dinâmica, o espaço necessário para armazenar os valores Q e o número de experiências necessárias para aprender crescem exponencialmente com cada agente adicionado (CLAUS; BOUTILIER, 1998). Desta forma, em problemas como roteamento de veículos em redes viárias onde o número de agentes aprendendo pode chegar a milhares ou milhões, o uso de JAL se torna inviável.

2.3.4.3 Jogos Competitivos

O algoritmo *mini-max* Q-learning, introduzido por (LITTMAN, 1994), foi o primeiro trabalho a combinar aprendizado por diferença temporal com a teoria dos jogos, para resolver um subconjunto de problemas de MARL. O *mini-max* Q-learning é uma modificação do JAL para jogos completamente competitivos (isto é: jogos de dois jogadores, soma-zero). A cada seleção de ação, um agente mini-max Q-learning irá escolher a ação com maior valor assumindo que o agente oposto irá buscar minimizar a recompensa que o agente aprendendo pode receber. Este algoritmo possui garantias de convergência para uma política fixa que recebe a maior recompensa possível contra o pior oponente possível (i.e. o agente oponente que tenta minimizar a recompensa do agente). Caso o agente oponente não seja o pior possível, o agente aprendendo com *mini-max* Q-learning irá receber a maior recompensa possível.

2.3.4.4 Jogos Cooperativos

(LAUER; RIEDMILLER, 2000) propõem uma extensão de múltiplos aprendizes independentes para jogos completamente cooperativos, chamada Q-learning distribuído (*distributed Q-learning*). Em ambientes determinísticos completamente cooperativos, se uma recompensa recebida por um agente, em um determinado par estado-ação, for menor do que a recompensa recebida pelo mesmo agente, mas em uma experiência anterior neste par, pode-se assumir que outro agente é culpado pela queda na recompensa. Por essa razão, no Q-learning distribuído os agentes nunca reduzem o valor de suas tabelas Q. Além

disso, se todos os agentes em um ambiente determinístico completamente cooperativo utilizarem o Q-learning distribuído, eles irão convergir para uma política ótima.

Uma alternativa para ambientes completamente cooperativos, porém estocásticos é proposta por (WANG; SANDHOLM, 2002). Nesta abordagem os autores apresentam uma modificação do JAL, chamada aprendizado adaptativo ótimo (em inglês: *optimal adaptative learning*), o qual possui garantias de convergência para um equilíbrio de Nash.

2.3.4.5 Jogos de Soma Geral

Em jogos de soma geral (em inglês: *general-sum games*) o algoritmo mais clássico é o Nash Q-learning (HU; WELLMAN, 2003), o qual tem por objetivo encontrar a melhor política para um agente em relação a outros agentes. Para fazer isso, os agentes precisam aprender as políticas dos demais agentes e construir a melhor resposta frente a isso. O processo de aprendizado de políticas dos demais agentes envolve a formação de suposições sobre o comportamento deles. Pode-se adotar uma abordagem puramente comportamental, inferindo políticas do agente diretamente com base em padrões observados ou pode-se considerar que os outros agentes são racionais e fazem suposições sobre as ações que os demais agentes tomarão.

(HU; WELLMAN, 2003) provam que o Nash *Q-learning* converge para um equilíbrio de Nash sob condições estritas. Na prática, (BUŞONIU; BABUSKA; DE SCHUTTER, 2008) mostram que às vezes o algoritmo converge sem as condições necessárias e que é mais provável que convirja quando comparado ao uso de múltiplos aprendizes independentes, em jogos de soma geral.

2.3.5 Funções de Recompensa

Embora esta seção esteja situada dentro de MARL, os conceitos aqui apresentados são básicos e compreendem tanto MARL quanto ML.

(SUTTON; BARTO, 1998) afirmam que além do(s) agente(s) e ambiente, existem outros subelementos importantes em aprendizado por reforço, como: política, função de recompensa e função valor.

A política é quem define o modo em que um agente aprendendo por reforço deve agir ao longo do tempo. A grosso modo, uma política é o mapeamento dos estados percebidos do ambiente para ações a serem tomadas quando nestes estados. Em muitos casos a política pode ser uma simples função ou tabela de consulta, mas em outros pode envolver extensa computação como processos de busca.

A função de recompensa é responsável por definir o objetivo da tarefa de aprendizado em sistemas multiagente. A grosso modo, a função de recompensa mapeia cada estado percebido (ou par estado-ação) do ambiente para um único número, a recompensa, ex-

pressando o quão desejado um estado é. O objetivo único de um agente que aprende por reforço é maximizar a recompensa total recebida ao longo da execução. Desta maneira, a função de recompensa define o que são ações boas ou ruins para o agente. As funções de recompensa são imediatas e definem as características avaliadas pelos agentes. (SUTTON; BARTO, 1998) ressaltam que estas funções devem ser necessariamente inalteráveis pelos agentes, já que podem servir como base para alterar uma política. Por exemplo, se uma ação selecionada por um agente é seguida por uma recompensa baixa, então esta política pode ser alterada para selecionar outras ações situações. Apesar disso, estas funções podem ser estocásticas.

Enquanto que as funções de recompensa indicam a qualidade dos estados através de uma recompensa imediata, uma função valor especifica o que é bom ao longo da execução. Em outras palavras, a função valor de um estado é o total de recompensa que um agente pode esperar acumular no futuro, a partir daquele estado. A medida que a função de recompensa determina a recompensa imediata de um estado do ambiente, valores de função indicam a recompensa a ser recebida ao longo do tempo, após levar em consideração os estados que são desejáveis seguir e as recompensas disponíveis naqueles estados. Por exemplo, um estado pode sempre levar a uma baixa recompensa imediata, mas o agente ainda pode receber uma recompensa alta, porque este estado pode levar a outros estados com recompensas altas. Entretanto, o caso contrário também pode ocorrer.

Como se pode perceber, tanto a função valor quanto a política são diretamente influenciadas pela função de recompensa. Desta forma, uma função de recompensa que expresse a tarefa a ser aprendida de forma inadequada pode levar a funções valor inadequadas e, portanto, a políticas subótimas. Além disso, da forma com que as funções de recompensa usualmente são implementadas, elas buscam maximizar a utilidade individual de um agente. Todavia, em MARL, mesmo que não haja uma coordenação explícita entre os agentes, pode ser mais benéfico para um coletivo de agentes que a função de recompensa esteja alinhada à utilidade global do sistema. Neste sentido, a próxima seção apresenta uma abordagem desenvolvida para estimular a emergência de tal comportamento em SMA.

2.3.6 Difference Rewards

Difference rewards são formas específicas de modelagem de recompensa, resultantes de um trabalho de Tumer e Wolpert (2004) sobre "inteligência coletiva". Muitos pesquisadores, enquanto implementam soluções MARL, desenvolvem funções de recompensa privadas para cada agente e procuram observar o comportamento emergente do SMA. Pesquisas em inteligência coletiva exploram como inverter este processo. Assim, focando-se na utilidade global (ou utilidade do sistema), tais pesquisas investigam como projetar funções de recompensa individuais que combinadas melhoram o desempenho do sistema.

Uma grande preocupação das pesquisas nesta área é assegurar que os agentes não atuarão no SMA contra o desempenho global. Aplicações como (TUMER; WOLPERT, 2000; TUMER; KHANI, 2009; AGOGINO; TUMER, 2012) ilustram como as abordagens baseadas em *difference rewards* podem contornar típicos problemas de agentes se comportando de forma gulosa, para maximizar sua recompensa individual.

Um típico exemplo deste problema é conhecido como paradoxo de Braess. (TUMER; WOLPERT, 2000) apresentam o problema em um cenário de escolha de rotas em rede de tráfego. Como ilustrado pela Fig. 2.3, ambas as redes apresentam rotas entre a origem O e o destino D. Para n agentes viajando de O para D, passar pelas cidades c_1 custa $10n$, c_2 custa $50+10n$ e c_3 custa $10+n$ (TUMER; WOLPERT, 2000). A única diferença entre as duas redes é a adição de uma rota significativamente menos custosa. Intuitivamente, o custo médio de todos os agentes cruzando a *Rede B* deve ser menor do que o da *Rede A*. Contudo, para estes problemas, um caminho mais curto (ou menos custoso) não considera a utilidade global e pode, potencialmente, aumentar o custo de todos os agentes. Em função disto, eleva-se o custo de todos os agentes - e consequentemente o custo global - o que leva ao surgimento do paradoxo de Braess.

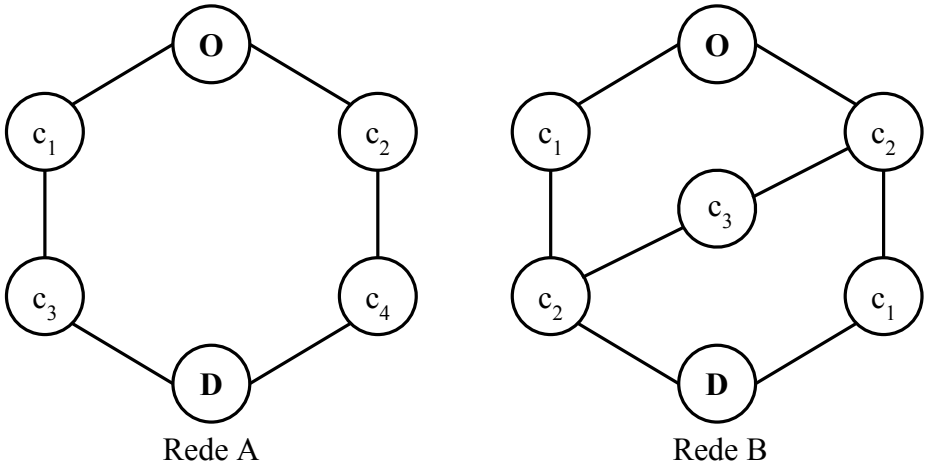


Figura 2.3: Um exemplo de paradoxo de Braess. Adaptado de (TUMER; WOLPERT, 2000).

Para solucionar este problema, os autores alinharam a função de recompensa à função de utilidade do sistema. Desta forma, ao invés de receber a recompensa global, cada agente foi recompensado com a diferença entre a função de utilidade do sistema e o que essa função seria caso o agente não estivesse atuando no sistema. Esta função de recompensa resultou em uma recompensa maior em comparação ao agente que age de forma individual se, e somente se, um aumento correspondente ocorre na função de utilidade global. Esta técnica de modelagem de função de recompensa em MARL é conhecida como *difference rewards* (AGOGINO; TUMER, 2012), e sua definição formal é dada por:

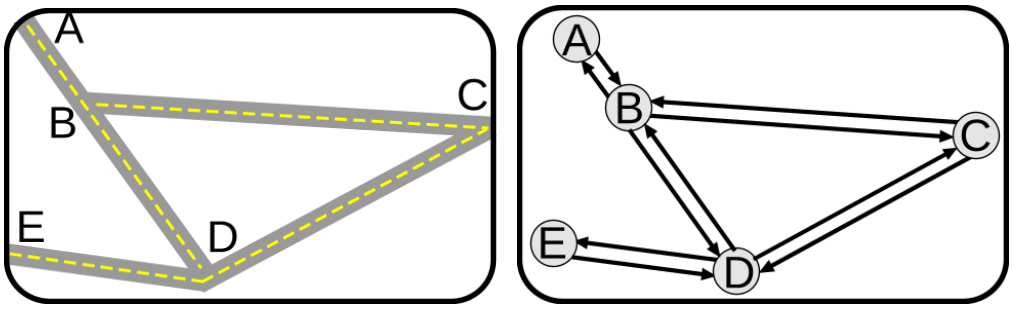
$$D_d \equiv G(z) - G(z_{-d}) \tag{2.6}$$

onde D_d é a recompensa recebida pelo agente d, z representa os pares estado-ação de todos os agentes, $G(z)$ a recompensa que todos os agentes deveriam receber do ambiente e $G(z_{-d})$ é a recompensa que todos os agentes deveriam receber do ambiente se o agente d não estivesse na simulação. Utiliza-se o termo z_{-d} para representar os pares estado-ação dos agentes caso d não esteja presente no sistema.

2.4 Oferta e Demanda em Sistemas de Transporte

Como relatado em (BAZZAN; KLÜGL, 2013), não existe uma definição simples do que constitui um sistema de transporte. Em função do contexto, perspectiva ou propósito da literatura técnica específica, diversas definições podem ser encontradas. Por exemplo, quando estão se referindo aos meios de transporte, cada meio pode ser referido a um sistema (por exemplo, o sistema de trânsito). Desta forma, um sistema de transporte pode ser visto como um composto de uma parte de oferta (estrutura física) e outra parte chamada demanda (viagens)(BAZZAN; KLÜGL, 2013). Além disso, a escala do sistema (i.e. urbano, interurbano, internacional e etc.) também pode ser utilizada para caracterizar um sistema de transporte.

A representação da oferta, em termos de infraestrutura de transporte, pode ser feita através dos conceitos de teoria dos grafos. Isto posto, a rede pode ser representada por um grafo $G(V, E)$, onde conjunto de vértices V representa os cruzamentos ou intersecções e o conjunto de arestas (ou links) E representa as vias que ligam as intersecções. Cada link $l_k \in L$ possui um custo (ou peso) c_k, o qual é representado por alguma forma de custo associada à travessia do link, como: seu comprimento, tempo de viagem para trafegá-lo, quantidade de combustível consumido, entre outros. A Fig. 2.4 apresenta, para uma rede viária hipotética e a sua respectiva representação em forma de grafo. Vale ressaltar que a representação do duplo sentido é representada por dois links, um para cada sentido.



(a) (b)

Figura 2.4: Um exemplo de rede viária (a) e sua correspondente representação em forma de grafo (b) (TAVARES, 2013).

A demanda, por conseguinte, representa o desejo de determinadas entidades em se locomoverem de um lugar para outro. Dentre as formas de modelagem de demanda

existentes, (ARFAOUI, 1999) afirma que a rede pode ser divida em zonas, distritos ou centroides (a partir de dados socioeconômicos, densidade populacional, entre outros fatores). A partir destes dados, uma matriz T chamada de matriz origem-destino (matriz OD) é construída. Esta matriz contém I linhas (zonas de origem) e J colunas (zonas de destino). Cada elemento T_{ij} representa o número de viagens da zona i para a zona j, em um dado intervalo de tempo. Diz-se que $i \in I$ e $j \in J$ são um par origem-destino (ou par OD). Para encontrar a proporção de viagens de um determinado par OD, em relação ao total de viagens, basta dividir os elementos de T_{ij} pelo total de viagens de T.

2.4.1 Alocação de Tráfego

Para que seja possível predizer a carga ou fluxo de uma rede de tráfego, deve-se determinar quais partes da infraestrutura as viagens estão dispostas a alocar. Isto significa que, de alguma forma, a demanda precisa ser alocada à oferta disponível. A alocação de tráfego (TA, em inglês: *traffic assignment*) também conhecida como alocação de rotas ou alocação de viagens modela esta interação. A partir da saída desta técnica é possível descrever o estado do sistema de transporte o qual é muito relevante para, por exemplo, avaliar as consequências de mudanças na infraestrutura.

De modo geral, TA aloca rotas que conectam as origens das viagens com seus respectivos destinos presentes na matriz OD. A conexão entre origem i e destino j geralmente não consiste de apenas um único link, mas uma sequência de nodos conectados $(n_0, n_1, n_2, \ldots)$ que juntos formam uma rota r_p. O comprimento de cada r_p é a soma dos comprimentos de todos os links l_k que conectam seus nodos. Os mais conhecidos algoritmos para determinar um caminho relevante são variantes dos algoritmos de busca de menor caminho, entre os quais se destacam o Dijkstra (DIJKSTRA, 1959) e as variantes do A^*. Ambos os algoritmos baseiam-se nos custos atribuídos aos links para determinar as rotas.

Dentre as diversas formas de avaliação do custo por trafegar em um link, pode se destacar o tempo de viagem para percorrê-lo. Este tempo de viagem pode ser representado por uma função como a VDF (em inglês: *volume-delay function*). Esta função expressa o tempo médio de viagem no link, de acordo com o seu volume de tráfego. Sua representação formal é dada por:

$$t_k = t_{k0}\left[1 + \mathcal{A}_k\left(\frac{V_k}{C_k}\right)^{\mathcal{B}_k}\right] \tag{2.7}$$

onde t_{k_0} é o tempo de viagem no link k sem fluxo, V_k é o volume (em veículos por unidade de tempo) e C_k é a capacidade do link. $\mathcal{A}_k$ e $\mathcal{B}_k$ são fatores que devem ser empiricamente definidos em cada link, para determinar como o tempo de viagem deve reagir ao aumento do volume.

Para demonstrar como se comporta o tempo de viagem em função do volume no link,

a Fig. 2.5, ilustra o crescimento do tempo de viagem no link que conecta os nodos 22 e 23 (neste sentido), escolhido arbitrariamente da rede Sioux Falls (apresentada na Seção 5.3). Os parâmetros da VDF deste link são: $C_k = 5000$, $\mathcal{A}_k = 0.15$ e $\mathcal{B}_k = 4$. Como pode ser observado, enquanto o volume de viagens é inferior à capacidade da rede, não há um aumento muito significativo no tempo de viagem. Porém, quando o volume excede a capacidade, o tempo de viagem aumenta exponencialmente.

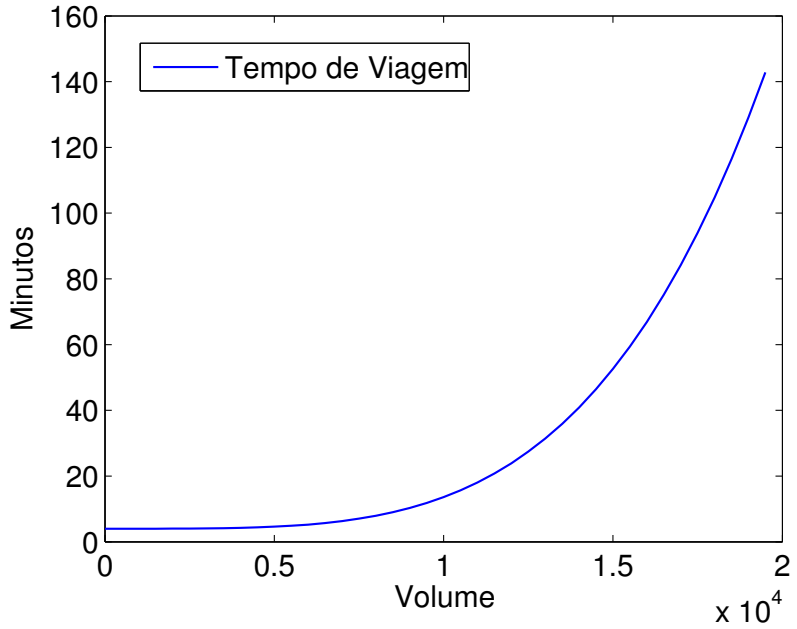


Figura 2.5: Comportamento da VDF para o link 22-23 da rede Sioux Falls.

Devido a este crescimento exponencial do tempo de viagem nos links, é fundamental que rotas sejam alocadas às viagens de maneira eficiente. A sobrecarga de links contribui consideravelmente para o aumento do tempo de viagem na rede, além de subutilizar a infraestrutura existente. A Seção 2.4.2 apresenta alguns dos principais métodos de TA.

2.4.2 Métodos de Alocação de Tráfego

Existem duas principais categorias de métodos para realizar TA: métodos para redes congestionadas e não congestionadas. Esta seção, apresenta um método para redes não congestionadas (Seção 2.4.2.1) e dois métodos direcionados para redes congestionadas (Seção 2.4.2.2 e Seção 2.4.2.3). Para uma leitura mais abrangente sobre estes métodos recomenda-se a leitura do capítulo 10 de (ORTÚZAR; WILLUMSEN, 2001).

2.4.2.1 All-Or-Nothing Assignment

All-or-nothing assignment (em português: alocação tudo ou nada) é o método mais simples de alocação de rotas existente. Ele assume que não existem efeitos de conges-

tionamentos na rede, tal que todas as viagens consideram os mesmos atributos durante o cálculo da rota. A ausência dos efeitos dos congestionamentos implica diretamente na escolha de rota.

A alocação de tráfego deste método é dada pela Eq. 2.8. Como pode ser observado, todas as viagens com origem em i e destino em j utilizarão os mesmos links. Para (ORTÚZAR; WILLUMSEN, 2001), esta suposição pode ser razoável em redes esparsas ou onde há poucas rotas alternativas e muita diferença de custo entre elas.

$$\left.\begin{array}{lll} T_{ijr^*} & = T_{ij} & \text{para a rota de menor custo } r^* \\ T_{ijr} & = 0 & \text{para todas as demais rotas} \end{array}\right\} \forall_{i,j} \tag{2.8}$$

O uso deste método geralmente está limitado aos interesses do planejador de tráfego e pode ser utilizado como uma estratégia para representar o comportamento desejável de motoristas, na ausência de congestionamentos na rede. Apesar de ser um método muito simples, tanto o *successive averages assignment* quanto o *incremental assignment*, apresentados a seguir, utilizam-no como estrutura básica.

2.4.2.2 Incremental Assignment

Esta é uma abordagem um pouco mais interessante e realista quando comparada ao *all-or-nothing assingment*. Neste método, divide-se o total de viagens da matriz T (para cada par OD) em n frações de matriz, de acordo com um fator de proporção p_n, tal que $\sum_n p_n = 1$. As matrizes fracionadas são carregadas na rede de forma incremental (por exemplo, 10%, 20%, etc.). Após cada fração de matriz ser carregada, os pesos dos links são novamente calculados. Segundo (ORTÚZAR; WILLUMSEN, 2001), são valores típicos de p_n : 0.4, 0.3, 0.2 e 0.1. Um pseudocódigo para este algoritmo é apresentado a

seguir:

Algoritmo 1: Pseudocódigo do algoritmo Incremental Assignment

Entrada: uma matriz T; um vetor fator de proporção p_n; um grafo $G(V, E)$;
Saída: custos dos links;

```
1  início
2      inicialize o custo c_k de todos os links k (geralmente com t_{k_0});
3      inicialize os fluxos de todos os links k: V_k = 0;
4      selecione uma fração p_n da matriz de viagens T tal que ∑_n p_n = 1;
5      n = 0;
6      repita
7          construa o conjunto de rotas de menor custo (uma para cada par OD)
           utilizando os custos atuais;
8          n ← n + 1;
9          carregue T_n = p_n T através do all-or-nothing e das rotas geradas;
10         obtenha o conjunto de fluxos auxiliares F_k;
11         acumule os fluxos em cada link V_k^n = V_k^{n-1} + F_k;
12         atualize o custo dos links com base em V_k^n;
13     até que todas as viagens de T sejam carregadas;
14 fim
```

Uma grande desvantagem deste método é que após um fluxo ser atribuído a um link, este fluxo não será mais removido e alocado para outro link. Logo, se as iterações iniciais alocarem muito tráfego para um determinado link, provavelmente o método não convergirá para uma boa solução. Apesar disso, (ORTÚZAR; WILLUMSEN, 2001) ressaltam que este método proporciona duas principais vantagens: facilidade de programação e que seu resultado pode ser interpretado como o congestionamento formado em períodos de pico.

2.4.2.3 Successive Averages

O método *successive averages* também é um método iterativo de TA, porém um pouco mais eficiente quando comparado com ao *incremental* e *all-or-nothing assignment*. Este método aloca rotas as viagens a partir de repetidas execuções do *all-or-nothing*. A cada repetição, custo e volume dos links são atualizados, por meio da distribuição das viagens nas iterações anteriores. Este processo é repetido até que não haja uma significativa mudança no custo dos links ou volumes. Por exemplo, no método *successive averages*, o fluxo na iteração n é calculado como uma combinação linear do fluxo na iteração anterior e de um fluxo auxiliar resultante da execução do *all-or-nothing* na iteração n. Este

procedimento pode ser representado por:

Algoritmo 2: Pseudocódigo do algoritmo Successive Averages

Entrada: uma matriz T; uma taxa de atualização do fluxo ϕ; um grafo $G(V, E)$;
Saída: custos dos links;

1 **início**
2 inicialize o custo c_k de todos os links k (geralmente com t_{k_0});
3 inicialize os fluxos de todos os links k: $V_k = 0$;
4 torne $n = 0$;
5 **repita**
6 construa o conjunto de rotas de menor custo (uma para cada par OD) utilizando os custos atuais;
7 $n \leftarrow n + 1$;
8 carregue toda a matriz T com o *all-or-nothing* para as rotas geradas;
9 obtenha o conjunto de fluxos auxiliares F_k;
10 calcule o fluxo atual por: $V_k^n \leftarrow (1 - \phi) V_k^{n-1} + \phi F_k$, com $0 \leq \phi \leq 1$;
11 atualize o custo dos links com base em V_k^n;
12 **até** *enquanto houver mudança significativa em* V_k^n *ou atingir um número máximo de iterações*;
13 **fim**

Como pode ser observado na linha 10, o parâmetro ϕ representa a taxa de atualização dos fluxos. Existem várias formas de modelar a atualização deste parâmetro, entre as quais a forma mais comum, segundo (ORTÚZAR; WILLUMSEN, 2001), é mantê-lo fixo em $\phi = 0.5$. No entanto, uma estratégia mais eficiente, muito utilizada na literatura, é atualizar-lo por $\phi = \frac{1}{n}$, onde n é o contador da iteração.

2.5 Simulação de Tráfego

Modelagem e simulação são umas das maiores áreas de aplicação para a ciência da computação, no domínio de transporte. Neste contexto, uma variedade de simuladores é apresentada como soluções para o gerenciamento e simulação de tráfego. Conforme afirmam (BAZZAN; KLÜGL, 2013), reproduzir sistemas de tráfego em um modelo é um meio estabelecido de predizer os efeitos de uma política em particular ou medida de infraestrutura. Já as simulações não são apenas utilizadas para testar o impacto de uma nova infraestrutura, elas também podem ser utilizadas para avaliar a viabilidade de técnicas inovadoras como comunicação ou sistemas de localização.

A inteligência artificial há muito tempo tem sido aplicada em simulações de tráfego. (BAZZAN; KLÜGL, 2013) afirmam que a partir da metade dos anos 90, simulação baseada em agentes tem se tornado um guarda-chuva muito relevante para diferentes abordagens em simulação de tráfego. Modelos baseados em agente utilizam o conceito de sistemas

multiagente como metáfora básica para a modelagem. Para a simulação de tráfego, isso implica que as entidades tomadoras de decisão (motoristas neste caso) de forma ativa decidem por si suas próprias atividades ou interações, baseadas nas suas informações locais. Ao contrário do que ocorre, por exemplo, nos métodos de TA mencionados anteriormente, em simulação baseada em agentes é possível simular cada decisão individual sobre tempo de partida, meio de transporte, rota e etc.

A literatura apresenta diferentes abordagens de simulação que diferem no nível de detalhe da representação do sistema. Dentre as existentes, destacam-se os seguintes níveis de detalhes:

- Macroscópico - Neste nível de detalhe, há uma abstração das ações dos indivíduos e suas interações. O sistema é representado de maneira agregada a partir de histogramas e dados de volumes, densidades e velocidades. Devido a essa abstração e agregação dos dados, estes modelos são computacionalmente mais fáceis de calcular e executar. Além disso, a tarefa de calibração está restrita a poucos parâmetros.
- Mesoscópico - Este é um nível de detalhe intermediário entre macroscópico e microscópico, combinando as vantagens de ambos, tais como cálculo eficiente (macro) e riqueza de detalhes e expressividade (micro). As entidades individuais são representadas com um nível de detalhe razoável, mas as interações entre elas são abstraídas. Por exemplo, uma manobra de mudança de pista pode ser representada como um evento instantâneo (para todos os veículos) e ser baseada na densidade da pista e não na interação entre os veículos propriamente ditos.
- Microscópico - Representa de maneira detalhada e individual as interações das entidades. Por exemplo, em termos de movimento, é possível modelar a mudança de pista e comportamento do tipo *car-following*[1]. Neste nível, o comportamento global do sistema emerge a partir das interações dos indivíduos. Como principal desvantagem, tais modelos são geralmente complexos, de desenvolvimento custoso e exigem mais parâmetros de configuração e calibração.
- Submicroscópico - Neste nível de detalhe, são simulados fatores como o atual funcionamento do veículo, o qual é modelado por leis da física. Os efeitos das ações do usuário do veículo, como: frear, acelerar, girar o volante e suas consequentes reações são possíveis de ser modeladas em simulações submicroscópicas.

Como a abordagem proposta nesta dissertação não exige da simulação detalhes como posição física dos veículos ou simular comportamentos como troca de pista e movimentos nos cruzamentos, optou-se por utilizar um modelo de simulação de tráfego macroscópico.

Uma discussão mais detalhada sobre os diferentes níveis de detalhes da representação

[1] *Car-following* é um modelo de tráfego onde a aceleração do veículo seguidor (*follower*) é calculada em relação ao veículo dianteiro (*loader*). A aceleração é calculada com base nas velocidades relativas dos veículos, taxas de aceleração e desaceleração, tempo de reação, entre outros parâmetros pré-definidos (KRAUSS, 1998).

de uma simulação de tráfego, bem como suas aplicações é encontrada em (BAZZAN; KLÜGL, 2007) e no Capítulo 6 de (BAZZAN; KLÜGL, 2013).

3 TRABALHOS RELACIONADOS

Esta seção apresenta os principais trabalhos relacionados ao tema desta dissertação. A seção foi organizada da seguinte forma. Na Seção 3.1, são apresentados os principais trabalhos envolvendo técnicas de aprendizado por reforço para o problema de escolha de rotas. A Seção 3.2 apresenta os principais trabalhos que utilizam *difference rewards* para modelar a função de recompensa de problemas de aprendizado por reforço multiagente de outros domínios. Na Seção 3.3 são apresentadas algumas abordagens alternativas para resolver o problema de escolha de rotas.

3.1 Abordagens de Aprendizado por Reforço para Escolha de Rotas

O problema de escolha de rotas é abordado em diversos trabalhos, muitos deles envolvendo cenários abstratos. Cenários abstratos de escolha binária de rota são abordados em (BEN-ELIA; SHIFTAN, 2010; CHMURA; PITZ, 2007; KLÜGL; BAZZAN, 2004; BAZZAN et al., 2000). Em (KLÜGL; BAZZAN, 2004), um cenário com duas rotas é estudado para avaliar os efeitos de diferentes estratégias para a tarefa de escolha binária de rotas. Os autores utilizam RL para reproduzir a tomada de decisão humana, em um cenário de percurso diário. Os resultados apresentam relevantes aspectos referentes ao processo de tomada de decisão dos agentes. No entanto, as avaliações são realizadas em um cenário simples, com apenas duas rotas, um par OD e 900 veículos. Nesta dissertação, o objetivo é avaliar o problema de escolha de rotas, em problemas mais complexos e com um número superior de agentes aprendendo.

Em (TUMER; AGOGINO, 2006), os autores aplicam MARL em um cenário de escolha de rotas para reduzir os efeitos causados pelos congestionamentos. Os autores avaliam sua abordagem para duas funções de recompensa distintas. A primeira tem por objetivo maximizar a função individual dos agentes, enquanto que a segunda, chamada *difference rewards*, tem por objetivo maximizar a utilidade do sistema. Embora os resultados afirmem que a segunda abordagem apresenta resultados superiores, os modelos são avaliados em um cenário de escolha de rotas sem estado, isto é, a rota não é construída ao longo da viagem. O problema de escolha de rotas sem estado, em geral, é muito mais simples de ser resolvido, pois os agentes devem escolher uma rota entre um conjunto pré-definido, que satisfaça as restrições de seu par OD. No entanto, pode ser que o conjunto de rotas disponíveis aos agentes subutilize a rede ou omita seu potencial. Nesta dissertação, o espaço solução dos agentes não é reduzido, pois a rota é construída ao longo da viagem. Entretanto, considerando que os agentes possam andar em círculos, podem existir infinitas rotas entre um par OD, e o mecanismo de aprendizagem deve lidar com isso. Além disso, o cenário utilizado pelos autores possui 1000 veículos, e 9 rotas possíveis para 1 par OD.

Em (BAZZAN; KLÜGL, 2008) os autores utilizam um mecanismo de re-roteamento para permitir aos agentes modificar sua rota durante a viagem. Esta abordagem foi avaliada em uma rede 6×6, em forma de grade, com 30 nodos, 60 links, 700 agentes aprendendo de forma independente através do algoritmo Q-learning.

A abordagem apresentada por (TAVARES; BAZZAN, 2012) modela o problema de escolha de rotas através do uso de múltiplos aprendizes independentes e avalia-os em dois cenários distintos. O primeiro cenário possui 10 nós, 24 links e 1001 veículos distribuídos em 9 pares OD. O segundo possui 36 nós, 60 links e 300 veículos distribuídos em 3 pares OD. Para ambos os cenários, utilizou-se uma função de recompensa alinhada aos objetivos individuais dos agentes. Os resultados se mostraram inferiores à abordagem baseada em *minority game* de (GALIB; MOSER, 2011), utilizada na comparação.

Em (BAZZAN, 2014) aprendizado por reforço é aplicado ao problema de roteamento de veículos, em uma perspectiva desagregada. O cenário utilizado é uma modificação da rede apresentada em (ORTÚZAR; WILLUMSEN, 2001)[Exercício 10.1], com 1700 veículos, 13 nós e 48 links. Os agentes aprendem de forma individual, com função de recompensa alinhada aos objetivos individuais e algoritmo de aprendizado Q-learning. Os resultados são comparados com alguns dos principais métodos de alocação de tráfego. O autor conclui que a abordagem permite um alto índice de heterogeneidade nas escolhas de rotas e, consequentemente, uma melhor distribuição dos agentes sobre a rede.

Uma abordagem utilizando *learning automata* (LA) em uma perspectiva multiagente é apresentada em (RAMOS; GRUNITZKI; BAZZAN, 2014). Os autores utilizam a teoria de *learning automata* (NARENDRA; THATHACHAR, 1989), onde um motorista (autômato) aprende a escolher sua rota baseado em experiências passadas. Os agentes aprendem a escolher a melhor rota entre as K rotas disponíveis, por meio de um esquema de aprendizado chamado *linear reward-inaction*, onde as rotas são escolhidas com tempo inversamente proporcional ao seu tempo de viagem. Os resultados são comparados com os do Q-learning e alguns métodos de alocação de tráfego, no cenário abstrato apresentado em (ORTÚZAR; WILLUMSEN, 2001)[Exercício 10.1]. Resultados experimentais apresentaram resultados superiores aos métodos utilizados na comparação. Apesar desta abordagem ser baseada em RL, seu conjunto de possíveis ações está restrito ao conjunto de K rotas pré-definas que, em geral, é um número muito inferior às possíveis soluções disponíveis no roteamento link a link.

No intuito de diferenciar esta dissertação dos trabalhos existentes, as principais características do problema de escolha de rotas em abordagens RL são sumarizadas na Tab. 3.1. Como podem ser observadas na segunda coluna, todas as abordagens utilizam o mesmo algoritmo de aprendizado com o conceito de múltiplos aprendizes independentes. A terceira coluna representa a forma de alinhamento da função de recompensa. Tanto a quarta quanto a quinta coluna representam as dimensões do maior cenário utilizado para avaliar a abordagem. Em (TAVARES; BAZZAN, 2012) e (BAZZAN; KLÜGL, 2008),

apesar de o cenário possuir mais nós do que a rede utilizada nesta dissertação, os agentes possuem apenas duas ações em cada estado (pelo fato de não possuir vias com duplo sentido), enquanto que, no maior cenário desta dissertação, este número varia entre 2 e 5. A quinta coluna representa a quantidade de agentes aprendendo simultaneamente no sistema. Esta dissertação apresenta uma quantidade de agentes aproximadamente 200 vezes maior do que em (BAZZAN, 2014), que é o trabalho que utiliza a maior quantidade de agentes, dentre os comparados. De fato, a quantidade de agentes aprendendo é um grande diferencial, já que este aumento torna a tarefa de aprendizagem muito mais difícil pois existem muitos agentes aprendendo e adaptando suas políticas simultaneamente.

Abordagem	**Algoritmo**	**Alinhamento de R**	**Nós**	**Links**	**Re-roteamento?**	**Agentes**
Klügl e Bazzan (2004)	Q-learning	local	2	2	não	900
Bazzan e Klügl (2008)	Q-learning	local	36	60	sim	700
Tavares e Bazzan (2012)	Q-learning	local	36	60	sim	1001
Ramos et al (2014)	L. Rew. Inac.	local	13	48	não	1700
Bazzan (2014)	Q-learning	local	13	48	sim	1700
Tumer e Agogino (2006)	Q-learning	local e global	2	9	não	1000
Este trabalho	Q-learning	local e global	24	76	sim	360600

Tabela 3.1: Aspectos desta dissertação e trabalhos relacionados a RL para o problema de escolha de rotas.

3.2 Abordagens de Aprendizado por Reforço para Outros Problemas

Além das abordagens apresentadas na seção anterior, existem inúmeras outras que utilizam aprendizado por reforço para resolver problemas nos mais variados domínios. A maior parte dessas abordagens faz uso do alinhamento local para modelar a função de recompensa. Dentre as abordagens que utilizam o alinhamento global para resolver problemas de outros domínios, destacam-se trabalhos como (DEVLIN et al., 2014; HOLMESPARKER et al., 2014; TUMER; KHANI, 2009; TUMER; AGOGINO, 2007; TUMER; WOLPERT, 2004). Todos estes trabalhos são desenvolvidos com a autoria ou coautoria do pesquisador da Universidade do Estado de Oregan, Kagan Tumer. Isso ocorre pelo fato de que ele é um dos criadores da técnica *difference rewards* e o principal pesquisador nesta linha. A seguir, os trabalhos anteriormente citados serão brevemente discutidos.

Em (TUMER; WOLPERT, 2004) a função *difference rewards* é apresentada dentro do *framework* de inteligência coletiva apresentado no trabalho. Neste trabalho, o função ainda era apresentada com a sua antiga demoninação, isto é, *wonderful life utility*. Apesar de não representar mais o estado da arte, o trabalho apresentou uma teoria bastante abrangente sobre o assunto, além de um compilação de resultados com a técnica em trabalhos passados. Dentre as aplicações, podemos destacar problemas de: roteamento de pacotes em redes de dados, controle de comunicação entre satélites, *minority games*,

além de uma série de problemas de busca. A partir dos resultados apresentados, foi constatado que em todos os problemas apresentados, o alinhamento global da função de recompensa é capaz de trazer benefícios para o sistema como um todo.

O probelma de controle de tráfego aéreo é abordado como um problema de aprendizado por reforço em (TUMER; AGOGINO, 2007). Os autores utilizam um simulador de tráfego aéreo desenvolvido pela NASA - agência do governo dos Estados Unidos da América responsável pela pesquisa e desenvolvimento de tecnologias e programas de exploração espacial -, para simular o problema de aprendizagem. Neste trabalho, os agentes (aéronaves) precisam aprender o espaço que devem manter entre eles e os demais agentes que utilizam o espaço aéro. O objetivo é minimizar o congestionamento do espaço aéreo. Resultados demonstraram que agentes modelados com a função de recompensa *difference rewards* são capazes de reduzir o congestionamento do espaço aéreo em até 67%, quando comparados aos métodos tradicionais utilizados pela indústria.

Uma variante da função *difference rewards* é proposta em (TUMER; KHANI, 2009), para reduzir do tempo de aprendizagem no problema *El farol bar*. Os autores modificam a função para que seja possível estimar a recompensa a ser recebida em ações não realizadas individualmente pelos agentes. Com base nesta previsão de recompensa, os agentes conseguem reduzir o tempo necessário para aprender um comportamento porque há uma redução considerável na exploração do espaço de busca.

Um segundo trabalho utilizando a função *difference rewards* para reduzir o tempo de aprendizado é apresentado em (DEVLIN et al., 2014). Os autores incorporam a função a uma técnica de inserção de conhecimento de domínio, conhecida como *potential-based reward shaping*. Os autores aplicam a técnica no *congestion game*, *beach problem domain*. Neste problema, os agentes (banhistas) precisam se coordenar para aprender em que lugar da praia desejam passar o dia. A recompensa dos agentes é maximizada quando a capacidade do espaço onde eles estão dispostos é respeitada. Deste modo, os agentes precisam aprender a se distribuir da melhor forma possível. A técnica proposta pelos autores se mostrou bastante eficiênte, conseguindo reduzir o tempo de aprendizagem em até 23.8 vezes, quando comparada a abordagens tradicionais de aprendizado por reforço. Além disso, as soluções obtidas pelos autores se mostraram 196% melhores do que as demais abordagens utilizadas na comparação.

No trabalho (HOLMESPARKER et al., 2014), os autores também utilizam uma modificação da função *difference rewards* para remover os efeitos dos demais agentes na função de recompensa. Deste modo, os agentes recebem uma recompensa "limpa", ou seja, sem o ruído causado pelas ações dos demais agentes. Como consequência, ações que poderiam ser consideradas ruins devido as ações dos demais agentes, podem ser estimuladas pela função proposta pelos autores. Os resultados são avaliados no *Gaussian Squeeze Domain*, problema onde os agentes precisam se coordenar para aprenderem a maximizar a capacidade de uma dada função objetivo. Os autores variaram a quantidade de agentes entre

100 e 1000 e concluem que os benefícios da abordagem são escalaveis em função do número de agentes aprendendo.

3.3 Abordagens Alternativas para Escolha de Rotas

O problema de alocação de tráfego é abordado por meio de algoritmos genéticos em (CAGARA; BAZZAN; SCHEUERMANN, 2014; CAGARA; SCHEUERMANN; BAZZAN, 2013; BURIOL et al., 2010). Nestas abordagens, cada veículo possui K rotas possíveis, cuja codificação é feita de forma binária. Os cromossomos dos indivíduos (possíveis soluções) representam as rotas atribuídas a cada veículo. Nestas abordagens o motorista não pode alterar seu conjunto inicial de rotas, para melhorar seu tempo de viagem. Além disso, as rotas não podem ser trocadas durante a viagem, mesmo que haja interseção entre elas. Nesta dissertação, ambas as questões são apropriadamente abordadas.

Os métodos *incremental assingment* e *successive averages assignment* - apresentados na Seção 2.4.2 - são aplicados em um cenário abstrato, em (ORTÚZAR; WILLUMSEN, 2001). Uma grande desvantagem destes métodos, quando comparados a abordagens baseadas em agentes, é que o processo de escolha de rotas em TA é feito por uma autoridade central, a qual possui observação global do estado do sistema. Em simulações baseadas em agentes, como abordado neste trabalho, o processo de escolha de rotas é feito de forma individual e descentralizada. Apesar disso, estes métodos são muito utilizados como *baseline* para comparação de métodos. Outras abordagens baseadas em alocação de tráfego são apresentadas em (SHIGEHIRO; MIYAKAWA; MASUDA, 2012; TONG; WONG, 2000).

4 ABORDAGEM PROPOSTA

Como apresentado nos capítulos anteriores, fazer a conexão entre demanda e oferta é um objeto de pesquisa recorrente na literatura. Várias abordagens têm sido propostas para resolver este problema. Entretanto, devido ao conjunto de restrições por elas impostas, a aplicação destas em cenários reais é questionável. Um exemplo de restrição, presente em abordagens de computação evolutiva ou TA, é a necessidade de uma autoridade central para atribuir rotas. Em cenários reais, isto implicaria em completa observação do estado da rede e comunicação entre a autoridade central e demanda. Embora a principal finalidade destes métodos seja avaliar os efeitos do tráfego na infraestrutura, por tratar do processo de tomada de decisão forma individual, abordagens baseadas em agentes se mostram muito mais robustas e factíveis de serem aplicadas no mundo real.

Diante deste cenário, este trabalho utiliza simulação baseada em agentes e aprendizado por reforço multiagente para realizar a conexão entre demanda e oferta, de forma inteligente e distribuída. As viagens da demanda são representadas por agentes capazes de aprender por reforço, e a rede de tráfego é modelada como o ambiente dos agentes. A tarefa de aprendizagem dos agentes é encontrar a rota com custo mais atraente, que satisfaça suas restrições de origem e destino. Apesar de esta tarefa ser usualmente referida na literatura como “escolha de rotas”, esta abordagem utiliza a denominação “aprendizado de rotas” para a tarefa de aprendizagem dos métodos propostos. A motivação para esta distinção é que neste trabalho, os agentes constroem uma rota que conecta sua origem e destino a partir de suas motivações baseadas em experiências passadas, enquanto em abordagens, como TA ou computação evolutiva os motoristas escolhem uma rota entre um conjunto de rotas previamente definido.

Neste trabalho, apresentamos duas abordagens para resolver o problema de aprendizagem de rotas, cuja diferença está no alinhamento da função de recompensa. Alinhar a função de recompensa, neste contexto, significa vincular o sinal numérico enviado pelo ambiente a algum interesse. Utilizamos duas formas de alinhamento de função de recompensa. A primeira alinha a função de recompensa à utilidade do agente, enquanto a segunda alinha a função de recompensa à utilidade global do sistema. A utilidade neste contexto é uma medida de satisfação. Para o problema de aprendizagem de rotas, um agente tem sua utilidade maximizada quando seu tempo de viagem na rede é reduzido, já o sistema tem sua utilidade maximizada quando o tempo médio de viagem na rede é reduzido.

Este capítulo está organizado da seguinte forma. A Seção 4.1 apresenta a modelagem dos agentes e do seu processo de tomada de decisão. As duas funções de recompensa são apresentadas na Seção 4.1.1. A Seção 4.2 apresenta como a simulação de aprendizado de rotas em rede de tráfego é realizada.

4.1 Modelagem dos Agentes

Neste trabalho, cada viagem da matriz OD é representada por um agente $d \in D$ (também chamado de veículo ou motorista). Cada agente d possui um nó de origem $n_d^{\uparrow}$ e um nó de destino $n_d^{\downarrow}$. O objetivo do agente é aprender uma rota r_d, que conecte sua origem com o seu destino, de modo que o custo total de viagem y_d seja minimizado.

O processo de tomada de decisão dos agentes é modelado através do processo de decisão de Markov. Devido à existência de um elevado número de agentes aprendendo simultaneamente em problemas de roteamento de veículos, optou-se por desconsiderar a existência dos demais agentes na modelagem do MDP. Isto é necessário devido à explosão do espaço de busca causado pelo uso de ações conjuntas. Desta forma, utiliza-se a técnica de múltiplos aprendizes independentes (ver Seção 2.3.4.1) para viabilizar a simulação deste processo com muitos agentes aprendendo (cenários com até 360600 agentes são avaliados neste trabalho).

Considerando que todos os agentes estão situados em uma rede viária $G(V, E)$ em comum, o MDP individual dos agentes é modelado da seguinte forma. Os estados do MPD são representados pelos nós da rede, e as ações são representadas pelos links que conectam os nós. Por exemplo, o estado de um agente d é um nó $s \in S$ onde o agente está situado. O conjunto de ações para este estado $A(s)$ é composto pelo conjunto de links saintes deste nó. A Fig. 4.1 ilustra uma situação hipotética, onde o agente d, situado no estado s_3, possui as seguintes ações disponíveis: $A(s_3) = \{a_1, a_2, a_3\}$. Este exemplo considera que todos os links da rede possuem duas vias (uma para cada sentido), mas isto é variável em função da topologia da rede. Além disso, considera-se que o agente possui o modelo de mundo completo, isto é, o agente possui o mapeamento completo da topologia da rede em seu MPD. Fazendo uma analogia com o mundo real, isto é totalmente aceitável nos dias atuais devido à popularização dos sistemas de navegação por satélite, presentes em dispositivos específicos ou smartphones.

O processo de aprendizagem de rotas é realizado pelos agentes de forma iterativa e individual. A cada iteração, os agentes escolhem e adicionam um novo nó em sua rota. Lembrando que uma rota é composta por um conjunto de nós conectados por links. Assim, ao percorrer o link a_d que conecta o seu estado atual s_d e o estado futuro s'_d, este nó estado futuro é adicionado a rota do agente. Este processo é realizado em duas etapas, uma de escolha de ação e outra de atualização da base de conhecimento.

Um algoritmo para a escolha de ação é apresentado em Alg. 3. Neste algoritmo um agente d, situado em um estado s_d, selecionada a ação a_d que o levará para um estado futuro s'_d. Esta escolha ação é realizada através da política de exploração ϵ-gulosa. Como se pode perceber no Alg. 3, a escolha de ação é influenciada por uma taxa de exploração ϵ e um valor entre $[0, 1]$, gerado a partir de uma função geradora de números aleatórios. Quando a restrição da linha 2 for satisfeita, a ação a_d será escolhida de forma

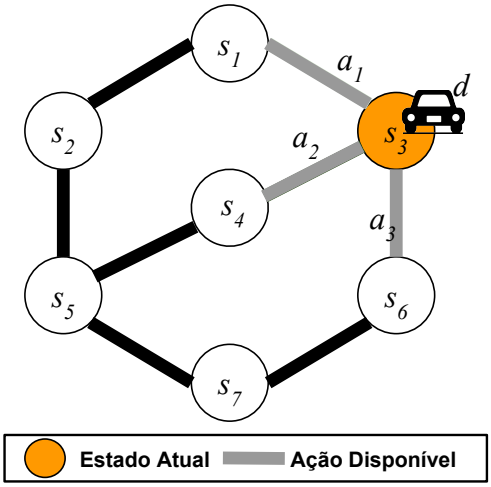


Figura 4.1: Mapeamento dos estados (S) e ações (A) do agente

aleatória entre o conjunto de ações disponíveis no MDP (Q_{s_d}). Caso contrário o algoritmo selecionará a ação com maior retorno esperado.

Algoritmo 3: Algoritmo CHOOSEACTION(d, ϵ) para escolha de ação do agente.

Entrada: d `/* um agente */`
Entrada: ϵ `/* fator de exploração */`
Saída: a_d `/* a próxima ação */`

1 **início**
2 **se** $\epsilon \geq newRandom()$ **então**
3 $a_d = selectRandom(A(s_d))$; `// exploração`
4 **senão**
5 $a_d = \arg\max_{a \in A(s_d)} Q(s_d, a)$; `// aproveitamento`
6 **fim**
7 **fim**

De posse da nova ação a_d, o agente irá executá-la, ou seja, trafegar pelo link da rede viária correspondente a ação a_d. Em seguida, a segunda etapa do processo de aprendizagem de rotas deverá ser executada. O Alg. 4 apresenta um método para realizar a atualização do MDP do agente. Neste algoritmo, o agente d atualiza a sua base de conhecimento (tabela Q), de acordo com a regra de atualização do algoritmo Q-learning (linha 3). Esta equação é sensível a dois parâmetros: uma taxa de aprendizagem (α) e um fator de desconto (γ). A recompensa r é recebida do ambiente por realizar a ação a_d no estado s_d, de acordo com a função de recompensa utilizada (descrita na próxima Seção). Em seguida o estado atual do agente s_d (sua posição na rede) é atualizado, a rota r_d é incrementada como o novo nó atual e o tempo total de viagem é incrementado de acordo com o tempo de viagem t_{a_d} correspondente ao custo por trafegar no link correspondente a ação a_d.

Algoritmo 4: Algoritmo UPDATETAKENACTION(α, γ, d) para atualização do MDP do agente.

Entrada: α; /* taxa de aprendizagem */
Entrada: γ; /* fator de desconto */
Entrada: d; /* agente */

início
  $r \leftarrow R(s_d, a_d)$;
  $Q(s_d, a_d) \leftarrow (1-\alpha)\, Q(s_d, a_d) + \alpha[r + \gamma \max_{a'_d} Q(s'_d, a'_d)]$;
  $s_d \leftarrow s'_d$;
  $r_d \leftarrow \{r_d, s_d\}$;
  $y_d \leftarrow y_d + t_{a_d}$;
fim

4.1.1 Funções de Recompensa

Esta seção apresenta as funções de recompensa utilizadas na modelagem dos agentes aprendizes por reforço. Duas funções de recompensa são utilizadas neste trabalho: a função individual (Seção 4.1.1.1) e a função *difference rewards* (Seção 4.1.1.2). Como o algoritmo de aprendizado empregado é o Q-learning, para facilitar a distinção entre os agentes com funções de recompensa distinta, daqui em diante será utilizada a nomenclatura IQ-learning para fazer referencia à abordagem que utiliza a função de recompensa individual, e DQ-learning para a abordagem que utiliza a função de recompensa *difference rewards*. As seções subsequentes apresentam a formulação das funções de recompensa.

4.1.1.1 Função Individual

Esta função é chamada de função individual porque se baseia apenas no link atual do agente para estimar o valor da recompensa. Sua formalização é dada pela Eq. 4.1, onde t_a é o tempo em minutos que o agente gastou para viajar no link a, em função do fluxo v_a existente. Como esta função retorna um sinal negativo, a recompensa do agente aumenta quando o tempo de viagem diminui. Desta forma, os agentes irão se deslocar na rede de modo que seu tempo de viagem individual seja minimizado. Esta função é utilizada pela abordagem IQ-learning e está alinhada a utilidade individual do agente. Lembrando que a utilidade/satisfação do agente é maximizada quando o seu tempo de viagem é reduzido.

$$R(s, a) = -t_a(v_a) \tag{4.1}$$

Desta forma, o sinal enviado por esta função indica ao agente a qualidade de uma ação, com base na influência desta ação com a utilidade do agente. Como consequência, todos os agentes IQ-learning irão atuar sob o ambiente de forma individualista, e, portanto, com uma constante adaptação da rota para o caminho menos custoso. Esta adaptação torna

os agentes egoístas, mas devido ao fato de que todos são capazes de adaptar suas rotas de maneira individual, pode ser que o sistema convirja para soluções indesejáveis, pois os agentes estarão constantemente competindo por recursos.

Uma grande vantagem proporcionada por esta função é que para efetuar o cálculo da recompensa, basta que o agente tenha apenas a percepção sobre a situação do trânsito no seu link atual. Pensando em um cenário real, é exatamente esta percepção que motiva um motorista a utilizar determinada rota, quando está trafegando sem o auxílio de sistemas de navegação, como o Google Maps[1] ou Waze[2] - estes sistemas são capazes de apresentar informações aproximadas sobre a utilização da rede, com base em informações enviadas pela comunidade de usuários.

4.1.1.2 Função Difference Rewards

A função de recompensa utilizada pelos agentes DQ-learning é baseada na função *difference rewards*, apresentada em (TUMER; AGOGINO, 2007) (ver Seção 2.3.6). Optou-se pelo uso desta função porque modelando as recompensas desta forma, espera-se que os agentes não atuem de forma egoísta, como ocorre nos agentes IQ-learning. Como consequência, espera-se que os agentes aprendam rotas que beneficiem o sistema como um todo, ao invés de rotas que maximizem apenas a recompensa individual dos agentes.

O cálculo da função é realizado de acordo com a Eq. 4.2, onde $G(z)$ representa a recompensa de todos os agentes em um determinado momento, e $G(z_{-d})$ representa a recompensa que todos os agentes deveriam receber, caso o agente d não estivesse na simulação. O cálculo de $G(z)$ é realizado através da Eq. 4.3, onde z é um termo geral utilizado para representar os pares estado-ação de todos os agentes, t_j é o tempo atual de viagem no link j e v_j é o fluxo de veículos. O cálculo de $G(z_{-d})$ é representado pela Eq. 4.4, onde z_{-d} representa os pares estado-ação de todos os agentes menos o agente d e j_d é o link onde o d está situado. Este cálculo é semlhante ao da Eq. 4.3, mas desconsidera os efeitos da ação do agente d em z.

$$R = G(z) - G(z_{-d}) \tag{4.2}$$

$$G(z) = \sum_{j \in J} -t_j(v_j)\, v_j \tag{4.3}$$

$$G(z_{-d}) = \sum_{j \in J \setminus j_d} [-t_j(v_j)\, v_j] - [t_{j_d}(v_{j_d} - 1)\,(v_{j_d} - 1)] \tag{4.4}$$

A função de recompensa quando modelada desta maneira, estima à recompensa do

[1] `https://maps.google.com.br`
[2] `https://www.waze.com/`

agente com base na sua contribuição individual para o desempenho do sistema. A utilidade do sistema é maximizada na medida em que o tempo de viagem na rede diminui. Desta forma, alinhando-se a função de recompensa do agente à utilidade do sistema, os agentes serão estimulados a tomar ações que contribuem para a redução do tempo de viagem na rede.

Uma questão muito importante no que se refere ao uso desta função é a necessidade de total observação do ambiente. Para que seja possível estimar a influência do agente no desempenho do sistema é necessário ter a informação sobre o tempo de viagem em todos os links. Em cenários reais, podemos fazer uma analogia deste sinal numérico a um sistema de navegação. Porém, neste caso, o sistema apresentaria a rota que fosse mais benéfica para todo o sistema.

4.2 Simulação de Aprendizado de Rotas

Para que seja possível simular o processo de aprendizagem de rotas de motoristas em redes viárias, cujo modelo básico foi apresentado nas seções anteriores, é necessário realizar simulações de tráfego. A simulação é executada de acordo com o Alg. 5. Antes de dar início ao processo de aprendizagem é necessário criar um agente para cada viagem presente na matriz OD (linha 5).

A criação do agente, inicialização do seu MDP, nodo de origem $n_d^{\uparrow}$, nodo de destino $n_d^{\uparrow}$, rota r_d e tempo de viagem y_d são realizados através do Alg. 6. Após os agentes serem devidamente criados, inicia-se a simulação. Esta simulação é realizada de forma episódica, onde μ representa o número máximo de episódios permitidos.

A cada episódio os veículos podem executar o processo de aprendizagem de rotas por τ passos de tempo. Caso o veículo não encontre uma rota que conecte sua origem com seu destino dentro desta quantidade de passos, ele voltará para o seu ponto de origem no próximo episódio. A cada passo de tempo, os agentes executam duas ações. Na primeira etapa do processo de aprendizagem, cada agente $d \in D^{\bullet} \subseteq D$ (onde $D^{\bullet}$ é o conjunto de agentes que ainda não chegaram no destino) escolhe uma ação a_d, com base em seu estado atual s_d. Esta ação é selecionada de acordo com o Alg. 3.

Depois de realizada a etapa de escolha da próxima ação, os agentes transitarão do estado s_d para o estado s'_d devido à execução da ação tomada a_d. Esta transição de estados é um deslocamento de um nó (s_d) da pra outro (s'_d) através de um link (a_d). Consequentemente, a transição de estados simultânea de todos os agentes irá alterar os fluxos v_l dos links l da rede. Com base nestes novos fluxos, na linha 20 do Alg. 5, os tempos de viagens em todos os links são atualizados.

Realizada a transição de estados de todos os agentes, e de posse das novas condições da rede viária, a tabela Q dos agentes deve ser atualizada (Alg. 5 linha 23). Após todos os agentes completarem sua viagem ou o critério de parada da linha 13 do Alg. 5 ser

satisfeita, a nova taxa de exploração ϵ deve ser atualizada. Esta taxa é atualizada com base em dois parâmetros, uma taxa de exploração inicial ϵ_0 e uma taxa de decaimento ϵ_f. Desta forma, a taxa de exploração inicial é reduzida linearmente ao longo dos episódios (Alg. 5 linha 26). Após esta atualização da taxa de exploração, os veículos são reiniciados pela função RESETDRIVERS(). Esta função reinicia os parâmetros r_d, y_d e insere o agente novamente em seu nó inicial ($s_d = n_d^{\uparrow}$), mas não altera as entradas da tabela Q. Após a reinicialização o processo de aprendizagem de rotas é executado novamente, até que se conclua o número máximo de episódios. Ao atingir o limite de episódios, a simulação é finalizada.

Como se pode observar, a execução desta simulação é exatamente igual para as duas abordagens propostas. Desta forma, tanto os agentes que utilizam o IQ-learning quanto os que utilizam o DQ-learning realizarão este mesmo processo de simulação. Entretanto, em cada abordagens os agentes recebem recompensas do ambiente de acordo com a definição da abordagem.

Algoritmo 5: SIMULACAO$\Big(T, G, \epsilon_0, \epsilon_f, \epsilon, \alpha, \gamma\Big)$

Entrada: G /* Um grafo com $|V|$ vértices e $|A|$ arestas */
Entrada: T /* Matriz OD com I origens e J destinos */
Entrada: $\epsilon, \epsilon_0, \epsilon_f$ /* Parâmetros de exploração */
Entrada: α, γ /* taxa de aprendizagem e fator de exploração */

início
  para todo *nó de origem* $i \in I$ **faça**
    para todo *nó de destino* $j \in J$ **faça**
      para todo $t \in T_{ij}$ **faça**
        $D \leftarrow$ CREATEDRIVER(G, i, j)
      fim
    fim
  fim
  $i \leftarrow 0$; // contador de episódios
  enquanto $i < \mu$ **faça**
    RESETDRIVERS(D);
    $j \leftarrow 0$; // contador de passos de tempo
    enquanto $j < \tau$ OU $D^{\bullet} \neq \oslash$ **faça**
      $j \leftarrow j + 1$;
      para todo $d \in D^{\bullet}$ **faça**
        $a_d \leftarrow$ CHOOSEACTION(d, ϵ);
        $s_d \leftarrow s'_d$
      fim
      para todo $l \in L$ **faça**
        UPDATETRAVELTIME(t_l);
      fim
      para todo $d \in D^{\bullet}$ **faça**
        UPDATETAKENACTION(d, α, γ);
      fim
    fim
    $\epsilon = \epsilon_0 \left(\epsilon_f\right)^i$;
    $i \leftarrow i + 1$;
  fim
fim

Algoritmo 6: Algoritmo CREATEDRIVER(d, G, n_i, n_j) para criar um novo motorista.

Entrada: G; /* Um grafo com $\mid S \mid$ vértices e $\mid A \mid$ arestas */
Entrada: n_i; /* Um nó de origem */
Entrada: n_j; /* Um nó de destino */
Saída: d; /* Um motorista */

```
1  início
2      para todo n ∈ N faça
3          para todo l ∈ L_n faça
4              Q_d(n, l) ← 0;
5          fim
6      fim
7      n_d^↑ ← n_i;
8      n_d^↓ ← n_j;
9      r_d ← {n_i};
10     y_d ← 0;
11     s_d ← n_i;
12 fim
```

5 EXPERIMENTOS

Este capítulo apresenta os experimentos realizados para validar as abordagens propostas nesta dissertação. São utilizados dois cenários que diferem em topologia, quantidade de agentes e, consequentemente, complexidade (do ponto de vista da tarefa de aprendizagem). O primeiro cenário possui 1700 agentes e é apresentado na Seção 5.2. Já o segundo, possui 360600 agentes e é apresentado na Seção 5.3. A seção 5.1 apresenta a escolha de parâmetros.

5.1 Parâmetros e Métodos Utilizados na Comparação

Antes de testar a efetividade do IQ-learning e DQ-learning, algumas simulações foram executadas neste cenário para avaliar as configurações dos parâmetros α, γ e ϵ do algoritmo de aprendizado por reforço. A métrica utilizada para avaliar estes parâmetros foi o tempo médio de viagem na rede, o qual é representado pelo tempo médio que todos os veículos levaram para concluir sua viagem.

A escolha dos parâmetros α e γ foi efetuada de maneira individual, para ambos os métodos propostos. Empiricamente, o conjunto de parâmetros que convergiu para melhores soluções foi $\alpha = 0.8$, $\gamma = 0.9$ para ambos os métodos. Neste problema, o algoritmo de aprendizado é mais sensível ao fator de desconto do que a taxa de aprendizagem. Para baixos valores de γ os agentes não conseguem aprender boas rotas para seu destino. Isto ocorre porque em problemas de aprendizagem de rotas, onde a próxima opção de escolha é dependente apenas da ação atual, é muito importante levar em consideração a saída das escolhas subsequentes, caso contrário, as próximas possibilidades de ações podem levar a soluções não desejáveis. Apesar disso, α não causa muito impacto na qualidade da solução encontrada.

Para a política de exploração ϵ-gulosa, buscou-se uma configuração que permitisse aos agentes explorar nos episódios iniciais com uma alta probabilidade e aproveitar com uma probabilidade muito alta nos episódios finais. Desta forma, ϵ é inicializado por $\epsilon_0 = 1.0$ e multiplicado por um fator $\epsilon_f = 0.92$ a cada episódio. O número de episódios e número máximo de passos por episódio foram definidos em $\mu = 150$ e $\tau = 100$, respectivamente. Desta forma, os agentes iniciam a simulação com uma taxa de exploração de 100% e a decrementam gradativamente, terminando a simulação com uma taxa de exploração $\cong 0\%$. A curva de exploração é apresentada na Fig. 5.4. Estes parâmetros foram definidos empiricamente. Outras configurações também foram avaliadas, mas os testes empíricos demonstraram que os resultados obtidos com 150 e 500 episódios são equivalentes. Sendo assim, optou-se por esta configuração por utilizar poucos episódios, que se traduz em menos tempo computacional para realizar os experimentos (embora esta métrica não seja relevante para avaliar o desempenho das abordagens).

Os resultados das abordagens propostas são comparados entre si e com três abordagens clássicas de TA, são elas: *successive averages assignment*, *incremental assignment*, e *all-or-nothing assignment*. Para o método *successive averages assignment* utilizou-se como critério para atualização dos fluxos $\phi = \frac{1}{n}$ e 100 iterações como critério de parada. Para o *incremental assignment* utilizou-se $p_n = 0.4, 0.3, 0.2, 0.1$.

5.2 Cenário OW: 1700 Agentes

Este cenário é uma adaptação do problema apresentando no Exercício 10.1 de (ORTÚZAR; WILLUMSEN, 2001) (cenário OW). A topologia da rede de tráfego é ilustrada na Fig. 5.1 e possui 13 nós e 24 links. Cada link possui duas vias, uma para cada sentindo. O valor sobre cada link, na Fig. 5.1 representa o tempo de viagem sem fluxo (t_{l_0}) para ambos os sentidos. A rede possui duas zonas de origens (A e B), representando duas áreas residenciais, e duas zonas de destino (L e M), representando dois shoppings centers.

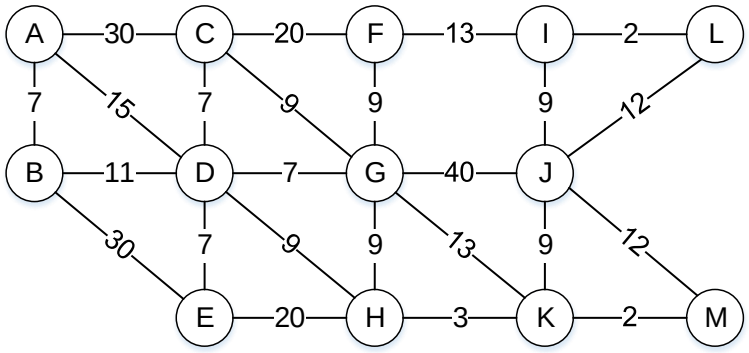


Figura 5.1: Topologia da rede OW adaptada de (ORTÚZAR; WILLUMSEN, 2001).

Assim como em (ORTÚZAR; WILLUMSEN, 2001), a matriz OD utilizada neste trabalho possui 1700 viagens distribuídas em 4 pares OD, como mostra a Tab. 5.1. A função de custo proposta pelos autores representa o tempo de viagem no link em minutos e é representada por:

$$t_l = t_{l_0} + 0.02 * q_l \tag{5.1}$$

onde, t_l é o tempo de viagem no link l, t_{l_0} é o tempo de viagem em l sem fluxo e q_l é o fluxo do link. Esta é uma versão mais simplista da VDF (Eq. 2.7), na qual o tempo de viagem é incrementado em 0.02 por cada veículo do fluxo.

Este cenário, apesar de parecer simples, apresenta uma série de características relevantes para a análise de desempenho dos algoritmos de aprendizado por reforço. Por exemplo, o cenário captura algumas propriedades importantes do mundo real como o compartilhamento de links entre rotas distintas. Apesar disso, para tornar a tarefa de

origem	destino	viagens
A	L	600
A	M	400
B	L	300
B	M	400
	total	1700

Tabela 5.1: Pares OD da rede OW (ORTÚZAR; WILLUMSEN, 2001, Exercício 10.1).

aprendizagem de rotas um pouco mais complexa, modificações nos custos de alguns links foram realizadas. Como na rede original os pesos dos links são muito semelhantes, os pesos dos links AC, BE, CF, EH e GJ (e direções opostas) foram significativamente aumentados para torná-los menos atrativos. Deste modo, o uso de links menos atrativos é indesejável por parte dos agentes, gerando uma maior competição por links mais atrativos. Para demonstrar os efeitos destas modificações, a Tab. 5.2, apresenta a rota de menor caminho e respectivo custo sem fluxo, para cada um dos pares OD, das versões original e modificada da rede. De acordo com os resultados apresentados na Tab. 5.2, houve um aumento significativo no tempo de viagem de todos os pares OD na versão modificada. Além disso, se pode notar que em todos os casos as rotas foram alteradas.

par OD	original		modificada	
	menor caminho	custo	menor caminho	custo
AL	A >C >G >J >I >L	28	A >D >G >F >I >L	46
AM	A >C >D >H >K >M	26	A >D >H >K >M	29
BL	B >D >G >J >I >L	32	B >D >G >F >I >L	42
BM	B >E >H >K >M	23	B >D >H >K >M	25

Tabela 5.2: Menor caminho e custo sem fluxo para os 4 pares OD da rede OW (ORTÚZAR; WILLUMSEN, 2001, Exercício 10.1) (versão original e modificada).

5.2.1 Resultados Experimentais

Nesta seção, são apresentados os experimentos realizados para avaliar o desempenho das abordagens IQ-learning e DQ-learning na rede OW. São avaliadas três métricas para a rede OW. A primeira, apresentada na Seção 5.2.1.1, mensura o tempo médio de viagem na rede e por par OD em minutos. A segunda, apresentada na Seção 5.2.1.2, avalia o percentual da infraestrutura da rede utilizada. Já a terceira, apresentada na Seção 5.2.1.3, avalia a curva de aprendizagem das abordagens propostas.

5.2.1.1 Tempo Médio de Viagem na Rede

Para realizar uma comparação entre os métodos propostos, neste experimento, avaliou-se o desempenho das abordagens através do tempo médio de viagem na rede, bem como o tempo médio de viagem por par OD. O tempo médio de viagem na rede (ou tempo médio de viagem), representa a média de tempo, em minutos, que todos os veículos levaram para completar sua viagem. O tempo médio por OD, representa a média de tempo que todos os veículos de um mesmo par OD levaram para completar sua viagem. Cada simulação foi executada 30 vezes. Para fins de comparação, neste mesmo cenário, avaliou-se as soluções obtidas com os métodos *successive averages assignmentincremental assignment* e *all-or-nothing*, apresentados na Seção 2.4.2.

A Tab. 5.3, apresenta os resultados do experimento mencionando anteriormente. Por se tratar de métodos determinísticos, os métodos de TA não apresentam desvio padrão (em parênteses). Apesar do método *all-or-nothing* apresentar os piores resultados, ele serve como um *baseline* para avaliar a qualidade dos demais métodos. Os métodos iterativos de TA (*incremental assignment* e *successive averages assignment*) apresentaram resultados muito próximos, mas superiores ao *all-or-nothing* em todos os casos. Através da análise dos experimentos realizados, é possível afirmar que as abordagens propostas apresentam resultados superiores a todos os demais métodos, para todos os casos utilizados na avaliação. Isto demonstra que a possibilidade de adaptar a rota durante a viagem, disponível nas abordagens baseadas em agentes, garante uma melhor exploração do espaço de busca do que os demais métodos. Entre as abordagens baseadas em agente, a que atingiu os melhores resultados foi o DQ-learning. Além de ter melhorado o tempo médio de viagem, o tempo de viagem de todos os pares OD também foram melhorados, quando comparados aos resultados do IQ-learning. Isto demonstra que, apesar de ambos os métodos apresentarem resultados muito superiores aos demais, o alinhamento da função de recompensa dos agentes à utilidade do sistema, garante uma melhor adaptação por parte dos motoristas.

par OD	viagens	DQ-learning	IQ-learning	inc. ass.	suc. avg.	all-or-noth.
AL	600	79.07 (0.27)	81.25 (0.23)	96.56	101.18	138.00
AM	400	64.41 (0.31)	67.86 (0.35)	90.40	86.49	97.00
BL	300	76.43 (0.44)	78.56 (0.25)	94.72	97.80	128.00
BM	400	61.80 (0.33)	64.32 (0.40)	85.68	83.04	87.00
TMV	1700	71.09 (0.13)	73.64 (0.23)	92.22	92.86	114.59.00

Tabela 5.3: Tempo Médio de Viagem (TMV) e tempo médio por par OD na rede OW. Valores entre parênteses representam o desvio padrão.

5.2.1.2 Fluxo de Veículos na Rede

O objetivo deste experimento é avaliar, para cada um dos métodos propostos, como os veículos são distribuídos na rede. Devido ao uso de simulação macroscópica neste trabalho, métricas como taxa de ocupação dos links não são realistas, pois o tempo é discreto e a simulação permite que sejam inseridos veículos além da capacidade dos links. Isto posto, utiliza-se como métrica o fluxo de veículos por intervalo de tempo. Neste caso, o intervalo de tempo é a duração de uma simulação completa. Os resultados obtidos com os métodos propostos foram comparados com os métodos *incremental assignment*, *successive averages* e *all-or-nothing*.

A Tab. 5.4 apresenta os fluxos de cada link e seu respectivo custo para cada um dos métodos avaliados. Para os métodos IQ-learning e DQ-learning, estes resultados são equivalentes a uma única simulação. Não foram utilizados resultados médios porque em tarefas de aprendizado multiagente pode não haver uma única solução, logo, para este domínio específico, duas soluções semelhantes podem distribuir os veículos sobre a rede de modo distinto. Os resultados apresentados para os métodos de RL foram escolhidos arbitrariamente do conjunto de 30 execuções realizadas no experimento anterior.

Como se pode observar na Tab. 5.4, os métodos de TA acabaram por alocar a demanda de forma ineficiente na rede. A grosso modo, os métodos de TA concentram grandes fluxos nos links que, em geral, formam as rotas de menor custo (sem fluxo). A última linha apresenta o percentual de links utilizados pela demanda. Percebe-se que o método que utilizou a maior quantidade de links (DQ-learning), é o mesmo que obteve os menores tempos de viagem (ver Tab. 5.3) na rede. Observando a Tab. 5.4 com mais atenção, percebe-se que, em muitos casos, como nos links BA e CD, os métodos de TA acabam por não utilizar estes links, enquanto as abordagens baseadas em agentes utilizando-os, mesmo que por poucos agentes (como no caso de BA).

Comparando os resultados obtidos com o DQ-learning e IQ-learning, pode-se atribuir o motivo pelos melhores resultados do DQ-learning a uma melhor distribuição da demanda sobre a rede. O DQ-learning utiliza $\approx 33.3\%$ a mais da infraestrutura disponível do que IQ-learning. Pensando do ponto de vista do alinhamento da recompensa à utilidade do sistema, este comportamento emergente é totalmente compreensível, já que alguns veículos podem optar por utilizar caminhos alternativos e menos atrativos para contribuir com a redução do tempo de viagem do coletivo. Ao contrário disso, no IQ-learning, os agentes acabam competindo por um número reduzido de recursos (links), mas com resultados ainda bastante atrativos, quando comparados aos dos métodos de TA.

		DQ-learning		IQ-learning		inc. assig.		succ. avg.		all-or-noth.	
link	c.s.f.	fluxo	custo	fluxo	custo	fluxo	custo	fluxo	custo	fluxo	custo
AB	7	376	14.52	404	15.08	0	7	0	7	0	7
AC	30	297	35.94	74	31.48	300	36	600	42	0	30
AD	15	653	28.06	709	29.18	700	29	400	23	1000	35
BA	7	317	13.34	187	10.74	0	7	0	7	0	7
BD	11	708	25.16	917	29.34	700	25	700	25	700	25
BE	30	68	31.36	0	30.00	0	30	0	30	0	30
CA	30	3	30.06	0	30.00	0	30	0	30	0	30
CD	7	15	7.30	3	7.06	0	7	0	7	0	7
CF	20	288	25.76	106	22.12	180	24	600	32	0	20
CG	9	385	16.70	157	12.14	180	13	0	9	0	9
DA	15	7	15.14	0	15.00	0	15	0	15	0	15
DB	11	14	11.28	0	11.00	0	11	0	11	0	11
DC	7	387	14.74	192	10.84	60	8	0	7	0	7
DE	7	126	9.52	3	7.06	0	7	0	7	0	7
DG	7	581	18.62	666	20.32	510	17	1100	29	900	25
DH	9	315	15.30	768	24.36	830	26	0	9	800	25
EB	30	3	30.06	0	30.00	0	30	0	30	0	30
ED	7	39	7.78	0	7.00	0	7	0	7	0	7
EH	20	159	23.18	3	20.06	0	20	0	20	0	20
FC	20	3	20.06	0	20.00	0	20	0	20	0	20
FG	9	14	9.28	0	9.00	0	9	0	9	0	9
FI	13	627	25.54	630	25.60	540	24	900	31	900	31
GC	9	4	9.08	0	9.00	0	9	0	9	0	9
GD	7	9	7.18	0	7.00	0	7	0	7	0	7
GF	9	356	16.12	524	19.48	360	16	300	15	900	27
GH	9	113	11.26	41	9.82	0	9	0	9	0	9
GJ	40	0	40.00	0	40.00	0	40	0	40	0	40
GK	13	534	23.68	258	18.16	330	20	800	29	0	13
HD	9	6	9.12	0	9.00	0	9	0	9	0	9
HE	20	7	20.14	0	20.00	0	20	0	20	0	20
HG	9	35	9.70	0	9.00	0	9	0	9	0	9
HK	3	547	13.94	812	19.24	830	20	0	3	800	19
IF	13	0	13.00	0	13.00	0	13	0	13	0	13
IJ	9	13	9.26	1	9.02	0	9	0	9	0	9
IL	2	635	14.70	648	14.96	540	13	900	20	900	20
JG	40	0	40.00	0	40.00	0	40	0	40	0	40
JI	9	19	9.38	17	9.34	0	9	0	9	0	9
JK	9	0	9.00	0	9.00	0	9	0	9	0	9
JL	12	265	17.30	253	17.06	360	19	0	12	0	12
JM	12	22	12.44	1	12.02	0	12	0	12	0	12
KG	13	1	13.02	0	13.00	0	13	0	13	0	13
KH	3	8	3.16	0	3.00	0	3	0	3	0	3
KJ	9	291	14.82	270	14.40	360	16	0	9	0	9
KM	2	829	18.58	847	18.94	800	18	800	18	800	18
LI	2	2	2.04	2	2.04	0	2	0	2	0	2
LJ	12	0	12.00	0	12.00	0	12	0	12	0	12
MJ	12	2	12.04	0	12.00	0	12	0	12	0	12
MK	2	48	2.96	47	2.94	0	2	0	2	0	2
p.l.u.			89.60%		56.25%		33.33%		20.83%		18.75%

Tabela 5.4: Tempo de viagem e fluxo em cada link para os métodos DQ-learning, IQ-learning, *incremental assingment*, *successive averages* e *all-or-nothing*, na rede OW. O tempo de viagem sem fluxo é representado por c.s.f. e o percentual de links utilizados (com fluxo > 0) é representado por p.l.u.

5.2.1.3 Curva de Aprendizado dos Métodos Propostos

No intuito de compreender os efeitos do alinhamento da recompensa ao longo da simulação, neste experimento, são avaliadas as taxas de aprendizagem das abordagens propostas, por meio do tempo médio de viagem ao longo dos episódios. A Fig. 5.2 apresenta o tempo de convergência e desvio padrão para as 30 execuções apresentadas na Seção 5.2.1.1. Os resultados apresentam curvas muito semelhantes no gráfico principal, mas como mostra o gráfico interno, o DQ-learning converge para melhores soluções do que o IQ-learning. De forma mais específica, do episódio ≈ 75 em diante (período onde a taxa de exploração é muito próxima de 0%) é possível perceber que o IQ-learning para de convergir, enquanto que o DQ-learning continua melhorando a solução média dos agentes.

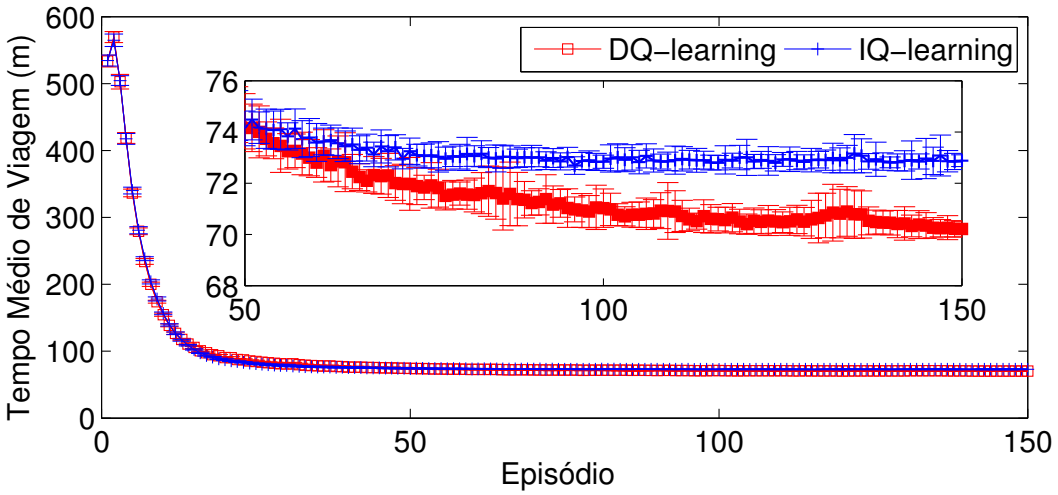


Figura 5.2: Tempo de convergência e desvio padrão ao longo dos episódios (principal) e ao final dos episódios (interno) das abordagens propostas na rede OW.

Para provar que os resultados apresentados por ambos os métodos são estatisticamente significativos, o seguinte teste para comprovar separabilidade dos dados foi realizado. Assumindo que os resultados encontrados pelo DQ-learning e IQ-learning a cada ponto são independentes, seus respectivos valores q_d e q_i, podem ser definidos por distribuições gaussianas. Deste modo, a diferença $d = \mid q_d - q_i \mid$ também é definida por uma distribuição gaussiana. A Fig. 5.3 apresenta o intervalo de confiança de d. Nos casos onde intervalo de confiança não contêm o valor zero, isto significa que os resultados são significativamente diferentes (considerando um intervalo de confiança de 95%) e, consequentemente, DQ-learning é melhor do que IQ-learning naqueles pontos (episódios). Como pode ser observado na Fig. 5.3, os resultados significantes atingidos pelo DQ-learning são mais predominantes ao final do período de exploração.

Uma questão muito importante que deve ser considerada neste experimento é que o processo de aprendizagem dos agentes é realizado de forma iterativa, no qual a rota é aprimorada a partir de sucessivas interações do agente com o ambiente. Assim, considerar

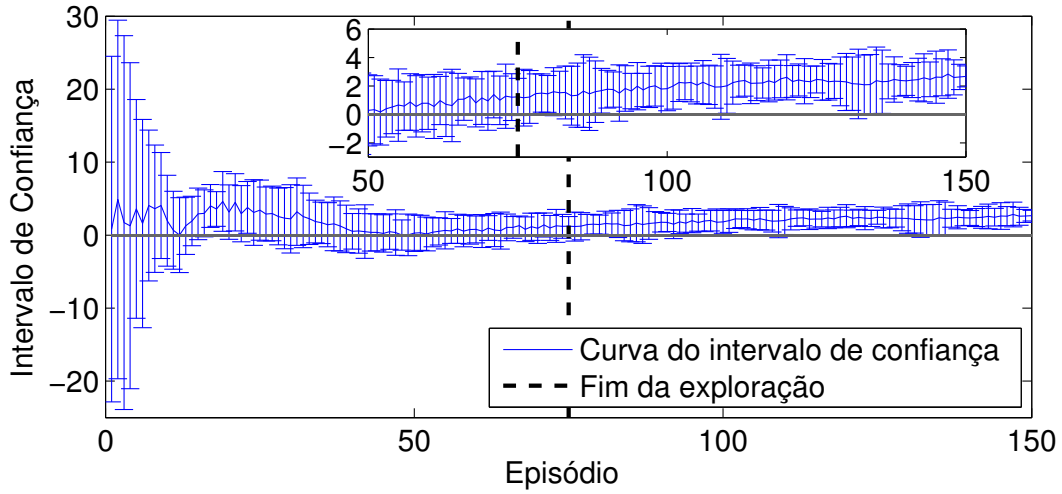


Figura 5.3: Intervalo de confiança da curva de diferença dos métodos ao longo dos episódios (principal) e nos episódios finais (interno) para a rede OW.

os episódios iniciais da simulação para avaliar a eficiência dos métodos pode não ser uma boa escolha. Então, para o intervalo de confiança apresentado na Fig. 5.3, avaliou-se em grupos de 5 episódios a frequência de tempo percentual em que o DQ-learning é estatisticamente melhor que o IQ-learning ao longo dos episódios, como ilustra a Fig. 5.4. Nos episódios iniciais, nada se pode concluir sobre o desempenho dos métodos. Isto é justificável pelo fato dos agentes aprenderem a rota com uma probabilidade de se comportar de maneira aleatória muito grande. No entanto, após o período de exploração é possível perceber que o DQ-learning supera o IQ-learning com mais frequência. Como pode ser observado, nos episódios finais a superioridade do DQ-learning se mantém. Isto significa que os agentes que utilizam o DQ-learning melhoram o tempo médio de viagem na rede, quando comparado ao tempo médio de viagem de agentes que utilizam o IQ-learning.

5.3 Cenário Sioux Falls: 360600 Agentes

A rede sioux falls é um conhecido problema de transporte apresentado na literatura para testar modelos de roteamento e alocação de tráfego. O conjunto de dados que compõem a rede, demanda e função de custo estão disponíveis em `http://www.bgu.ac.il/~bargera/tntp/`. Um repositório mantido pela universidade de Ben Gurion cuja finalidade é armazenar e disponibilizar problemas de transporte.

A matriz OD da rede sioux falls (ver Apêndice A) possui uma demanda de 360600 viagens distribuídas em 576 pares OD. A topologia da rede viária, composta por 24 nós e 76 links, é ilustrada na Fig 5.5. Os valores entre os nós e entre os links são seus respectivos identificadores. A função de custo utilizada nesta rede é a mesma da Eq. 2.7, na página 33. Apesar de existirem algumas variações para este problema, a demanda, parâmetros

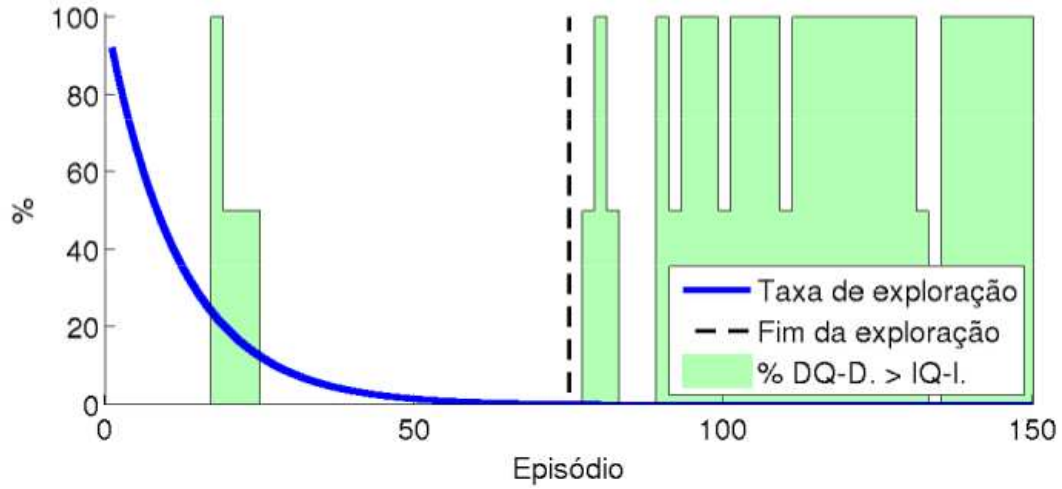


Figura 5.4: Frequência de tempo percentual para grupos de 5 episódios em que DQ-learning é estatisticamente melhor do que o IQ-learning na rede OW.

da rede e função de custo são utilizados de acordo com a versão padrão apresentada no repositório.

Embora este cenário possa ser considerado simples e de pequenas dimensões perante aos métodos de otimização, do ponto de vista de MARL, onde o processo de tomada de decisão é individual, esta afirmação não se sustenta. As principais motivações para o uso deste cenário na validação das abordagens propostas são: a grande quantidade de viagens e dimensões da rede. Comparado ao cenário OW, este possui uma demanda aproximadamente 200 vezes maior, além de superar em quantidade de estados (nós) e ações (links).

A próxima seção apresenta os resultados obtidos neste cenário.

5.3.1 Resultados Experimentais

Esta seção apresenta os resultados experimentais realizados para avaliar os métodos propostos nesta rede. Duas métricas são utilizadas para avaliar a qualidade das soluções obtidas. A primeira (Seção 5.3.1.1), assim como no cenário anterior, avalia o tempo médio de viagem na rede e o tempo médio de viagem por par OD em minutos. A segunda (Seção 5.3.1.2) avalia a curva de aprendizagem dos métodos propostos e compara seus resultados.

5.3.1.1 Tempo Médio de Viagem na Rede

Neste experimento são avaliados os tempos médios de viagem por par OD e o tempo médio de viagem na rede para ambos os métodos propostos e métodos de TA utilizados na comparação. Em função de haverem 576 pares na matriz OD do problema, apenas um conjunto destes será discutido individualmente. Os pares OD apresentados na Tab.

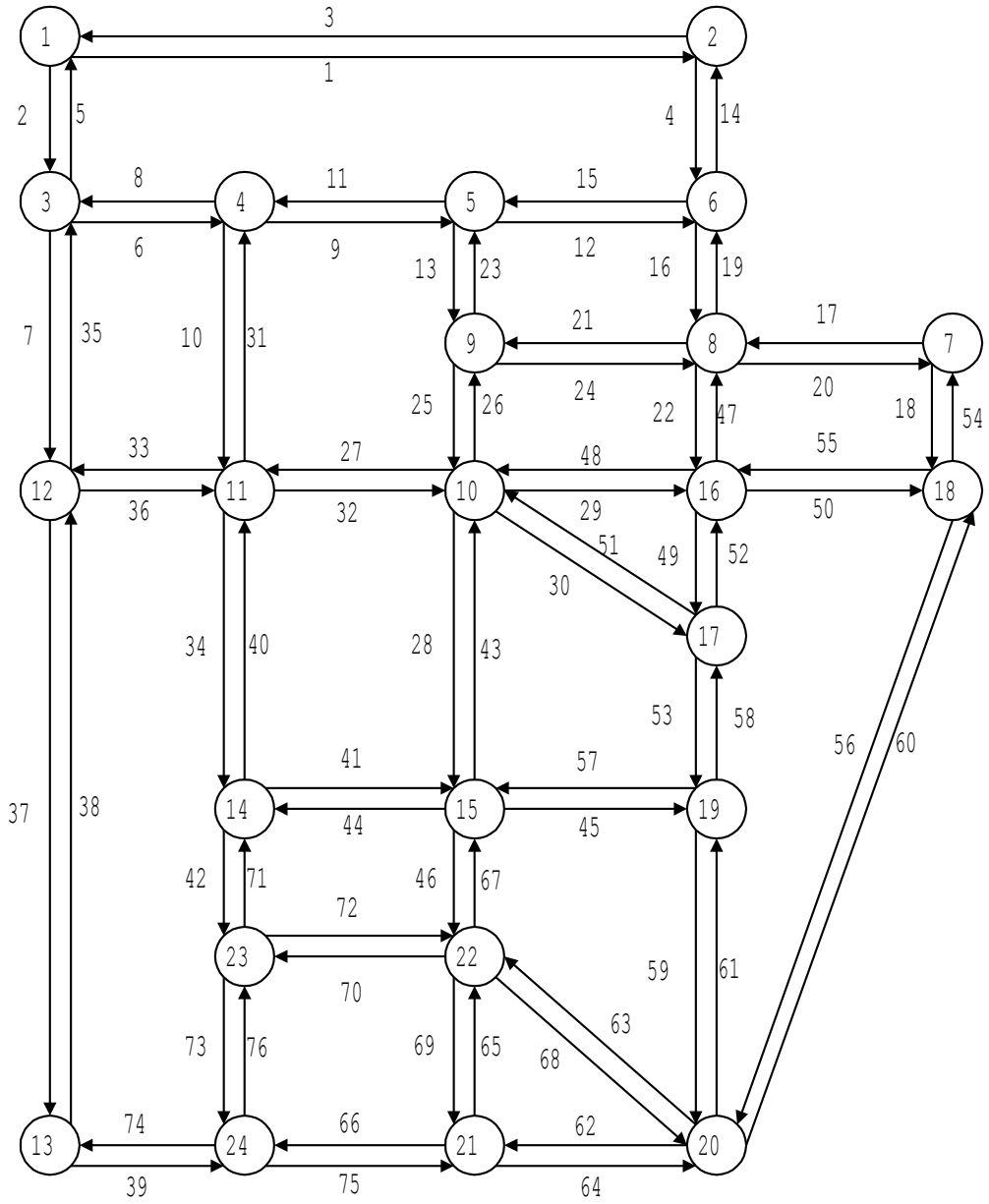


Figura 5.5: Topologia da rede sioux falls.

5.5 foram selecionados de modo que as mais diversas características da matriz OD como distância topológica e quantidade de viagens fossem cobertas.

par OD	viagens	IQ-learning	DQ-learning	inc. ass.	suc. avg.	all-or-noth.
1-20	3000	23.09 (0.1781)	24.03 (0.1678)	22.27	22.27	22.76
2-11	2000	17.48 (0.0001)	17.38 (0.0412)	18.08	20.00	17.48
9-16	4000	8.51 (0.4095)	8.25 (1.5240)	12.86	7.39	11.52
10-16	44000	7.21 (0.1232)	8.86 (3.0388)	10.54	12.80	122.50
13-2	3000	17.01 (0.0000)	17.01 (0.0008)	17.00	17.00	17.00
15-13	7000	12.24 (0.0003)	12.34 (0.0915)	13.78	12.03	12.06
18-13	1000	17.07 (0.0002)	17.20 (0.2129)	19.00	17.00	17.00
20-24	4000	9.48 (0.0008)	9.60 (0.6691)	9.66	9.61	9.61
22-13	13000	10.07 (0.0011)	11.36 (1.6582)	9.86	10.15	10.15
TMV	360600	9.88 (0.0474)	10.38 (0.2689)	10.43	10.52	17.21

Tabela 5.5: Tempo médio de viagem na rede (TMV) e tempo médio por par OD em minutos na rede sioux falls. Valores entre parênteses representam o desvio padrão.

Como pode ser observado na Tab. 5.5, as abordagens baseadas em agentes obtiveram os menores tempos. A abordagem IQ-learning além de obter o menor tempo de viagem na rede também apresenta soluções muito mais estáveis quando comparadas ao DQ-learning. Pode-se perceber que o desvio padrão do IQ-learning é aproximadamente 5 vezes menor do que DQ-learning. Isto é um indicador de que apesar da presença de 360600 agentes aprendendo simultaneamente, os agentes aprendendo com o IQ-learning convergem para soluções muito semelhantes ao longo das repetidas execuções, enquanto os agentes que aprendem com o DQ-learning apresentam uma variação maior nas rotas encontradas. Esta diferença no tempo médio de viagem na rede será melhor discutida, por meio do percentual de ocupação dos links, na seção 5.3.1.2.

Analisando os tempos médios de viagem por par OD pode-se perceber que as abordagens baseadas em agentes apresentaram os melhores resultados. Novamente, os desvios padrões do IQ-learning são menores do que obtidos pelo DQ-learning. Pode-se perceber também que no par 10-16 cujo número de viagens é o maior da matriz OD, além de o IQ-learning apresentar tempo de viagem inferior, o desvio padrão também é muito menor. Para este mesmo par OD, pode-se perceber que os demais métodos apresentaram tempos superiores, principalmente no caso do *all-or-nothing* que concentra as 44000 viagens na mesma rota.

Embora as abordagens que apresentaram melhores resultados sejam as baseadas em agentes, tanto o *successive averages* quanto o *incremental assignment* apresentaram resultados muito próximos ao DQ-learning. Um dos motivos para isto é o fato de os links da rede sioux falls possuírem uma capacidade muito grande. Desta forma, embora o número de viagens na rede seja grande a infraestrutura da rede é robusta o bastante para comportar esta demanda. Assim, a partir de algumas iterações, métodos menos inteligentes como os mencionados anteriormente atingem resultados razoáveis.

5.3.1.2 Curva de Aprendizado dos Métodos Propostos

Neste experimento as curvas de aprendizagem dos métodos IQ-learning e DQ-learning para a rede sioux falls são apresentadas e discutidas. Assim como na Seção 5.2.1.3, as curvas neste experimento avaliam a convergência do tempo de viagem na rede dos métodos IQ-learning e DQ-learning. A Fig. 5.6 apresenta esta convergência ao longo dos episódios da simulação. Cada curva representa à média e desvio padrão de 30 execuções. Como pode ser observado no gráfico principal, as curvas apresentam comportamento bastante semelhante. Porém, ao avaliar o gráfico interno, pode-se perceber que os métodos convergem para soluções distintas após o episódio 70. Lembrando que a partir do episódio 75 a taxa de exploração é muito próxima a 0%.

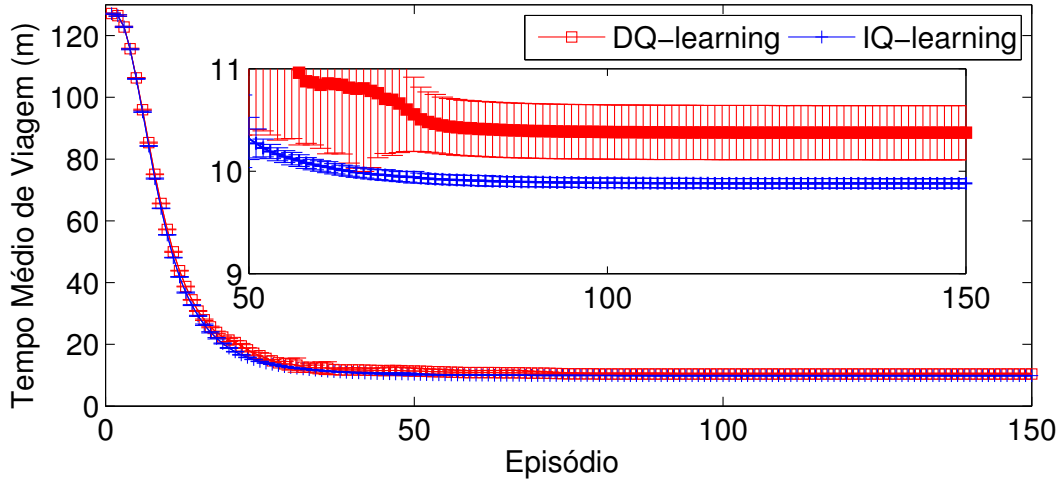


Figura 5.6: Tempo de convergência e desvio padrão ao longo dos episódios (principal) e ao final dos episódios (interno) das abordagens propostas na rede sioux falls.

Apesar de o IQ-learning conseguir uma redução de aproximadamente 0.5 minutos quando comparado ao DQ-learning e das barras de erro não se sobreporem, ainda não se pode concluir que o alinhamento local apresentou melhores resultados neste experimento. Para tanto, o mesmo teste realizado na Seção 5.2.1.3 é realizado neste experimento. Logo, assumindo que os resultados encontrados pelo DQ-learning e IQ-learning a cada ponto são independentes, seus respectivos valores q_d e q_i, podem ser definidos por distribuições gaussianas. Deste modo, a diferença $d = \mid q_i - q_d \mid$ também é definida por uma distribuição gaussiana. A Fig. 5.7 apresenta o intervalo de confiança de d. Nos casos onde o intervalo de confiança não contém o valor zero, para um intervalo de confiança de 95%, pode-se concluir que o IQ-learning apresenta resultados superiores ao DQ-learning, no cenário Sioux Falls.

Analisando a Fig. 5.7 pode-se observar que nos episódios iniciais as soluções encontradas pelos métodos são distintas, e, consequentemente, IQ-learning é melhor. Entretanto, a partir do fim do período de exploração, observando o gráfico interno da Fig. 5.7 pode-se

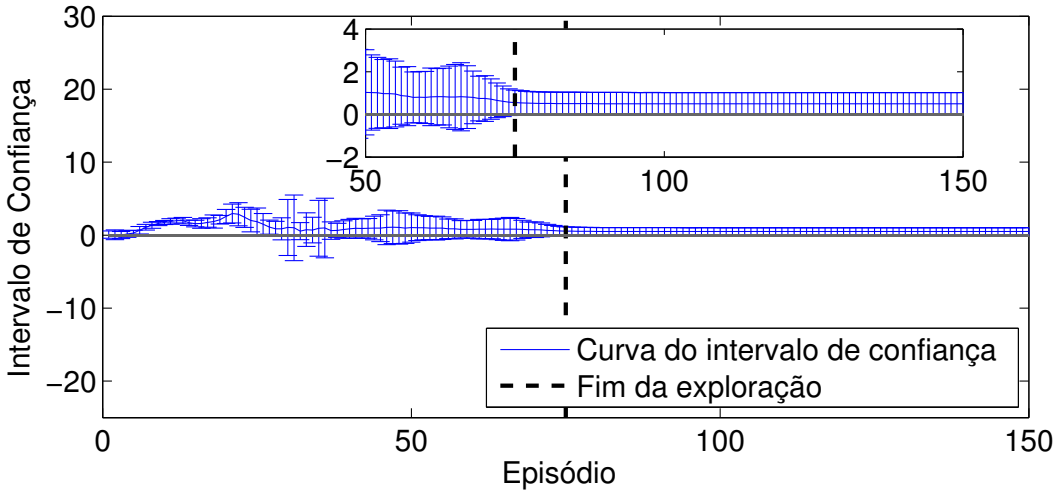


Figura 5.7: Intervalo de confiança da curva de diferença dos métodos ao longo dos episódios (principal) e nos episódios finais (interno) para a rede OW.

perceber que as barras de erro sobrepõem o ponto zero. Para facilitar a visualização desta sobreposição, a Fig. 5.8 apresenta a frequência de tempo percentual para grupos de 5 episódios em que o IQ-learning é melhor. Como pode ser observado, apenas nos episódios iniciais o IQ-learning apresenta resultados melhores. Contudo, devido ao fato deste intervalo compreender o período de exploração, não se pode assumir que o IQ-learning é melhor que o DQ-learning. Logo, as soluções apresentadas por ambos os métodos são equivalentes.

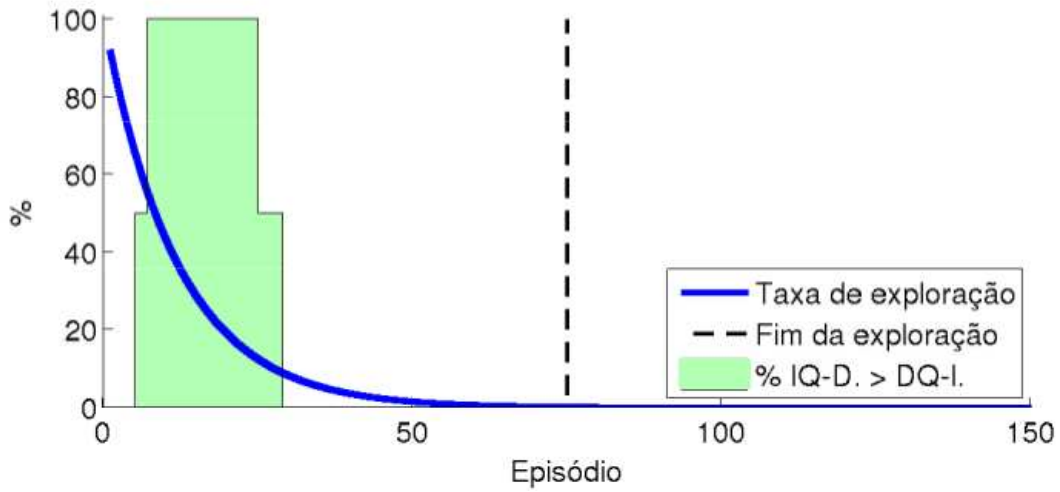


Figura 5.8: Frequência de tempo percentual para grupos de 5 episódios em que IQ-learning é estatisticamente melhor que o DQ-learning na rede sioux falls.

5.4 Resumo dos Experimentos

Esta seção sumariza a análise dos experimentos realizados em ambos os cenários para avaliar os métodos propostos. Na Seção 5.2, as abordagens DQ-learning e IQ-learning foram avaliadas sob três perspectivas diferentes em um cenário com 1700 agentes, 13 nós e 24 links. Inicialmente, avaliaram-se os tempos médios de viagem obtidos com os métodos. Neste experimento, foi possível observar que o alinhamento da função de recompensa à utilidade do sistema consegue proporcionar uma redução considerável no tempo médio de viagem na rede. Além disso, na média, os agentes de cada par OD se beneficiaram do uso da função de recompensa *difference rewards* quando comparados aos resultados obtidos com a função individual. Para explicar melhor este ganho de desempenho, para este mesmo cenário, foram avaliados os percentuais de links utilizados durante a simulação para ambos os métodos propostos. Assumindo que no cenário OW todas as viagens se deslocam de um mesmo sentido para outro na rede (de A ou B para L ou M), intuitivamente pode-se presumir que alguns links não são utilizados pelos agentes devido ao fato de comporem caminhos inversos aos pares OD presentes no problema. No entanto, resultados demonstram que o DQ-learning utiliza aproximadamente 33% mais links do que o IQ-learning. Desta forma, o ganho de desempenho obtido pelo alinhamento da função de recompensa à utilidade global do sistema, para a rede OW, se explica por meio de uma melhor distribuição dos veículos sobre a rede. Estes experimentos foram comparados com três métodos de alocação de tráfego existentes na literatura: *incremental assignment*, *successive averages* e *all-or-nothing*. As abordagens propostas neste trabalho apresentaram resultados superiores em todos os casos avaliados quando comparados aos resultados obtidos através dos métodos de alocação de tráfego. Ainda no cenário OW, um último experimento foi realizado para comprovar a significância das soluções encontradas pelo IQ-learning e DQ-learning. Resultados mostraram que as soluções adquiridas através do DQ-learning são estatisticamente superiores às obtidas com o IQ-learning. Porém, isto somente ocorre nos episódios finais. Durante os iniciais, onde a taxa de exploração é muito alta, as soluções encontradas por ambos os métodos são consideradas equivalentes.

O segundo cenário utilizado para avaliar as abordagens propostas é o sioux falls. Esta rede possui 360600 viagens distribuídas em 576 pares OD, 24 nós e 76 links. Comparado ao cenário anterior, do ponto de vista da tarefa de aprendizagem por reforço, este cenário é muito mais complexo. Tanto a grande quantidade de agentes quanto o crescimento dos estados e ações do MDP tornam a tarefa de aprendizagem de rotas mais difícil. Para este cenário, na Seção 5.3.1.1 foram avaliados os tempos médios de viagem na rede e os tempos médios de viagem por par OD. Para este cenário, entre as abordagens propostas, a que apresentou os melhores resultados médios foi o IQ-learning. Em um segundo experimento, as curvas de aprendizado dos métodos ao longo dos episódios para ambos os métodos propostos foram avaliadas. Resultados experimentais demonstraram que as cur-

vas apresentam comportamento muito semelhante. Testes estatísticos evidenciam que os resultados obtidos com DQ-learning e IQ-learning são considerados equivalentes. Os tempos médios de viagem na rede e por par OD foram comparados com os mesmos métodos de TA utilizados no cenário OW e em todos os casos os métodos propostos apresentaram resultados superiores. Para este cenário não foi avaliado o percentual de links utilizados na simulação porque a matriz OD possui uma grande quantidade de pares OD. Isto implica em uma ocupação de 100% dos links, independentemente do método utilizado.

O alinhamento da função de recompensa à utilidade global do sistema demonstrou-se mais eficiente para o cenário OW. Já para o cenário com muitos agentes aprendendo, embora os resultados dos métodos serem estatisticamente equivalentes, o alinhamento da função de recompensa à utilidade do agente apresentou soluções mais estáveis. Isto pode ser explicado pelo fato de que a função *difference rewards* estima o sinal de recompensa do agente com base em sua contribuição para o desempenho global do sistema, mas como na rede sioux falls existe uma quantidade muito grande de agentes aprendendo simultaneamente, a contribuição deste agente não é tão expressiva quanto em um cenário com apenas 1700 agentes. Deste modo, é possível concluir que, o alinhamento da função de recompensa à utilidade do sistema é mais eficiente na presença de poucos agentes, mas equivalente ao alinhamento à utilidade do agente quando aplicado em cenários com um grande número de agentes aprendendo.

6 CONSIDERAÇÕES FINAIS

Este capítulo apresenta uma visão geral das questões discutidas ao longo dos capítulos desta dissertação, bem como as suas conclusões, contribuições e perspectivas de continuação.

Nesta dissertação, utilizamos aprendizado por reforço para resolver o problema de alocação de tráfego em redes viárias. Tratamos este problema como um problema de aprendizado de rotas, onde as viagens da matriz OD são os agentes e a rede viária é o ambiente dos agentes. Utilizamos duas funções de recompensa que diferem no alinhamento da recompensa enviado do ambiente ao agente para modelar o processo de aprendizagem de rotas dos agentes. A primeira função é chamada função individual e tem seu sinal numérico alinhado à utilidade individual do agente. Isto significa que o agente que utiliza esta função age no sistema de forma egoísta, ou seja, toma ações que maximizam a sua utilidade individual (reduzem o seu custo de deslocamento) independentemente desta ação ser prejudicial ao sistema ou não. A segunda função de recompensa é a apresentada em (TUMER; AGOGINO, 2007) e conhecida como *difference rewards*. Esta função tem como característica o alinhamento do sinal numérico à utilidade do sistema. Desta forma, o agente que aprende por reforço com esta função irá tomar ações que maximizam o desempenho do sistema ao invés de sua utilidade individual como ocorre na primeira função.

De posse destas duas funções, modelamos o processo de aprendizado por reforço dos agentes através da técnica de múltiplos aprendizes independentes com o algoritmo Q-learning. A abordagem de aprendizagem de rotas utilizada constrói a rota do agente durante a sua viagem na rede. Este método de aprendizagem de rotas torna a tarefa ainda mais complexa, já que a quantidade de estados e ações é dependente da topologia da rede viária.

A partir do modelo básico de aprendizado por reforço, avaliamos os resultados obtidos pelos agentes para cada uma destas funções em dois problemas de alocação de tráfego que diferem em quantidade de agentes aprendendo e topologia da rede viária. No cenário com 1700 agentes apresentado na Seção 5.2, observamos que o alinhamento da função de recompensa à utilidade do sistema consegue reduzir o custo de deslocamento dos agentes na rede, devido a uma melhor distribuição dos veículos sobre a infraestrutura disponível. Neste mesmo cenário, comparamos as duas abordagens baseadas em agentes propostas nesta dissertação com os métodos de alocação de tráfego *incremental assignment*, *successive averages* e *all-or-nothing*. Em todos os casos as abordagens propostas apresentam resultados superiores. Porém, em todos os casos avaliados o alinhamento da recompensa à utilidade do sistema apresentou resultados superiores.

No cenário sioux falls apresentado na Seção 5.3, aplicamos os métodos propostos com 360600 agentes aprendendo simultaneamente. Através dos resultados experimentais foi

possível observar que ambas as abordagens propostas apresentam soluções estatisticamente equivalentes. Porém, comparados aos métodos de alocação de tráfego *incremental assignment*, *successive averages* e *all-or-nothing* os resultados obtidos com as abordagens baseadas em agentes são superiores.

Através dos experimentos foi possível perceber que a função *difference rewards* é sensível à quantidade de agentes aprendendo. Devido ao fato da função utilizar a contribuição do agente no desempenho do sistema para a estimação do reforço, em problemas com menor quantidade de agentes aprendendo, a contribuição de cada indivíduo é muito mais expressiva do que nos casos em que há um número muito grande de agentes (como 360600). Embora em ambientes com estas condições as soluções encontradas por ambos os métodos propostos sejam equivalentes, a aplicação de *difference rewards* apresenta algumas desvantagens em relação à função individual, das quais podemos destacar a necessidade de completa observação do ambiente por cada agente para que o reforço seja estimado, enquanto a função individual utiliza apenas observação local. Apesar disso, a recompensa do *difference rewards* estimula uma cooperação implícita entre os agentes capaz de beneficiar o sistema como um todo no cenário com 1700 agentes.

6.1 Contribuições

Apesar de esta dissertação utilizar um conjunto de ferramentas já consolidadas na literatura para resolver o problema aprendizado de rotas, ela apresenta algumas contribuições importantes para o estado da arte das técnicas de aprendizado por reforço. Em primeiro lugar, podemos ressaltar que este trabalho é o primeiro a utilizar tantos agentes aprendendo simultaneamente em um mesmo ambiente. Como apresentado na Seção 3.1, dentre às abordagens apresentadas para o problema de escolha de rotas, a que apresenta a maior quantidade de agentes utiliza 1700. Sendo assim, esta dissertação prova que é possível aplicar a MARL em problemas com centenas de milhares de agentes aprendendo simultaneamente.

Outra contribuição que merece destaque é a aplicação da função *difference rewards* em problemas de aprendizagem de grande escala. Foi possível identificar que os benefícios proporcionados pelo alinhamento da função de recompensa à utilidade do sistema são sensíveis ao número de agentes aprendendo. Embora esta função apresentasse resultados superiores à função individual no cenário com 1700 agentes, no cenário com 360600 agentes os resultados obtidos foram equivalentes em ambos os casos.

6.2 Perspectivas de Continuação

Como apresentado anteriormente, a correta escolha/modelagem da função de recompensa é um fator chave para se adquirir bons resultados em RL. Este é um tema recor-

rente na literatura e pode trazer inúmeros benefícios para os métodos de RL, dos quais podemos destacar a redução do tempo de aprendizagem. Neste contexto, abordagens de *reward shaping* (NG; HARADA; RUSSELL, 1999), como *potential-based reward shaping*(WIEWIORA; COTTRELL; ELKAN, 2003; DEVLIN; KUDENKO, 2012) são propostas como alternativas para inserir um sinal de recompensa extra no processo de RL. Neste sentido, uma possível extensão deste trabalho, seria utilizar tanto o alinhamento da função de recompensa local quando o global em um mesmo processo de aprendizagem. Deste modo, pode ser que uma correta combinação destas técnicas beneficie o sistema como um todo. Um trabalho neste sentido é apresentado (DEVLIN et al., 2014), onde os autores utilizam uma combinação de *reward shaping* e *difference rewards* para de MARL com até 100 agentes.

Outra possibilidade de extensão deste trabalho seria o desenvolvimento de técnicas de alinhamento da função de recompensa à utilidade do sistema que não sejam sensíveis à quantidade de agentes aprendendo. Além disso, outro fator importante a ser melhorado seria a redução da observabilidade do sistema necessária para a estimação da recompensa na função *difference rewards*. O fato de necessitar de completa observação do sistema pode inviabilizar a aplicação de *difference rewards* em domínios onde esta característica não é disponível como cenários de desastre.

REFERÊNCIAS

AGOGINO, A. K.; TUMER, K. A multiagent approach to managing air traffic flow. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v.24, n.1, p.1–25, 2012.

ARFAOUI, A. **Estimation des matrices origine-destination a partir des comptages**: etude de quelques modeles. [S.l.]: Laboratoire de Statistiques Théoriques et Appliquées, Université Pierre et Marie Curie, 1999.

BAZZAN, A. L. C. Learning-based traffic assignment: how heterogeneity in route choices pays off. In: INTERNATIONAL WORKSHOP ON AGENTS IN TRAFFIC AND TRANSPORTATION (ATT-2014), 8. **Proceedings...** [S.l.: s.n.], 2014. held at the 13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2014).

BAZZAN, A. L. C. et al. Wayward Agents in a Commuting Scenario (Personalities in the Minority Game). In: INT. CONF. ON MULTI-AGENT SYSTEMS (ICMAS). **Proceedings...** IEEE Computer Science, 2000. p.55–62.

BAZZAN, A. L. C.; KLÜGL, F. Sistemas Inteligentes de Transporte e Tráfego: uma abordagem de tecnologia da informação. In: KOWALTOWSKI, T.; BREITMAN, K. K. (Ed.). **Anais das Jornadas de Atualização em Informática**. [S.l.]: SBC, 2007.

BAZZAN, A. L. C.; KLÜGL, F. Re-Routing Agents in an Abstract Traffic Scenario. In: ADVANCES IN ARTIFICIAL INTELLIGENCE, Berlin. **Anais...** Springer-Verlag, 2008. n.5249, p.63–72. (Lecture Notes in Artificial Intelligence).

BAZZAN, A. L. C.; KLÜGL, F. **Introduction to Intelligent Systems in Traffic and Transportation**. [S.l.]: Morgan and Claypool, 2013. 1-137p. n.3. (Synthesis Lectures on Artificial Intelligence and Machine Learning, v.7).

BELLMAN, R. E. **Dynamic Programming**. Princeton: Princeton University Press, 1957.

BEN-ELIA, E.; SHIFTAN, Y. Which road do I take? A learning-based model of route-choice behavior with real-time information. **Transportation Research Part A: Policy and Practice**, [S.l.], v.44, n.4, p.249–264, 2010.

BERNSTEIN, D. S. et al. The Complexity of Decentralized Control of Markov Decision Processes. **Mathematics of Operations Research**, Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA, v.27, n.4, p.819–840, 2002.

BURIOL, L. S. et al. A biased random-key genetic algorithm for road congestion minimization. **Optimization Letters**, [S.l.], v.4, p.619–633, 2010.

BUŞONIU, L.; BABUSKA, R.; DE SCHUTTER, B. A comprehensive survey of multiagent reinforcement learning. **Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on**, [S.l.], v.38, n.2, p.156–172, 2008.

CAGARA, D.; BAZZAN, A. L. C.; SCHEUERMANN, B. Getting You Faster to Work: a genetic algorithm approach to the traffic assignment problem. In: ANNUAL CONFERENCE ON GENETIC AND EVOLUTIONARY COMPUTATION (COMPANION), 16., New York, NY, USA. **Proceedings...** ACM, 2014. p.105–106. (GECCO '14).

CAGARA, D.; SCHEUERMANN, B.; BAZZAN, A. L. C. A Methodology to Evaluate the Optimization Potential of Co-ordinated Vehicular Route Choices. In: GI/ITG KuVS FACHGESPRÄCH INTER-VEHICLE COMMUNICATION (FG-IVC 2013), 1., Innsbruck. **Anais...** [S.l.: s.n.], 2013.

CHMURA, T.; PITZ, T. An Extended Reinforcement Algorithm for Estimation of Human Behavior in Congestion Games. **Journal of Artificial Societies and Social Simulation**, [S.l.], v.10, n.2, 2007.

CLAUS, C.; BOUTILIER, C. The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems. In: FIFTEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Proceedings...** [S.l.: s.n.], 1998. p.746–752.

DEVLIN, S. et al. Potential-based difference rewards for multiagent reinforcement learning. In: AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, 2014. **Proceedings...** [S.l.: s.n.], 2014. p.165–172.

DEVLIN, S.; KUDENKO, D. Dynamic Potential-based Reward Shaping. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 11. **Proceedings...** International Foundation for Autonomous Agents and Multiagent Systems, 2012. p.433–440.

DIJKSTRA, E. W. A note on two problems in connection with graphs. **Numerische Mathematik**, [S.l.], v.1, p.269–271, 1959.

FUDENBERG, D.; TIROLE, J. **Game Theory**. [S.l.]: The MIT Press, 1991. (MIT Press Books).

GALIB, S. M.; MOSER, I. Road traffic optimisation using an evolutionary game. In: GENETIC AND EVOLUTIONARY COMPUTATION, 13., New York, NY, USA. **Proceedings...** ACM, 2011. p.519–526. (GECCO '11).

GUESTRIN, C.; LAGOUDAKIS, M. G.; PARR, R. Coordinated Reinforcement Learning. In: NINETEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), San Francisco, CA, USA. **Proceedings...** Morgan Kaufmann, 2002. p.227–234.

HOLMESPARKER, C. et al. CLEANing the reward: counterfactual actions to remove exploratory action noise in multiagent learning. In: AUTONOMOUS AGENTS AND MULTI-AGENT SYSTEMS, 2014. **Proceedings...** [S.l.: s.n.], 2014. p.1353–1354.

HU, J.; WELLMAN, M. P. Nash Q-learning for General-sum Stochastic Games. **J. Mach. Learn. Res.**, [S.l.], v.4, p.1039–1069, 2003.

KAELBLING, L. P.; LITTMAN, M. L.; CASSANDRA, A. R. Planning and Acting in Partially Observable Stochastic Domains. **Artificial Intelligence**, [S.l.], v.101, n.1–2, p.99–134, 1998.

KLÜGL, F.; BAZZAN, A. L. C. Route Decision Behaviour in a Commuting Scenario. **Journal of Artificial Societies and Social Simulation**, [S.l.], v.7, n.1, 2004.

KOK, J. R.; VLASSIS, N. Sparse cooperative Q-learning. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING (ICML), 21., New York, USA. **Proceedings...** ACM Press, 2004. p.481–488.

KOK, J.; VLASSIS, N. Collaborative Multiagent Reinforcement Learning by Payoff Propagation. **Journal of Machine Learning Research**, [S.l.], v.7, p.1789–1828, 2006.

KRAUSS, S. **Microscopic Modelling of Traffic Flow**: investigation of collision free vehicle dynamics. 1998. Tese (Doutorado em Ciência da Computação) — DLR (Hauptabteilung Mobilität und Systemtechnik).

LAUER, M.; RIEDMILLER, M. An Algorithm for Distributed Reinforcement Learning in Cooperative Multi-Agent Systems. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 17. **Proceedings...** Morgan Kaufmann: San Francisco: CA, 2000. p.535–542.

LITTMAN, M. L. Markov Games as a Framework for Multi-Agent Reinforcement Learning. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, ML, 11., New Brunswick, NJ. **Proceedings...** Morgan Kaufmann, 1994. p.157–163.

LITTMAN, M. L.; DEAN, T. L.; KAELBLING, L. P. On the complexity of solving Markov decision problems. In: ANNUAL CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE, UAI, 11., Montreal, Québec, Canada. **Proceedings...** [S.l.: s.n.], 1995. p.394–402.

MITCHELL, T. M. **Machine Learning**. 1.ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

NARENDRA, K. S.; THATHACHAR, M. A. L. **Learning Automata**: an introduction. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.

NASH, J. F. Non-Cooperative Games. **Annals of Mathematics**, [S.l.], v.2, n.54, p.286–295, 1951.

NG, A. Y.; HARADA, D.; RUSSELL, S. Policy invariance under reward transformations: theory and application to reward shaping. In: IN PROCEEDINGS OF THE SIXTEENTH INTERNATIONAL CONFERENCE ON MACHINE LEARNING. **Anais...** Morgan Kaufmann, 1999. p.278–287.

OLIVEIRA, D. de.; BAZZAN, A. L. C. Multiagent Learning on Traffic Lights Control: effects of using shared information. In: BAZZAN, A. L. C.; KLÜGL, F. (Ed.). **Multi-Agent Systems for Traffic and Transportation**. Hershey, PA: IGI Global, 2009. p.307–321.

ORTÚZAR, J.; WILLUMSEN, L. G. **Modelling Transport**. 3rd.ed. [S.l.]: John Wiley & Sons, 2001.

PUTERMAN, M. L. **Markov Decision Processes**: discrete stochastic dynamic programming. New York, NY, USA: Wiley–Interscience, 2005. (Wiley Series in Probability and Statistics).

RAMOS, G. de. O.; GRUNITZKI, R.; BAZZAN, A. L. C. On improving route choice through learning automata. In: FIFTH INTERNATIONAL WORKSHOP ON COLLABORATIVE AGENTS – RESEARCH & DEVELOPMENT (CARE 2014). **Proceedings...** [S.l.: s.n.], 2014. p.1–12.

SHIGEHIRO, Y.; MIYAKAWA, T.; MASUDA, T. Road traffic control based on genetic algorithm for reducing traffic congestion. **Electronics and Communications in Japan**, [S.l.], v.95, n.4, p.11–19, 2012.

SUTTON, R.; BARTO, A. **Reinforcement Learning**: an introduction. Cambridge, MA: MIT Press, 1998.

SUTTON, R. S. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 7. **Proceedings...** [S.l.: s.n.], 1990. p.216–224.

TAVARES, A. R. **Uma abordagem baseada em agentes para simulação de tarifação viária e comunicação inter-veicular**. 2013. Dissertação (Mestrado em Ciência

da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil.

TAVARES, A. R.; BAZZAN, A. L. C. Independent learners in abstract traffic scenarios. **Revista de Informática Teórica e Aplicada**, [S.l.], v.19, n.2, p.13–33, 2012.

TONG, C.; WONG, S. A predictive dynamic traffic assignment model in congested capacity-constrained road networks. **Transportation Research Part B: Methodological**, [S.l.], v.34, n.8, p.625 – 644, 2000.

TUMER, K.; AGOGINO, A. Agent reward shaping for alleviating traffic congestion. In: WORKSHOP ON AGENTS IN TRAFFIC AND TRANSPORTATION, Hakodate, Japan. **Anais...** [S.l.: s.n.], 2006.

TUMER, K.; AGOGINO, A. Distributed agent-based air traffic flow management. In: AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS, 6., New York, NY, USA. **Proceedings...** ACM, 2007. p.1–8.

TUMER, K.; KHANI, N. Learning from actions not taken in multiagent systems. **Advances in Complex Systems**, [S.l.], v.12, n.04n05, p.455–473, 2009.

TUMER, K.; WOLPERT, D. Collective Intelligence and Braess' Paradox. In: SEVENTEENTH NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Proceedings...** AAAI Press, 2000. p.104–109.

TUMER, K.; WOLPERT, D. A Survey of Collectives. In: TUMER, K.; WOLPERT, D. (Ed.). **Collectives and the Design of Complex Systems**. [S.l.]: Springer, 2004. p.1–42.

TUYLS, K.; WEISS, G. Multiagent Learning: basics, challenges, and prospects. **AI Magazine**, [S.l.], v.33, n.3, p.41–52, 2012.

WANG, X.; SANDHOLM, T. Reinforcement learning to play an optimal nash equilibrium in team markov games. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 15 (NIPS-2002). **Anais...** [S.l.: s.n.], 2002.

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. **Machine Learning**, Hingham, MA, USA, v.8, n.3, p.279–292, 1992.

WIEWIORA, E.; COTTRELL, G.; ELKAN, C. Principled methods for advising reinforcement learning agents. In: TWENTIETH INTERNATIONAL CONFERENCE ON MACHINE LEARNING. **Proceedings...** [S.l.: s.n.], 2003. p.792–799.

WOOLDRIDGE, M. J. **An Introduction to MultiAgent Systems**. Chichester: John Wiley & Sons, 2009. 461p. Second edition.

APÊNDICE A - MATRIZ OD DA REDE SIOUX FALLS

Os experimentos apresentados na Seção 5.3 são realizados na rede sioux falls. Esta rede apresenta 24 zonas de origem e 24 zonas de destino que determinam a matriz OD. A proporção de viagens da matriz OD em milhar é apresentada na Tab. 6.1.

OD	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
1	0	1	1	5	2	3	5	8	5	13	5	2	5	3	5	5	4	1	3	3	1	4	3	1
2	1	0	1	2	1	4	2	4	2	6	2	1	3	1	1	4	2	0	1	1	0	1	0	0
3	1	1	0	2	1	3	1	2	1	3	3	2	1	1	1	2	1	0	0	0	0	1	1	0
4	5	2	2	0	5	4	4	7	7	12	14	6	6	5	5	8	5	1	2	3	2	4	5	2
5	2	1	1	5	0	2	2	5	8	10	5	2	2	1	2	5	2	0	1	1	1	2	1	0
6	3	4	3	4	2	0	4	8	4	8	4	2	2	1	2	9	5	1	2	3	1	2	1	1
7	5	2	1	4	2	4	0	10	6	19	5	7	4	2	5	14	10	2	4	5	2	5	2	1
8	8	4	2	7	5	8	10	0	8	16	8	6	6	4	6	22	14	3	7	9	4	5	3	2
9	5	2	1	7	8	4	6	8	0	28	14	6	6	6	9	14	9	2	4	6	3	7	5	2
10	13	6	3	12	10	8	19	16	28	0	40	20	19	21	40	44	39	7	18	25	12	26	18	8
11	5	2	3	15	5	4	5	8	14	39	0	14	10	16	14	14	10	1	4	6	4	11	13	6
12	2	1	2	6	2	2	7	6	6	20	14	0	13	7	7	7	6	2	3	4	3	7	7	5
13	5	3	1	6	2	2	4	6	6	19	10	13	0	6	7	6	5	1	3	6	6	13	8	8
14	3	1	1	5	1	1	2	4	6	21	16	7	6	0	13	7	7	1	3	5	4	12	11	4
15	5	1	1	5	2	2	5	6	10	40	14	7	7	13	0	12	15	2	8	11	8	26	10	4
16	5	4	2	8	5	9	14	22	14	44	14	7	6	7	12	0	28	5	13	16	6	12	5	3
17	4	2	1	5	2	5	10	14	9	39	10	6	5	7	15	28	0	6	17	17	6	17	6	3
18	1	0	0	1	0	1	2	3	2	7	2	2	1	1	2	5	6	0	3	4	1	3	1	0
19	3	1	0	2	1	2	4	7	4	18	4	3	3	3	8	13	17	3	0	12	4	12	3	1
20	3	1	0	3	1	3	5	9	6	25	6	5	6	5	11	16	17	4	12	0	12	24	7	4
21	1	0	0	2	1	1	2	4	3	12	4	3	6	4	8	6	6	1	4	12	0	18	7	5
22	4	1	1	4	2	2	5	5	7	26	11	7	13	12	26	12	17	3	12	24	18	0	21	11
23	3	0	1	5	1	1	2	3	5	18	13	7	8	11	10	5	6	1	3	7	7	21	0	7
24	1	0	0	2	0	1	1	2	2	8	6	5	7	4	4	3	3	0	1	4	5	11	7	0

Tabela 6.1: Matriz OD da rede sioux falls. As linhas representam as zonas de origem e as colunas as zonas de destino.