UNISINOS

Programa de Pós-Graduação em

Computação Aplicada

Mestrado Acadêmico

Lincoln Vinicius Schreiber

Aprendizado por Reforço Profundo Explicável: Um Estudo com Controle Semafórico Inteligente

São Leopoldo, 2022

Lincoln Vinicius Schreiber

APRENDIZADO POR REFORÇO PROFUNDO EXPLICÁVEL:
um estudo com controle semafórico inteligente

Dissertação apresentada como requisito parcial
para a obtenção do título de Mestre pelo
Programa de Pós-Graduação em Computação
Aplicada da Universidade do Vale do Rio dos
Sinos — UNISINOS

Orientador:
Prof. Dr. Gabriel de Oliveira Ramos

São Leopoldo
2022

S378a Schreiber, Lincoln Vinicius.
Aprendizado por reforço profundo explicável : um estudo com controle semafórico inteligente / Lincoln Vinicius Schreiber. – 2022.
90 f. : il. ; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, 2022.
"Orientador: Prof. Dr. Gabriel de Oliveira Ramos."

1. Aprendizado por reforço profundo. 2. IA explicável. 3. Controle semafórico adaptativo. 4. Sistema multiagente. I. Título.

CDU 004

Dados Internacionais de Catalogação na Publicação (CIP)
(Bibliotecária: Amanda Schuster – CRB 10/2517)

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001 / This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Para minha família.

RESUMO

Com o rápido aumento dos níveis de urbanização, tornou-se ainda mais evidente o problema do congestionamento para a sociedade, o meio ambiente e a economia. Uma abordagem prática para aliviar este problema é o controle semafórico adaptativo, do inglês *Adaptive Traffic Signal Control* (ATSC). A utilização de algoritmos de aprendizado por reforço profundo mostrou grande potencial para esse controle. Entretanto, tais métodos podem ser vistos como caixas pretas, visto que suas políticas aprendidas não são facilmente compreensíveis ou explicáveis. Essa falta de explicabilidade dos algoritmos pode estar limitando seu uso em condições reais. Um framework que pode fornecer explicações para qualquer modelo de aprendizado profundo é o SHAP. Ele considera os modelos como caixas pretas e utiliza técnicas post-hoc para explicá-los, fornecendo explicações baseadas na resposta desse modelo com diferentes entradas, sem analisar ou entrar em pontos internos (tais como parâmetros e arquitetura). O então estado da arte, para uso do SHAP com um algoritmo de aprendizado por reforço profundo para controlar semáforos, consegue demonstrar consistência na lógica da tomada de decisão do agente, apresenta também que o agente reage diferentemente conforme o tráfego de cada pista. Todavia, apresenta algumas limitações na explicabilidade encontrada e não consegue demonstrar de forma intuitiva a relação de alguns sensores com as ações escolhidas pelo agente. Além disso, precisa apresentar diversas figuras para entender o impacto dos estados nas possíveis ações.

Este trabalho apresenta duas abordagens baseadas no algoritmo Deep Q-Network capaz de explicar a política aprendida através do framework SHAP. Nossa abordagem considera duas técnicas distintas para aproximação de função: XGBoost e Multi-Layer Perceptron. Cada abordagem passou por um processo de estudo e otimização de seus hiperparâmetros. O ambiente foi caracterizado como um MDP e modelado de duas formas diferentes, chamadas MDP Cíclico e MDP Seletor. Cada uma dessas modelagens permitiu escolher diferentes ações e ter representações diferentes do ambiente. Por meio do framework SHAP, ambas abordagens puderam apresentar o impacto das *features* em cada ação, o que promove a compreensão de como o agente se comporta diante das diferentes condições de tráfego. Este trabalho também apresenta uma descrição sobre a aplicação de IA Explicável no controle semafórico inteligente, demonstrando como interpretar o modelo e as limitações da abordagem. Além disso, como resultado final, as abordagens melhoraram o tempo de viagem, a velocidade e o *throughput* em dois cenários distintos, superando os *baselines* FixedTime, SOTL e MaxPressure.

Palavras-chave: Aprendizado por reforço profundo. IA explicável. Controle semafórico adaptativo. Sistema multiagente.

ABSTRACT

With the fast increase in urbanization levels, the problem of congestion has become even more evident for society, the environment, and the economy. One practical approach to alleviating this problem is adaptive traffic signal control (ATSC). Deep reinforcement learning algorithms have shown great potential for such control. However, these methods can be viewed as black boxes since their learned policies are not easily understood or explainable. The lack of explainability of these algorithms may be limiting their use in real-world conditions. One framework that can provide explanations for any deep learning model is SHAP. It considers models as black boxes and explains them using post-hot techniques, providing explanations based on the response of that model with different inputs, without analyzing or going into internal points (such as parameters and architecture). The state of the art for using SHAP with a deep reinforcement learning algorithm to control traffic lights can demonstrate consistency in the logic of the agent's decision making, also presenting the reaction according to the traffic in each lane. However, it could not demonstrate the relation of some sensors with the chosen action intuitively and needed to present several figures to understand the impact of the state on the action.

This paper presents two approaches based on the Deep Q-Network algorithm to explain the policy learned through the SHAP framework. The first uses the XGBoost algorithm as a function approximation, and the second uses a neural network. Each approach went through a process of studying and optimizing its hyperparameters. The environment was characterized as an MDP, and we modeled it in two different ways, namely Cyclic MDP and Selector MDP. These models allowed us to choose different actions and have different representations of the environment. Both approaches presented the impact of features on each action through the SHAP framework, which promotes understanding of how the agent behaves under different traffic conditions. This work also describes the application of Explainable AI in intelligent traffic signal control, demonstrating how to interpret the model and the limitations of the approach. Furthermore, as a final result, our methods improved travel time, speed, and throughput in two different scenarios, outperforming the FixedTime, SOTL, and MaxPressure baselines.

Keywords: Deep reinforcement learning. Explainable AI. Adaptive traffic signal control. Multi-agent system.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ANN	Rede neural artificial (em inglês: *Artificial Neural Network*).
ATSC	Controle semafórico adaptativo (em inglês: *Adaptive Traffic Signal Control*).
DRL	Aprendizado por reforço profundo (em inglês: *Deep Reinforcement Learning*).
DVU	Unidade de veículo condutor (em inglês: *driver-vehicle unit*).
EUA	Estados Unidos da América.
LIME	Explicações locais interpretáveis agnósticas a modelo (em inglês: *Local Interpretable Model-agnostic Explanations*).
MDP	Processo de decisão de Markov (em inglês: *Markov Decision Process*).
RL	Aprendizado por reforço (em inglês: *Reinforcement Learning*).
RL-TSC	Aprendizado por reforço para controle semafórico (em inglês: *Reinforcement Learning for Traffic Signal Control*).
SHAP	SHapley Additive exPlanations.
TD	Diferenças temporais (em inglês: *temporal-difference*).
VDF	*Volume-Delay Function*.
XAI	IA explicável (em inglês: *Explainable AI*).

SUMÁRIO

1 INTRODUÇÃO

De acordo com o relatório de mobilidade urbana da Texas A&M Transportation Institute (SCHRANK; EISELE; LOMAX, 2019), o congestionamento do tráfego urbano é um problema grave tanto para o ambiente quanto para a sociedade e a economia. Os dados de 2017 desse relatório sugerem que, considerando 494 áreas urbanas dos Estados Unidos da América (EUA), os congestionamentos fizeram os americanos gastarem 3,3 bilhões de galões (12,5 bilhões de litros) de combustível a mais que o necessário, além de adicionar 8,8 bilhões de horas em tempo desperdiçado decorrente de congestionamentos. Tal cenário gerou um prejuízo de aproximadamente 166 bilhões de dólares em termos de combustível e tempo.

Ainda segundo Schrank, Eisele e Lomax (2019), foram gastos em média 54 horas e 21 galões (79,5 litros) de combustível no congestionamento por veículo, o que representa um custo anual médio de 1010 dólares por veículo. Para efeitos de comparação, um trajeto que duraria 20 minutos em tempo normal (sem congestionamentos) costuma-se levar 34 minutos, em média, quando há congestionamentos.

Existem diversas abordagens que buscam reduzir congestionamentos. Segundo Bazzan e Klügl (2013), uma forma de reutilizar com eficiência, a infraestrutura já existente, é utilizar controle semafórico inteligente, do inglês *intelligent traffic signal control*. Em princípio, a reutilização da infraestrutura permite implementar o sistema com custos relativamente baixos quando comparado com outras abordagens. Além disso, os motoristas e pedestres já possuem aceitação sobre tais sinais de trânsito. Essa abordagem também é conhecida, na literatura, como controle semafórico adaptativo, do inglês *Adaptive Traffic Signal Control* (ATSC)[1] (MANNION; DUGGAN; HOWLEY, 2016; WEI et al., 2019a).

Entretanto, estabelecer um tempo ótimo para os semáforos não é uma tarefa simples. Mesmo em sistemas mais comuns como os sistemas baseados em regras, a tarefa é considerada complexa de mais para ser resolvida manualmente por seres humanos. Uma forma automatizada de resolver esse problema é utilizar inteligência artificial. Segundo Mannion, Duggan e Howley (2016), aplicações de *machine learning* têm se tornado cada vez mais comuns na pesquisa ATSC nos últimos anos.

O uso de aprendizado por reforço, do inglês *Reinforcement Learning* (RL), nessa área tem demonstrado resultados promissores. Aprendizado por reforço para controle semafórico, do inglês *Reinforcement Learning for Traffic Signal Control* (RL-TSC), tem vários benefícios, como a capacidade de se adaptar em tempo real para melhorar continuamente seu desempenho, considerando rapidamente as mudanças no fluxo de tráfego. Outro ponto interessante é que os agentes RL-TSC conseguem aprender estratégias ATSC de forma autônoma (MANNION; DUGGAN; HOWLEY, 2016; BAZZAN, 2009).

[1] Nesse trabalho, as siglas são definidas com base na escrita original (na literatura estrangeira). Por exemplo, utilizamos ATSC ao invés de usar CAST para representar o controle semafórico adaptativo, visto que, sua escrita original é *Adaptive Traffic Signal Control*.

Em RL, um agente precisa aprender um comportamento que maximiza seu retorno (recompensa) apenas interagindo com o ambiente (SUTTON; BARTO, 2018). Nesse caso, o agente não tem conhecimento prévio sobre como funciona o ambiente e nem como suas ações afetam o mesmo. Essas configurações se assemelham ao problema de ATSC, onde o controlador de uma interseção precisa otimizar o tráfego sem conhecer antecipadamente a dinâmica do fluxo dos veículos ou o impacto de suas escolhas sobre tal fluxo.

Na literatura, muitos trabalhos aplicam redes neurais profundas em conjunto com técnicas de aprendizado por reforço (como *Deep Q-learning* (GENDERS; RAZAVI, 2016)) para otimizar a tomada de decisões. Tal abordagem é chamada de aprendizado por reforço profundo, do inglês *Deep Reinforcement Learning* (DRL). Apesar das diferentes abordagens desenvolvidas para ATSC, tais métodos podem ser vistos como caixas pretas, uma vez que as políticas aprendidas não são facilmente compreensíveis ou explicáveis.

Segundo Gunning (2017) e Wollenstein-Betech et al. (2020) a falta de explicabilidade de modelos de inteligência artificial representa uma grande preocupação em cenários reais, como o tráfego. Como as políticas aprendidas não podem ser compreendidas ou reguladas pelo ser humano, a aplicabilidade de tais métodos fica limitada. Apenas alguns trabalhos abordam formas de explicabilidade de seus modelos (AULT; HANNA; SHARON, 2020; RIZZO; VANTINI; CHAWLA, 2019). Todavia, apesar de Ault, Hanna e Sharon (2020) apresentarem um algoritmo baseado no DQN que possui uma função de controle regulável, seu modelo apresenta desempenho inferior aos algoritmos padrões de DRL. Já Rizzo, Vantini e Chawla (2019) propõem o uso de um framework de explicabilidade para explicar as decisões do seu modelo, mas apresentam algumas limitações na explicabilidade encontrada.

A área de IA explicável, do inglês *Explainable AI* (XAI), tem se tornado amplamente ativa nos últimos anos (ADADI; BERRADA, 2018). A pesquisa sobre XAI visa promover a produção de modelos explicáveis de alto desempenho (GUNNING, 2017). Assim, além de modelos explicáveis de forma intrínseca (facilmente interpretável, transparente), vários frameworks[2] de explicabilidade foram introduzidas nos últimos anos. Por exemplo, Ribeiro, Singh e Guestrin (2016) apresentaram explicações locais interpretáveis agnósticas a modelo, do inglês *Local Interpretable Model-agnostic Explanations* (LIME), e Lundberg e Lee (2017) apresentaram o SHapley Additive exPlanations (SHAP).

O framework SHAP (SHapley Additive exPlanations), pode fornecer explicações para qualquer modelo de aprendizado profundo. O framework unifica em um único modelo diversas propriedades de outros algoritmos. Além disso, o SHAP utiliza técnica post-hoc e trata os modelos como uma caixa preta, fornecendo explicações baseadas na resposta desse modelo com diferentes entradas, sem analisar ou entrar em pontos internos (tais como parâmetros e arquitetura). Dada sua capacidade de explicar modelos, o SHAP tem sido adotado em diversos contextos, como na modelagem de fatores de mortalidade nos EUA (LUNDBERG et al., 2020)

[2]Framework pode ser considerado um conjunto de conceitos e estruturas para resolver um problema específico. O uso dessa palavra se dá pela familiaridade da área da computação com a palavra, e por não existir uma palavra única com o mesmo significado em português.

e no auxílio a anestesistas durante cirurgias (LUNDBERG et al., 2018).

Visando reduzir o congestionamento através do uso eficiente da estrutura viária existente, diversos trabalhos na literatura adotam técnicas de aprendizado por reforço no controle semafórico inteligente. Entretanto, como descrito, poucas abordagens focam em explicar as políticas do agente. Motivados por esse desafio, neste trabalho introduzimos duas abordagens de aprendizado por reforço profundo para o controle semafórico inteligente. Com o uso de técnicas de explicabilidade post-hoc conseguimos otimizar o congestionamento com um desempenho similar ao do estado da arte. Como diferencial, permitimos a explicação das políticas aprendidas pelos modelos.

1.1 Questão de Pesquisa e Hipóteses

Com base no contexto discutido anteriormente, a questão de pesquisa deste trabalho é: como otimizar o controle semafórico de forma eficiente utilizando aprendizado por reforço profundo sem abdicar da explicabilidade da política?

A partir dessa pergunta, a seguinte hipótese foi levantada: um modelo que utiliza um algoritmo de aprendizado por reforço profundo, que apesar de não ser explicável por si só, pode se encaixar nos padrões necessários para utilizar alguma abordagem de explicabilidade, seja ela intrínseca ou post-hoc. Com isso, o algoritmo pode atingir um desempenho aceitável comparado com algoritmos do estado da arte e ainda assim obter políticas explicáveis.

1.2 Objetivos e Contribuições Científicas

O objetivo geral desse trabalho é apresentar um ou mais modelos que permitam reduzir o congestionamento com dados reais, tendo um desempenho competitivo a outros algoritmos estado-da-arte. Além disso, os modelos devem possibilitar que suas políticas aprendidas apresentem o máximo de informações para sua explicação.

Para que o objetivo geral seja atingido, se destacam como objetivos específicos: (i) utilizar uma abordagem de aprendizado por reforço profundo para otimizar o controle semafórico; (ii) considerar dados e restrições que sejam realistas e condizentes com o mundo real; (iii) utilizar um modelo intrínseco ou utilizar o framework SHAP para apresentar formas de explicabilidade do modelo e suas políticas; (iv) estabelecer explicações para que o modelo seja considerado interpretável; (v) Apresentar limitações da explicabilidade utilizada e o impacto que ela pode ter no desempenho dos modelos.

As **principais contribuições** deste estudo são:

- A comparação sob a ótica de desempenho e explicabilidade, em dois diferentes cenários, de dois diferentes métodos de aproximação de função para um algoritmo de aprendizado por reforço profundo no controle semafórico adaptativo.
- A implementação de um novo *pipeline* utilizando diferentes modelagens do ambiente (MDPs), *experience replay* e aproximadores de função (XGBoost e redes neurais) para o Q-Learning.
- A identificação das possibilidades e limitações do uso de modelos post-hoc para o contexto do controle semafórico inteligente.
- Uma apresentação unificada de todas as ações, em um determinado estado, com as suas respectivas contribuições locais em uma única imagem, permitindo simplificar a interpretação dos valores SHAP no atual contexto.

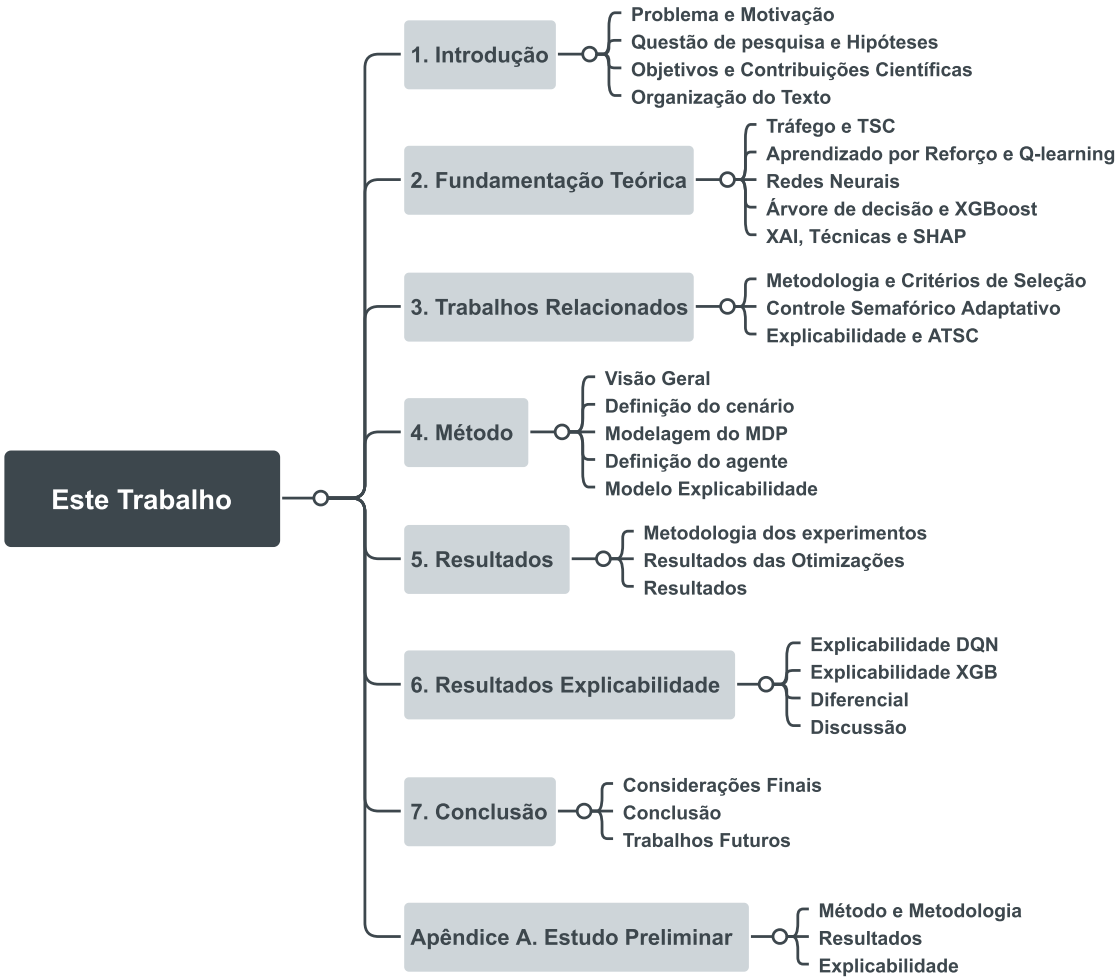
1.3 Organização do texto

Para uma melhor compreensão e separação dos conteúdos apresentados neste trabalho, a Figura 1 mostra a divisão dos capítulos e seus principais tópicos. Além da introdução do trabalho no Capítulo 1, temos:

- Capítulo 2: apresenta a **fundamentação teórica**, com uma visão geral dos conceitos e fundamentos de trânsito, aprendizado por reforço profundo, Q-learning, redes neurais, arvores de decisão, XGBoost, *explainable ai* e framework SHAP.
- Capítulo 3: descreve uma relação de **trabalhos relacionados** aos principais objetivos desse estudo, assim como o processo de identificação desses trabalhos. Além de destacar as abordagens clássicas e apresentar abordagens que utilizam aprendizado por reforço profundo e SHAP.
- Capítulo 4: visão geral do **método**, algoritmo proposto, modelagem do ambiente e definição do agente.
- Capítulo 5: apresenta os **resultados** do trabalho, incluindo a metodologia de avaliação, metodologia dos experimentos, resultados das otimizações realizadas e resultados comparativos dos modelos.
- Capítulo 6: resultados de **explicabilidade** para o XGBoost, resultados para o DQN, discussão sobre as limitações encontradas para a explicabilidade no controle semafórico.
- Capítulo 7: apresenta as principais contribuições e **conclusões** obtidas, além dos trabalhos futuros.

- Apêndice A: apresenta o **estudo preliminar** realizado antes dos resultados apresentados nos capítulos 5 e 6. Contém o método, a metodologia e os resultados do modelo e das explicações.

Figura 1: Organização do Trabalho e Principais Tópicos.



Fonte: Elaborado pelo autor

2 FUNDAMENTAÇÃO TEÓRICA

Para um melhor entendimento dos conceitos relacionados ao presente trabalho, este capítulo visa apresentar alguns termos gerais relacionados ao conteúdo base desta pesquisa. Além disso, é importante ressaltar que em todas as seções são recomendados artigos e livros para completar a leitura sobre os assuntos citados, visto que alguns conceitos são apresentados brevemente.

A Seção 2.1 tem objetivo de introduzir os conceitos e termos do tráfico e o controle semafórico, além de apresentar as necessidades das simulações e alguns simuladores. Na Seção 2.2 são apresentados os fundamentos do aprendizado por reforço, é introduzido o algoritmo Q-learning e também o conceito de aproximação de função. Redes Neurais e seus principais conceitos são apresentados na Seção 2.3. A Seção 2.4 apresenta árvores de decisões e o algoritmo XGBoost. Por sua vez, na Seção 2.5 são apresentados os conceitos de *blackbox*, escopo e tipos de explicação, além de introduzir a framework SHAP e o valor Shapley. Por fim, uma breve discussão sobre os conceitos é feita na Seção 2.6.

2.1 Tráfego e Controle Semafórico

O tráfego pode ser visto sob diversos panoramas, e de maneira geral, pode-se buscar diversas formas de tentar resolver os problemas de congestionamento. Segundo Bazzan e Klügl (2013) podemos observar esse fluxo principalmente em duas formas, baseado na demanda (veículos), ou na oferta (infraestrutura); ambos serão explicados abaixo de maneira resumida. Entretanto, para definições mais aprofundadas, recomendam-se os capítulos 3 e 4 de Bazzan e Klügl (2013).

A infraestrutura (oferta) é o que a rede viária pode oferecer, como ruas, pontes, viadutos, cruzamentos. Com isso, pode-se descrever o sistema de tráfego do ponto de vista de componentes físicos, sendo composto por nós (*nodes*), representando cruzamentos, e por ligações (*links*) representando as ruas e estradas.

Diferente da infraestrutura, a demanda não é fixa. Os veículos que utilizam a infraestrutura variam com frequência, pois essa demanda descreve o comportamento dos usuários e suas escolhas. Consequentemente, a demanda é altamente diferenciada pela hora do dia, dia da semana, finalidade, tipos de carga, importância da duração da viagem etc.

Esse trabalho é focado em como melhorar a oferta, e consequentemente resolver o problema de congestionamento do ponto de vista da infraestrutura. Algumas alternativas para minimizar o congestionamento pela infraestrutura podem ser: modificar avenidas, duplicar ruas, mudar sentido de vias, adicionar semáforo, mudar o tempo dos semáforos, adicionar rotatória etc. Para isso, também assumimos que a demanda (veículos) mantém suas origens e destinos, sem modificar o seu comportamento (por exemplo, continuar saindo no mesmo horário), apenas se adaptam as mudanças realizadas na infraestrutura.

Uma das formas de lidar com a infraestrutura é utilizando o controle semafórico. Assim como dito na introdução, segundo Bazzan e Klügl (2013) tradicionalmente este é o foco da

maioria dos trabalhos da área. Podemos resolver o controle semáforo baseado em regras ou baseado em algumas lógicas mais complexas, mas isso tem um desempenho inferior quando se tenta aplicar nas condições reais, com fluxo variado. Por isso é necessário um estudo sobre esse controle, de uma forma que ele consiga se adaptar (ser inteligente) e é esse então o objetivo da subseção seguinte.

2.1.1 Controle Semafórico

Uma forma de enviar simples mensagens para controlar o tráfego de forma eficiente e segura é utilizando semáforos. Normalmente, os controladores desses semáforos estão conectados a luzes que informam os motoristas quando podem avançar ou quando devem parar. Um breve estudo é apresentado abaixo, entretanto, para mais informações, recomendamos o livro Bazzan e Klügl (2013).

Existem duas maneiras de operação para um controlador de semáforos. A primeira operação considera o estado de um único cruzamento, onde o controlador pode ser categorizado como predefinido ou atuante. Quando categorizamos o controlador como predefinido, o mesmo segue sequências e tempos já definidos, e que não se alteram. Quando o controlador é atuante, recebe sinais de sensores que provem informações de volume de veículos, número de pedestres e outros eventos, e então o controlador responde a esses dados, podendo modificar a sequência e os tempos das fases.

A segunda operação seria determinada no fluxo, considerando uma rota sincronizada. Essa maneira precisa de um controlador mestre ou uma implementação de colaboração entre os controladores de cada intersecção, permitindo assim realizar um movimento progressivo por diversos cruzamentos consecutivos. Um exemplo desse controle é a implementação da *green wave*, onde diversos sinais ficam verdes em momentos específicos permitindo que grupos de veículos atravessem várias intersecções sem precisar parar (KOONCE; RODEGERDTS, 2008).

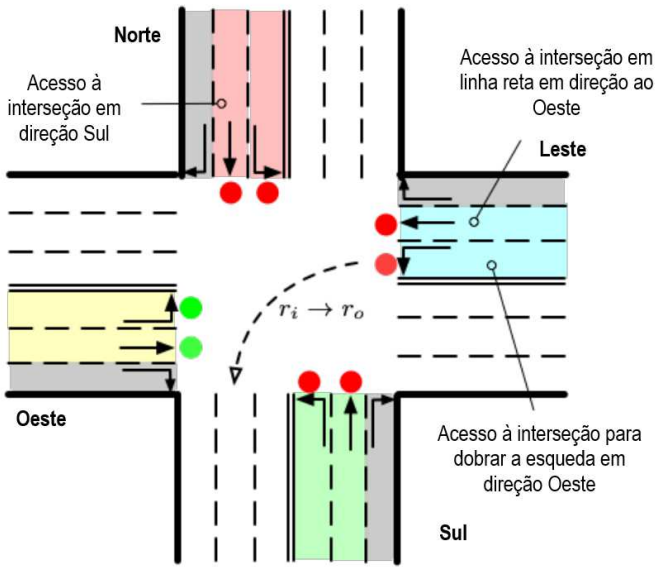
Indiferente da maneira escolhida para operar os controladores, ambas podem ser definidas por um sistema manual, ou por um sistema inteligente. Definir o melhor estado de tempo para os semáforos não é uma tarefa simples, mesmo em sistemas mais comuns como os sistemas por regras, é considerada uma tarefa complexa para humanos resolverem manualmente. Com isso, diversos estudos são realizados para implementar técnicas que buscam encontrar o tempo ótimo para interseções sinalizadas. Um levantamento sistemático desses trabalhos é descrito na Seção 3.1.

Considerando diversas possibilidades de controle semafórico inteligente, fica óbvio que não podemos testar qualquer hipótese existente na prática, pois isso pode ter um impacto negativo grande na vida das pessoas. Uma forma de validar esse tipo de controlador inteligente antes de ir para o mundo real é através de simulação, conforme discutido na Subseção 2.1.4.

2.1.2 Termos e Definições

A Figura 2 apresenta os movimentos e sinalizações do trânsito na visão de um cruzamento. A estrutura viária, os movimentos do tráfego e os sinais de trânsito são compostos pelos seguintes elementos: acesso, interseção, faixa, movimentos de trânsito, veículos, sinal de movimento, fase e sequência de fases. Os termos são descritos na lista abaixo.

Figura 2: Definições dos movimentos e sinalização de trânsito.



Fonte: Traduzido de Wei et al. (2019a)

- Acesso: faixa que liga uma entrada e uma saída a um cruzamento é referida como um acesso. Geralmente, há dois tipos de acesso: entrada e saída.

- Interseção e cruzamento[1]: união de diversos acessos, normalmente controlada por semáforos. A Figura 2 apresenta um cruzamento.

- Faixa: um acesso consiste em um conjunto de faixas. Semelhante à definição de acesso, existem duas categorias de faixas: faixas de entrada e de saída.

- Movimento de trânsito: um movimento de trânsito refere-se ao movimento dos veículos que se deslocam de um acesso ao outro, definido por $(r_i \rightarrow r_o)$, onde r_i é a faixa de entrada e r_o a faixa de saída. Normalmente são definidos como movimento reto, dobrar a esquerda e dobrar a direita.

- Veículo e carro[1]: elemento que se movimenta pelas faixas do cruzamento.

[1]Durante esse trabalho, os termos indicados são considerados sinônimos e são utilizados alternadamente.

- Sinal de movimento: é definido como o deslocamento permitido ou proibido baseado no semáforo da faixa, podendo estar em verde (permitido) ou vermelho (proibido). Na Figura 2, os movimentos permitidos são $O \rightarrow N$ ($\rightharpoonup$) e $O \rightarrow L$ ($\rightarrow$).

- Fase: uma fase é uma combinação de sinais de movimento. Na Figura 2, as fases são retratadas com as cores amarelo, verde, azul e vermelho. Por exemplo, a fase amarela possui os movimentos $O \rightarrow N$ ($\rightharpoonup$) e $O \rightarrow L$ ($\rightarrow$).

- Sequência de fases: define um conjunto de fases e sua ordem de mudanças.

2.1.3 Informações Complementares

Nessa seção, apresentamos algumas informações que devem ser levadas em consideração sobre o trânsito. Após uma fase, deve-se determinar um tempo de amarelo e um tempo de vermelho (para todas as fases). Isso permite que os carros consigam concluir o movimento e também parem em segurança. Também é necessário definir um tempo mínimo de verde para que os pedestres possam se movimentar pelo cruzamento. Além disso, em alguns cruzamentos, o movimento de dobrar à direita é permitido em qualquer fase, como pode-se observar na Figura 2, os movimentos de dobrar à direita não estão associados a nenhuma fase.

Normalmente, o controlador do cruzamento decide a fase que está ativa no momento. Com isso, os movimentos são escolhidos indiretamente. Caso o controlador defina que um movimento específico deva ser executado, deve-se escolher uma fase que compõe esse movimento para efetivamente permitir ele. Na Figura 2, os movimentos permitidos são $O \rightarrow N$ ($\rightharpoonup$) e $O \rightarrow L$ ($\rightarrow$), e só estão permitidos porque a fase amarela foi a escolhida pelo controlador do cruzamento[2].

2.1.4 Simulação e Simuladores

Como descrito na Seção 2.1.1, não podemos testar qualquer hipótese na prática, visto que isso pode ter um impacto negativo na vida das pessoas. Uma forma de validar os algoritmos e as abordagens para o controle semafórico adaptativo, do inglês *Adaptive Traffic Signal Control* (ATSC) é simular instâncias de cruzamentos com fluxos de veículos probabilísticos ou adquiridos do tráfego real.

Em termos gerais, existem três principais modelos de simulação de fluxo de tráfego. Nessa seção serão apresentados, de forma resumida, cada um dos três, com suas respectivas características e simuladores existentes. Para um entendimento mais profundo sobre ambos os modelos, recomendamos o capítulo 6 do livro Bazzan e Klügl (2013), algumas leituras complementares também são indicadas no texto.

[2]No exemplo dado, podemos garantir que a fase amarela foi escolhida, pois é a única que possui ambos os movimentos ativos em sua composição.

Existem modelos de simulação de fluxo de tráfego **microscópico**, os quais focam na mobilidade individual de cada veículo (BARCELÓ et al., 2010; HARRI; FILALI; BONNET, 2009), chamada de unidade de veículo condutor, do inglês *driver-vehicle unit* (DVU). O fluxo do tráfego é formado pela simulação dessas unidades e suas interações. As variáveis principais adquiridas desse modelo são posição, velocidade e aceleração de cada DVU. Existem diversos modelos comerciais e não comerciais de simulação microscópica. o simulador *opensource* SUMO, do inglês *Simulation of Urban MObilit*[3], é um exemplo deles. Além disso, Segundo Bazzan e Klügl (2013), os modelos microscópicos devem ser buscados ao ter os seguintes objetivos[4]:

- Considerar que uma única DVU afeta o fluxo, seja baseado em suas decisões individuais, iteração com a infraestrutura ou com outros veículos.
- Testar a eficiência de mudanças particulares para otimização do tráfego, por exemplo, simular o efeito da mudança dos limites de velocidade da pista.
- Modelar o comportamento humano ao dirigir, incluindo erros, tempo de reação, entre outros, ou a iteração entre diferentes tipos de unidades no tráfego (carro, ônibus, ciclista, pedestre, etc.).

A simulação de fluxo de tráfego **macroscópico** foca no fluxo completo da rodovia, considerando a densidade geral do tráfego, distribuição de veículos e restrições (cruzamentos e semáforos) (HAMAN et al., 2017; BARCELÓ et al., 2010). Essa forma é descrita de forma análoga aos líquidos ou gases em movimento. Assim como a simulação microscópica, temos como uma das variáveis do ambiente a velocidade e a aceleração. Entretanto, ao invés de individual, ela é a média do aglomerado de indivíduos presentes. Além disso, a densidade do tráfego, o fluxo e a variação da velocidade também são variáveis.

Tipos de veículos, atividades individuais (mudança de faixa e velocidade) são difíceis de serem descritos macroscopicamente. Entretanto, por essa visão agregada, os modelos macroscópicos são mais rápidos de calcular e executar. Então se não tem como objetivo o indivíduo em si, ou, trata-se de uma análise de uma grande rede, é aconselhável utilizar um modelo macroscópico. Não são utilizados simuladores no modelo macroscópico. Para descrever a relação entre os volumes de tráfego e o tempo de viagem dos veículos, são utilizadas fórmulas do tipo *Volume-Delay Function* (VDF).

A simulação de fluxo de tráfego **mesoscópico** (ou híbrido) preenche lacunas entre os modelos macroscópicos e microscópicos, combinando vantagens de ambos, focando em uma mobilidade dos veículos, como no microscópico, mas voltado para um grupo, tentando se aproxima da distribuição dos mesmos (como no macroscópico). Muitas vezes, eles são combinados com um procedimento baseado em simulação para atribuição dinâmica de tráfego, porque um modelo

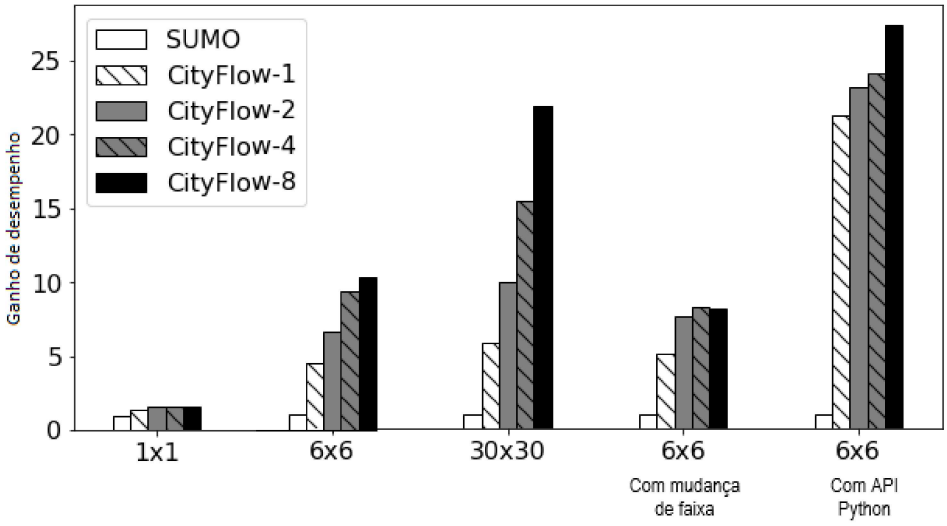
[3]https://www.eclipse.org/sumo/

[4]Os objetivos representados foram escolhidos e resumidos pelo autor considerando a relevância e proximidade com o assunto principal do trabalho. No documento original, existem mais objetivos, e os mesmos são maiores e mais complexos.

mesoscópico pode ser simulado rapidamente, mesmo quando as escolhas de rota dos veículos são explicitamente consideradas (HAMAN et al., 2017; BARCELÓ et al., 2010). Todavia, os modelos mesoscópicos são menos detalhados que os microscópicos. Um exemplo de simulador mesoscópicos é o MEZZO (BURGHOUT; KOUTSOPOULOS; ANDREASSON, 2005).

Neste trabalho assumimos a perspectiva microscópica, dado que precisamos simular precisamente os veículos para viabilizar o controle semafórico. Dentre todos os simuladores estudados, dois deles se destacam dos demais: o **SUMO** e o **CityFlow**. Apesar do SUMO ser o mais utilizado no estado da arte atualmente, o CityFlow tem sido amplamente aceito e utilizado em trabalhos recentes. Além disso, a Figura 3 apresenta uma comparação de desempenho entre ambos os simuladores em diferentes cenários. Baseado nessa comparação, decidiu-se utilizar o simulador CityFlow.

Figura 3: Comparação de desempenho entre o SUMO e o CityFlow com diferentes números de threads (1, 2, 4, 8). Os ambientes testados variaram desde uma pequena rede rodoviária 1x1 até a rede rodoviária 30x30.



Fonte: Traduzido de Zhang et al. (2019)

2.2 Aprendizado por Reforço

Aprendizado por reforço, do inglês *Reinforcement Learning* (RL) é um ramo do aprendizado de máquina que, segundo Sutton e Barto (2018), permite que um agente aprenda com interações em um ambiente. Os principais componentes do RL são o agente e o ambiente. O agente é o tomador de decisões que precisa aprender como resolver uma determinada tarefa. O ambiente se refere ao mundo no qual o agente está situado, e com o qual ele interage.

Um problema em RL é normalmente formulado como um processo de decisão de Markov, do inglês *Markov Decision Process* (MDP) (BELLMAN, 1957; SUTTON; BARTO, 2018). Um MDP consiste em uma tupla $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$, onde:

- $\mathcal{S}$ representa o conjunto de estados do ambiente.
- $\mathcal{A}$ representa o conjunto de ações que o agente pode executar.
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ especifica uma função de transição probabilística.
- $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ é a função de recompensa esperada $r(s, a)$ que o agente recebe após executar a ação a no estado s.

Em cada etapa, o agente seleciona uma ação $a \in \mathcal{A}$ no estado $s \in \mathcal{S}$. Ao executar essa ação, o agente obtém uma recompensa r e termina no estado $s' \in \mathcal{S}$. Este processo é então repetido para o próximo estado até que um estado terminal seja atingido (assumindo que o problema seja episódico). A ideia principal por trás deste processo se resume em derivar um mapeamento dos estados para ações que produzem um alto retorno (ou seja, a soma da recompensa esperada ao longo do tempo). Este mapeamento é chamado política e é denotado como π. O objetivo do agente é encontrar uma política ótima π^* que maximize seu retorno esperado.

Dependendo das informações disponíveis para o agente, as abordagens de RL podem ser classificadas em duas categorias: (i) baseada em modelos, do inglês *model-based* (quando a dinâmica do ambiente —$\mathcal{P}$ e $\mathcal{R}$— são levadas em consideração); (ii) sem modelo, do inglês *model-free* (quando essas dinâmicas são ignoradas).

Cada abordagem tem vantagens e desvantagens. Entretanto, para implementar com sucesso uma abordagem baseada em modelos, é necessário conhecer a função de transição $\mathcal{P}$, que é relativamente complexa de ser determinada, mesmo em sistemas simples e dificilmente vai estar disponível nos problemas do mundo real, como o controle semafórico (WIERING; VAN OTTERLO, 2012; RAMOS et al., 2020). Além disso, de acordo com o El-Tantawy, Abdulhai e Abdelgawad (2013), para problemas como controle de tráfego, o uso de uma abordagem baseada em modelos acrescenta complexidade desnecessária em comparação com abordagens sem modelo, mesmo podendo ser mais eficiente para lidar com ambientes não estacionários.

2.2.1 Q-Learning e Aproximação de Função

Um algoritmo de RL que tem sido amplamente utilizado no controle semafórico adaptativo e que utiliza diferenças temporais, do inglês *temporal-difference* (TD), é o Q-learning (WATKINS, 1989; WATKINS; DAYAN, 1992). É um algoritmo off-policy e o agente que utiliza o algoritmo Q-learning obtém informações do ambiente selecionando ações em um determinado estado. Além disso, podemos usar um processo que visa equilibrar o ganho de conhecimento (*exploration*) para o uso do conhecimento (*exploitation*). Uma estratégia amplamente utilizada para isso é a exploração ε-greedy, onde o agente escolhe uma ação randômica com probabilidade ε (*exploration*) ou escolhe a melhor ação, conforme o conhecimento já adquirido, com probabilidade $1 - \varepsilon$ (*exploitation*), onde $\varepsilon \in [0, 1]$.

Em particular, a regra de atualização do Q-learning é definida pela Equação 2.1, onde $\alpha \in [0, 1]$ denota a taxa de aprendizagem e $\gamma \in [0, 1[$ específica um fator de desconto em recompensas futuras.

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a) - Q_t(s_t, a_t)) \tag{2.1}$$

A escolha do α e γ desempenha um papel importante na convergência do algoritmo Q-learning. Altos valores de α tornar o processo de aprendizagem instável, já que a política é ajustada rapidamente na direção das novas observações. Por outro lado, os baixos valores para α evitam este problema ao custo de diminuir a velocidade de aprendizagem. O fator de desconto γ permite ao agente controlar quão importantes são as recompensas futuras em comparação com as imediatas (as recebidas no estado atual): quanto mais alto γ, mais importantes são as recompensas futuras. Estes parâmetros precisam ser ajustados de acordo com a tarefa em consideração.

Além disso, foi demonstrado por Watkins e Dayan (1992); Sutton e Barto (1998) que o algoritmo Q-learning garante convergência para uma política ótima se algumas condições forem mantidas, dentre elas destaca-se: (i) todos os pares de estados e ações devem ser visitado infinitas vezes. Com isso, cada ação deve ter uma probabilidade não zero de ser selecionada pela política em cada estado; (ii) a taxa de aprendizagem deve se aproximar a zero no infinito.

O algoritmo padrão de Q-learning assume que o estado e os espaços de ação são discretos. Neste sentido, a política pode ser representada como uma tabela onde cada linha armazena o valor Q para um determinado par de ação-estado. Entretanto, este tipo de abordagem torna-se impraticável quando o estado e os espaços de ação são contínuos ou grandes (em alguns casos infinitos). De modo a superar este problema, uma abordagem típica é aplicada para aproximar a política usando uma função parametrizada, que é conhecida como aproximação de função (SUTTON; BARTO, 2018). Tal função é tipicamente definida como uma rede neural, o que melhora a generalização da política e frequentemente acelera o processo de aprendizagem.

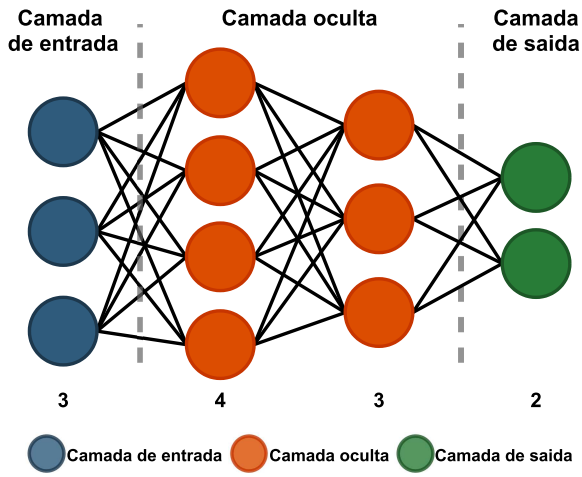
De fato, o chamado aprendizado por reforço profundo, do inglês *Deep Reinforcement Learning* (DRL), é responsável por muitos dos avanços recentes no campo do RL. Um algoritmo representativo do DRL é o Deep Q-Network (DQN) (MNIH et al., 2015), que usa uma rede neural profunda para se aproximar da política. A DQN treina sua rede neural usando repetição de experiências (do inglês *experience replay*) que consiste em armazenar tuplas de experiência em um *replay buffer* e usar esse *buffer* para ajustar os pesos da rede. Neste trabalho, usamos tanto o DQN para o controle de sinais de tráfego, quanto uma implementação de aproximação de função com o XGBoost. O uso de um algoritmo alternativo ao DQN se da pela possibilidade de comparação dos resultados de explicabilidade. A teoria de redes neurais é apresentada na Seção 2.3 e sobre XGBoost e árvores de decisões na Seção 2.4. Maiores detalhes sobre as implementações são apresentados dos algoritmos pode ser visto na Seção 4.4.

2.3 Redes Neurais Artificiais

Rede neural artificial, do inglês *Artificial Neural Network* (ANN), é uma técnica de aprendizado de máquina (ML) inspirada no cérebro humano (HAYKIN, 2007). Possuem a capacidade de aprender e reconhecer padrões através de um conjunto de neurônios que captam, processam e distribuem informações pela rede. Esses neurônios (também chamados de nós) podem ser alocados em uma ou mais camadas, interligados por várias conexões que são, geralmente, associadas a pesos. A história da ANN teve início com a criação do primeiro modelo matemático do neurônio biológico feito por McCulloch e Pitts (1943).

A Figura 4 apresenta uma rede neural artificial (ANN) multicamada totalmente conectada (*multilayer fullyconnected*). Em termos de topologia, uma ANN consiste em: (i) uma camada de entrada onde o número de neurônios (ou nós) corresponde ao número de variáveis que serão usadas para alimentar a rede neural; (ii) camadas ocultas, onde o número de camadas e o número de nós variam conforme os problemas; (iii) camada de saída, onde o número de nós corresponde a saída do modelo. Diferente das camadas de entrada e saída, não existe uma fórmula ou processo para determinar o número de neurônios nem o número de camadas que devemos ter na camada oculta. Ou seja, para um mesmo problema, uma ANN pode ter n camadas ocultas e cada camada oculta pode possuir k neurônios artificias (HAYKIN, 2007).

Figura 4: Diagrama de uma rede neural artificial (ANN).



Fonte: Elaborado pelo autor.

Ao observar isoladamente um neurônio das camadas ocultas ou da camada de saída, o que encontraremos é um único neurônio artificial. O neurônio artificial, é uma unidade de processamento de informação utilizado em redes neurais, sua arquitetura foi definida por (MCCULLOCH; PITTS, 1943). Os sinais de entrada, representados pelo conjunto $\{x_1, x_2, x_3, ..., x_m\}$, são somados de forma ponderada pelos pesos $\{w_1, w_2, w_3, ..., w_{jm}\}$, resultando em um potencial de ativação v. Além disso, esse potencial de ativação é somado cotextitbias), que possui o papel de ajudar o modelo a se adaptar melhor aos dados fornecidos. Esse resultado é passado para uma função de ativação que gera uma saída y. O m representa o número de entradas do

neurônio, os índices i e j se referem a neurônios diferentes na rede, onde o neurônio j se encontra à direita do neurônio i. As equações 2.2 e 2.3 representam matematicamente o neurônio artificial explicado (HAYKIN, 2007).

$$v_j(n) = \sum_{i=1}^{m} x_i * w_i + b \tag{2.2}$$

$$y_j(n) = \varphi_j(v_j(n)) \tag{2.3}$$

A função de ativação determina como se comporta a saída do neurônio perante os valores de entrada multiplicado pelos seus respectivos pesos (HAYKIN, 2007). Abaixo são apresentadas algumas das funções mais utilizadas:

- **Linear:** essa função tem como saída a própria entrada. A Equação 2.4 e a Figura 5a demonstram esse conceito.

$$\varphi(x) = x \tag{2.4}$$

- **ReLU:** unidade linear retificada, do inglês Rectified Linear Unit (ReLU). Matematicamente ela é definida pela Equação 2.5 e pode ser observada na figura 5b. É linear (identidade) para todos os valores positivos e 0 para todos os valores negativos, comparada com a sigmoide ela tem um custo computacional baixíssimo.

$$\varphi(x) = Max(0\ ,\ x) \tag{2.5}$$

- **Sigmoide:** é definida pela Equação 2.6. O parâmetro a é responsável por determinar a inclinação da função. A Figura 5c demonstra a função com o parâmetro a igual a 2.

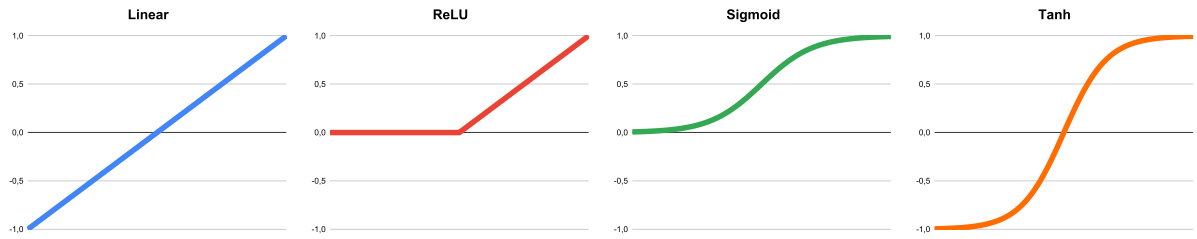
$$\varphi(x) = \frac{1}{1 + e^{-a*x}} \tag{2.6}$$

- **Tanh:** essa função tem como saída a tangente hiperbólica da entrada. A Equação 2.7 e a Figura 5d demonstram esse conceito.

$$\varphi(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.7}$$

Figura 5: Exemplo de funções de saída.

(a) Função Linear. (b) Função ReLU. (c) Função Sigmoide. (d) Função Tanh.



Fonte: Elaborado pelo autor.

2.4 Arvore de Decisão e XGBoost

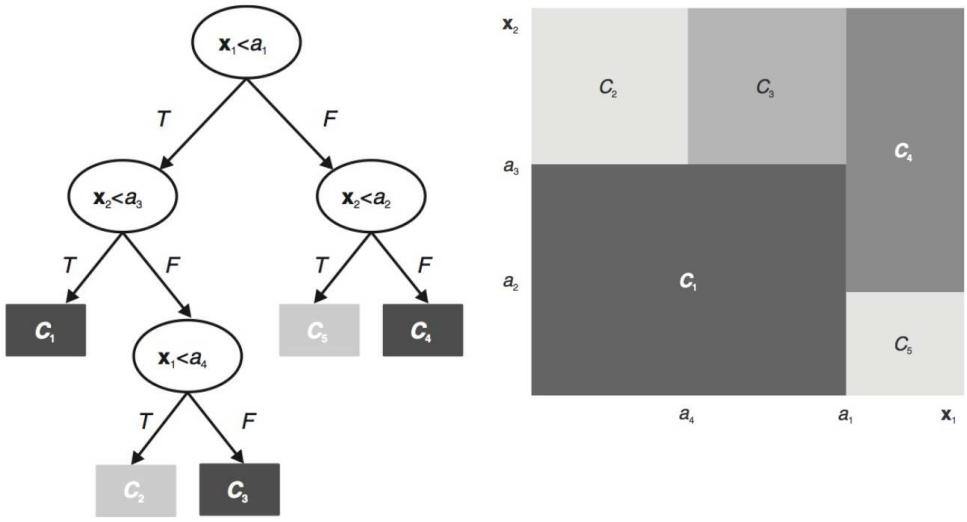
Para entender árvores de decisão, precisamos entender inicialmente a estrutura de dados chamada árvore. De modo geral, árvores são estruturas formadas por um conjunto de elementos que armazenam informações chamadas nós. Toda árvore possui um nó de ponto de partida chamado raiz, esse possui o maior nível hierárquico. Além disso, todo nó raiz possui ligações para outros elementos, esses denominados nós filhos. Os nós filhos podem possuir nós filhos, por sua vez, também podem possuir nós filhos. Um nó que não possuir um filho é conhecido como nó terminal, ou nó folha (CARVALHO et al., 2011).

Tendo as definições de árvore definida, árvores de decisão são métodos de aprendizado de máquina supervisionado que possuem uma ou diversas arvores que armazenam regras em seus nós (nó raiz e filhos), e os nós folhas representam as saídas do modelo, ou as “decisões” (por exemplo, um valor x de saída) (RUSSELL; NORVIG, 2002). Os valores de entrada e saída podem ser discretos ou contínuos. Uma decisão é tomada através do caminho seguido a partir do nó raiz até o nó folha, esse caminho, na verdade, é uma sequência de testes baseados nos valores de entrada do modelo.

Na Figura 6 temos a representação de uma árvore de decisão (Figura 6a) e a divisão correspondente no espaço (Figura 6b) definido pelas entradas $x1$ e $x2$. ‘T’ e ‘F’ representam que a lógica é verdadeira ou falsa, respectivamente. A_1 até A_4 representam números nos quais foram escolhidos os testes de cada nó. C_1 até C_5 são as saídas do modelo. Cada nó da árvore corresponde a uma região no espaço. As regiões definidas pelos nós folha da árvore são espaços mutuamente excludentes, e a união dessas regiões cobre todo esse espaço (CARVALHO et al., 2011).

Figura 6: Uma árvore de decisão e as regiões de decisão no espaço de objetos.

(a) Árvore de decisão exemplo. (b) Região de decisão.



Fonte: Carvalho et al. (2011)

Segundo Carvalho et al. (2011), uma árvore de decisão usa a estratégia de dividir para conquistar para resolver um problema de decisão. Um problema mais complexo é dividido em problemas mais simples, e assim sucessivamente. Por fim, a soluções desses subproblemas podem ser combinadas para produzir uma solução do problema inicial. Algoritmos como ID3 (QUINLAN, 1979), ASSISTANT (CESTNIK; KONONENKO; BRATKO, 1987), CART (BREIMAN et al., 1984) e C4.5 (QUINLAN, 1993) utilizam essa ideia.

O uso de árvores de decisões, segundo Russell e Norvig (2002), trazem muitas vantagens, como ser escalável para muitos dados, ser versátil ao utilizar dados contínuos ou discretos e ser de fácil compreensão. Todavia, as árvores podem ter uma precisão sub-ótima (parte disso é culpa da escolha gulosa), e pode ser muito custoso (em relação a tempo de execução) ter uma nova predição se a árvore for muito profunda. As árvores de decisões também são instáveis, se adicionar um novo exemplo que acabe modificando o teste do nó raiz, acabamos modificando a árvore por inteira.

Um algoritmo que vem resolvendo diversos dos problemas de árvores de decisão é o algoritmo **XGBoost**. Foi projetado para ser altamente eficiente, flexível e portátil. Ele implementa algoritmos de aprendizagem de máquinas com o framework de Gradient Boosting. É amplamente utilizado por cientistas de dados para alcançar ótimos resultados em pouco tempo (CHEN; GUESTRIN, 2016). O XGBoost funciona mais de dez vezes mais rápido do que outras soluções populares existentes, além de conseguir escalar bilhões de dados em ambientes distribuídos.

Segundo Chen e Guestrin (2016), a escalabilidade do XGBoost se deve a vários sistemas importantes e algumas otimizações algorítmicas. Dentre as importantes inovações do algoritmo, as que mais se destacam são: (i) o Algoritmo consegue lidar com dados esparsos (*sparse data*);

(ii) o algoritmo é otimizado para funcionar com computação paralela e distribuida; (iii) o algoritmo consegue processar conjuntos de dados muito grandes, mesmo aqueles que não cabem na memória principal de um computador (algoritmo conhecido como *out-of-core*, ou *external memory algorithm*); (iv) o algoritmo combina todas essas técnicas e consegue lidar com conjuntos de dados ainda maiores e com a menor quantidade de recursos.

2.5 Inteligência Artificial Explicavel (XAI) e Técnicas

Segundo Puiutta e Veith (2020), quanto mais poderosos e flexíveis os modelos de aprendizado de máquina (ML) ficam, mais opacos eles se tornam, tornando-se praticamente caixas pretas, onde não é possível compreender seus parâmetros. Além disso, Gunning (2017) e Wollenstein-Betech et al. (2020) apresentam que falta de explicabilidade representa uma grande preocupação em cenários do mundo real. Todavia, a pesquisa sobre IA explicável, do inglês *Explainable AI* (XAI), visa promover a produção de modelos com alto desempenho e são explicáveis (GUNNING, 2017). Nessa seção, apresentaremos os conceitos gerais e a taxonomia dos métodos de XAI, além de informações sobre a framework SHAP.

2.5.1 Fundamentos

Segundo Adadi e Berrada (2018), XAI é necessário por questões regulatórias, para benefícios comerciais ou por preocupações éticas. XAI é essencial para que os usuários possam entender, confiar apropriadamente e administrar efetivamente os resultados do modelo. Ainda assim, podemos classificar as necessidades de explicar um modelo em 4 razões: (i) explicar para justificar; (ii) explicar para controlar; (iii) explicar para melhorar; (iv) explicar para descobrir. Podemos categorizar os métodos de XAI baseado em dois fatores. Quando a informação é extraída, podendo ser intrínseco ou post-hoc. Baseado no escopo da explicação, podendo ser global ou local.

Uma informação importante que define se o método é intrínseco ou *post-hoc* é o momento em que a explicação é gerada/extraída do algoritmo. Um modelo intrínseco é um modelo que é construído facilmente interpretável (quando não é autoexplicativo). O *post-hoc* é o contrário, a explicação é obtida analisando o modelo após o treinamento, criando um segundo modelo para fornecer explicações para o modelo original. Segundo Puiutta e Veith (2020), um modelo *post-hoc* pode ser aplicado a um modelo intrínseco, mas o contrário não pode ser afirmado.

Segundo Molnar (2020), Arrieta et al. (2020) e Puiutta e Veith (2020), uma explicação local tenta responder o porquê do modelo ter feito certa predição/decisão naquele determinado momento. Além disso, uma explicação local tenta identificar as contribuições de cada *feature*[5] de entrada em cada saída do modelo. Por outro lado, explicação global explica o comportamento

[5] A palavra *feature* pode significar característica ou atributo de entrada. De modo a facilitar o entendimento, evidenciamos esse significado para o termo. Além disso, a partir desse momento, deixamos de destacar a palavra em itálico no texto.

geral do modelo, tentando explicar toda a lógica do modelo. Ainda assim, segundo Du, Liu e Hu (2019), as técnicas de interpretabilidade global levam os usuários a confiar em um modelo, enquanto as técnicas locais levam a confiar na predição.

A explicabilidade post-hoc é aplicada em modelos que não são facilmente interpretáveis, e recorre a diversos meios para melhorar sua interpretabilidade. Na Figura 7 podemos observar as seis técnicas de explicação post-hoc descritas por Arrieta et al. (2020). Além disso, diferentes técnicas podem ser aplicadas para as mesmas categorias de dados, modelos, e intenções do autor.

- **Explicação por texto**: as explicações por texto, segundo Bennetot et al. (2019), trazem explicações para o modelo gerando textos que ajudam explicar os resultados. Além disso, estão incluídos todos os métodos que geram símbolos que podem representar a funcionalidade do modelo (ARRIETA et al., 2020).

- **Explicação visual**: uma técnica de explicação visual busca a visualização do comportamento do modelo. Podem ser combinadas com outras técnicas para melhorar a compreensão. Além disso, diversos dos métodos existentes de visualização aplicam redução de dimensionalidade para facilitar na interpretação humana. É considerada a forma mais adequada para introduzir os usuários que não são familiarizados com *machine learning*.

- **Explicação local**: a ideia dessa técnica é segmentar o espaço de soluções, dando explicações para alguns subespaços menos complexos, mas relevantes para todo o modelo.

- **Explicação utilizando exemplo**: similar a como os seres humanos se comportam ao tentar explicar um determinado processo. As explicações utilizando exemplo consideram a extração de exemplos representativos que captam as relações internas e as correlações encontradas pelo modelo.

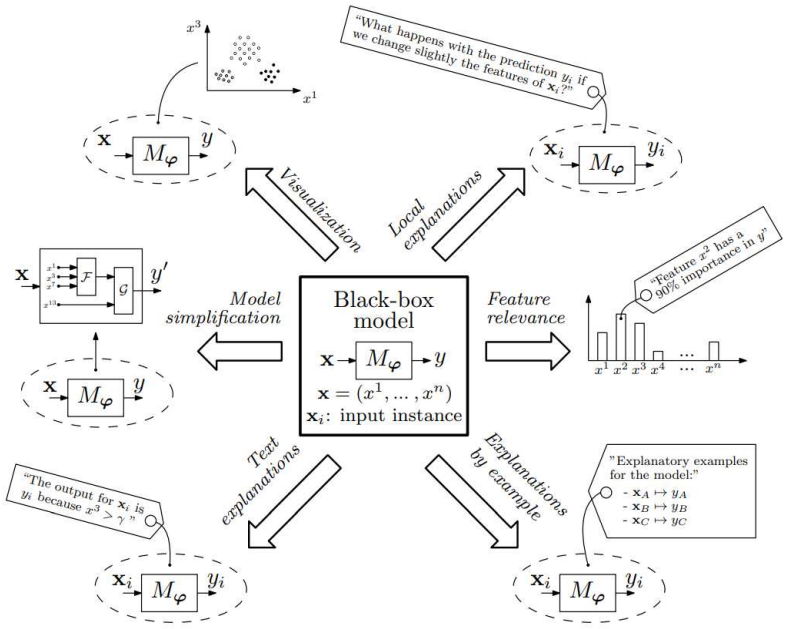
- **Explicação por simplificação**: denota as técnicas nas quais um sistema totalmente novo é construído com base no modelo a ser explicado. Esse novo modelo otimiza a semelhança com o anterior, geralmente reduz a complexidade, mas mantêm o desempenho similar.

- **Explicação da relevância das features**: procura esclarecer o funcionamento interno de um modelo através da atribuição de uma pontuação baseada no cálculo de relevância de suas variáveis. Os métodos de explicação da relevância das features pode ser considerado um método indireto para explicar o modelo.

2.5.2 SHAP (SHapley Additive exPlanations)

O SHAP (SHapley Additive exPlanations) de Lundberg e Lee (2017) é uma abordagem em teoria dos jogos para explicar as previsões individuais (saídas) de qualquer modelo de apren-

Figura 7: Diagrama conceitual mostrando as diferentes abordagens de explicabilidade post-hoc disponíveis.



Fonte: Arrieta et al. (2020)

dizado de máquina (ML). SHAP prove explicações locais baseado nos clássicos valores de Shapley.

Segundo Lundberg e Lee (2017), quando trabalhamos com modelos complexos, como modelos de aprendizagem profunda (*Deep Learning*), utilizar o próprio modelo para tentar explicar ou interpretar as saídas não ajuda, pois devido à sua complexidade é difícil de interpretar. Portanto, nesses casos, devemos procurar modelos que forneçam explicações. Além disso, segundo García e Aznarte (2020), quando se tenta interpretar um modelo complexo, somos forçados a encontrar um modelo simplificado para prover essas explicações.

Considere f como a predição original do modelo a ser explicado, e considere g como o modelo explicável. Nós nos concentramos em métodos locais projetados para explicar uma predição $f(x)$ com base em uma única entrada x. Normalmente os modelos de explicação simplificam as entradas x para x', onde essa entrada é mapeada por $x = h_x(x')$. Como Lundberg e Lee (2017) afirmam, métodos locais tentam garantir que $g(z') \approx f(h_x(z'))$ sempre que $z' \approx x'$. Segundo Molnar (2020), uma inovação que o SHAP tem é representar os valores de Shapley, através de uma explicação do valor com um método de atribuição de características aditivas (*additive feature attribution*), como um modelo linear. Lundberg e Lee (2017) especifica o modelo de explicação pela Equação 2.8.

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \tag{2.8}$$

Onde $z' \in \{0,1\}^M$, M é o número de features de entrada simplificado, e $\phi \in \mathbb{R}$ é a

contribuição para cada feature j. Com isso podemos explicar a saída do modelo como uma soma de valores e a contribuição de cada feature de entrada. Além disso, Lundberg e Lee (2017) descrevem três propriedades desejáveis: precisão local, *missingness* e consistência.

Propriedade 1. (Precisão Local): ao aproximar o modelo original f para uma determinada entrada x, essa propriedade requer que o modelo de explicação g iguale a saída de f para entrada simplificada x', confome definido na Equação 2.9.

$$f(x) = g(x') = \phi_0 + \sum_{j=1}^{M} \phi_j x'_j \tag{2.9}$$

Nesse caso, o modelo $g(x')$ corresponde ao modelo original $f(x)$ quando $x = h_x(x')$, onde $\phi_0 = f(h_x(0))$ significa a saída do modelo sem as features.

Propriedade 2. (*Missingness*): requer que um elemento ausente recebe uma contribuição zero, conforme evidenciado na Equação 2.10.

$$x'_j = 0 \Rightarrow \phi_j = 0 \tag{2.10}$$

Propriedade 3. (Consistência): se um modelo muda, de modo que a contribuição marginal da feature aumenta ou permanece a mesma (independentemente das outras features), o valor Shapley também aumenta ou permanece o mesmo. Tendo a Equação 2.11 verdadeira, onde $f_x(z') = f(h_x(z'))$ para qualquer modelo f e f' e todas as entradas $z' \in \{0,1\}^M$, então garantimos a propriedade pela Equação 2.12. Onde, $z'\backslash i$ indica $z'_i = 0$.

$$f'_x(z') - f'_x(z'\backslash i) \geq f_x(z') - f_x(z'\backslash i) \tag{2.11}$$

$$\phi_i(f', x) \geq \phi_i(f, x) \tag{2.12}$$

Segundo García e Aznarte (2020), para calcular os valores SHAP, como uma medida unificada de importância das features, definimos $f_x(S) = f(h_x(z')) = E[f(x)|x_S]$, onde S é o conjunto de índices dos elementos não zero de z'. $E[f(z)|x_S]$ é o valor esperado da função condicionado ao subconjunto S das features de entrada. Os valores SHAP combinam essas expectativas condicionais com a teoria dos jogos e com os valores clássicos de Shapley para atribuir valores ϕ_i para cada feature. Segundo Lundberg e Lee (2017), o único modelo de explicação possível g que segue o método de atribuição de características aditivas (definido na Equação 2.8) e satisfaz as Propriedades 1, 2 e 3 é:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z'\backslash i)] \tag{2.13}$$

Onde $|z'|$ é o número de elementos não zero de z'. Este resultado implica que métodos não baseados nos valores de Shapley violam a precisão local (Propriedade 1) e a consistência (Propriedade 2). No entanto, esta definição de valor SHAP é projetada para se alinhar com Shapley regression values (LIPOVETSKY; CONKLIN, 2001), Shapley sampling values (ŠTRUMBELJ; KONONENKO, 2014) e Quantitative input influence (DATTA; SEN; ZICK, 2016). Além disso, permite conectar com LIME (RIBEIRO; SINGH; GUESTRIN, 2016), DeepLIFT (SHRIKUMAR; GREENSIDE; KUNDAJE, 2017) e Layer-wise Relevance Propagation (BACH et al., 2015).

SHAP unificou diversas abordagens para ser possível explicar os resultados de qualquer modelo de aprendizado profundo. Considerando a dificuldade de calcular o exato valor SHAP, existem formas de aproximar esses valores (LUNDBERG; LEE, 2017). A primeira delas é utilizar uma aproximação *model-agnostic*, chamada de Kernel SHAP. Onde o SHAP se une ao Linear LIME, que usa um modelo de explicação linear para aproximar a função f. As outras duas formas utilizam métodos de aproximação específica (Gradient SHAP e Deep SHAP), com isso conseguem rápidas aproximações. Deep SHAP utiliza DeepLIFT (SHRIKUMAR; GREENSIDE; KUNDAJE, 2017) para fazer a aproximação dos valores SHAP. Além disso, Lundberg, Erion e Lee (2018) apresentam o Tree SHAP, um algoritmo para calcular valores SHAP para modelos baseados em Árvores de Decisão (como o XGBoost).

2.6 Discussão

Partindo da ideia de reduzir o congestionamento, com os conceitos apresentados neste capítulo, nos capítulos seguintes apresentaremos duas abordagens de DRL para o controle semafórico adaptativo, utilizando o algoritmo Deep Q-learning. Para realizar a simulação dos algoritmos, nos baseamos na comparação entre os simuladores CityFlow e SUMO apresentada na Figura 3. O simulador CityFlow foi escolhido porque em alguns casos chegou a ter um desempenho 25 vezes maior.

Além disso, sobre a ótica da explicabilidade, escolhemos duas abordagens distintas. A primeira delas é o XGBoost que apesar de ser um algoritmo baseado em árvores de decisão (autoexplicativas em suas escolhas), precisa utilizar um modelo para explicar suas decisões (como o Tree SHAP). A segunda abordagem é DQN padrão (Q-learning com redes neurais), e utilizamos um modelo post-hoc de explicabilidade, baseada nos conceitos do framework SHAP utilizando o método de aproximação Deep SHAP.

3 TRABALHOS RELACIONADOS

Este capítulo apresenta os trabalhos relacionados ao controle semafórico adaptativo, do inglês *Adaptive Traffic Signal Control* (ATSC) , e trabalhos que abordam explicabilidade na área. De fato, diversos autores já realizaram trabalhos com a ideia de controlar semáforos de forma inteligente, mas poucos procuram aplicar métodos de explicabilidade em seus modelos. Por esse motivo, o capítulo também apresenta uma breve discussão sobre cada trabalho, considerando suas principais contribuições e limitações. A Seção 3.1 apresenta a metodologia aplicada para a escolha dos trabalhos relacionados. A Seção 3.2 apresenta os trabalhos com foco no ATSC. A Seção 3.3 apresenta os trabalhos que descrevem abordagens para explicabilidade no controle semafórico adaptativo. Por fim, a Seção 3.4 relata as considerações finais sobre o capítulo.

3.1 Metodologia e Critérios de Seleção e Parada

A metodologia de pesquisa utilizada consiste em verificar na literatura artigos com objetivos similares ao deste trabalho que se destacam dos demais pelas suas propostas e contribuições. Para realizar essa pesquisa, foi utilizada principalmente a técnica *Snowballing* (WOHLIN, 2014). Essa a técnica consiste em identificar novos estudos relevantes nas listas de referências e nas listas de citações dos estudos já conhecidos. A técnica é simples e intuitiva. Segundo Choong et al. (2014), apesar do método ser demorado, a eficácia da técnica de *snowballing* faz com que a mesma seja a melhor prática para revisões sistemáticas.

Segundo Badampudi, Wohlin e Petersen (2015), a técnica mostra uma eficiência superior à de outros métodos em diversos casos estudados pelos autores. Contudo, os resultados dependem da escolha do conjunto inicial adequado, pois caso o conjunto não seja bem construído, podem ser perdidos resultados relevantes que são encontrados pelos métodos comparados. Ainda assim, Greenhalgh e Peacock (2005) apresentam um estudo onde 51% das referências encontradas em uma revisão sistemática são identificadas por *snowballing*.

Além disso, Wohlin (2014) define os termos *snowballing backward* (que significa coletar e analisar as referências citadas pelos trabalhos) e *snowballing forward* (coletar e analisar os estudos que citam os trabalhos). *Citation Track* e *Pursuing References List* são sinônimos de *snowballing*, mas normalmente representam apenas o *snowballing backward* (SILVA, 2017).

3.1.1 Metodologia

A primeira etapa da técnica *snowballing* é definir o conjunto inicial (conjunto *seed*), esse é o conjunto de estudos fornecidos como entrada para a execução do *snowballing*. A partir dele, novos artigos vão ser encontrados e então selecionados. Seguindo as diretrizes propostas por Wohlin (2014), o conjunto inicial deve ser diversificado e suficientemente grande, para ser possível abranger estudos independente de autor, ano, conferências, revistas e editoras.

Para definir o conjunto inicial, se estabeleceu uma pesquisa sobre trabalhos referência na área, como *surveys* e *reviews*. Esses trabalhos foram chamados de conjunto base, do qual vão ser extraídos o conjunto inicial para realizar a técnica *snowballing*. Seguindo diretrizes propostas por Wohlin (WOHLIN, 2014) para definir esse conjunto base, pesquisamos *"intelligent traffic signal control"* no Google Scholar, e selecionamos trabalhos com base nos seguintes critérios de inclusão: (a) trabalho deve ser um survey ou um review; (b) deve ser escrito em inglês; (c) deve ter no mínimo uma citação. Optamos por interromper a busca após encontrar 10 trabalhos distintos.

O conjunto base foi separado em dois diferentes grupos, baseado em seus respectivos focos: (i) o primeiro grupo, tem inicialmente o foco em controle semafórico adaptativo (ATSC); (ii) o segundo grupo tem foco em formas de explicabilidade na área de ATSC. Na Tabela 1 pode-se observar os artigos de ambos os grupos. Além disso, na tabela também é possível identificar o veículo, a editora, o ano e o número de citações[1] que os artigos têm.

Tabela 1: Conjunto base de trabalhos selecionados para a técnica de snowballing. Grupo (i) indica ter foco apenas no ATSC, grupo (ii) foca em explicabilidade. Os artigos foram ordenados cronologicamente.

Id	Grupo	Referência	Veículo	Editora	Ano	Citações[1]
1	(i)	(PAPAGEORGIOU et al., 2003)	Proceedings of the IEEE	IEEE	2003	1342
2	(i)	(BAZZAN, 2009)	Autonomous Agents and Multi-Agent Systems	Springer	2009	258
3	(i)	(BAZZAN; KLÜGL, 2014)	The Knowledge Engineering Review	Cambridge University Press	2014	291
4	(i)	(MANNION; DUGGAN; HOWLEY, 2016)	Autonomic road transport support systems	Springer	2016	214
5	(i)	(YAU et al., 2017)	ACM Computing Surveys (CSUR)	ACM New York, NY, USA	2017	117
6	(i)	(WEI et al., 2019a)	ArXiv	-	2019	85
7	(i)	(EOM; KIM, 2020)	European transport research review	Springer	2020	3
8	(i)	(WEI et al., 2021)	ACM SIGKDD Explorations Newsletter	ACM New York, NY, USA	2021	9
9	(ii)	(RASHEED et al., 2020)	IEEE Access	IEEE	2020	8
10	(ii)	(OMEIZA et al., 2021)	ArXiv	-	2021	4

Fonte: Elaborado pelo autor.

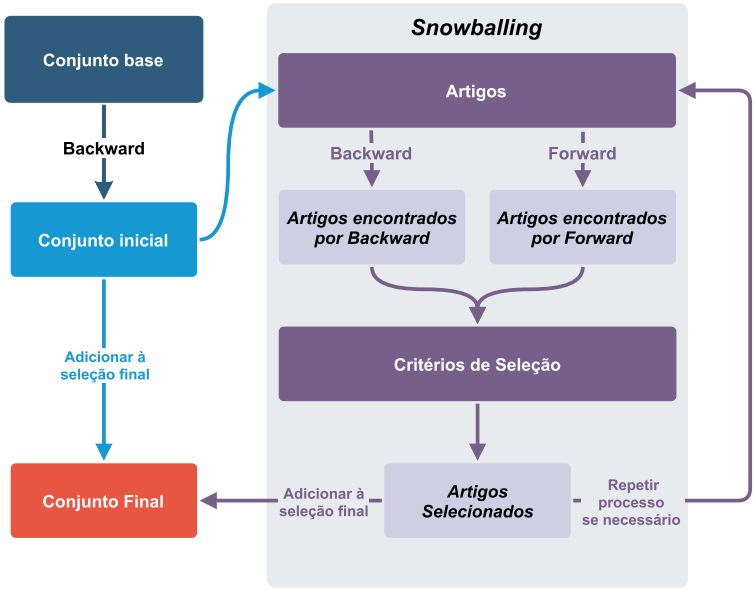
A Figura 8 apresenta o fluxo utilizado para selecionar os artigos. O conjunto inicial de artigos foi selecionado realizando o *snowballing backward* dos artigos do conjunto base. Em seguida, a partir deste conjunto inicial, foram aplicadas as técnicas *snowballing backward* e *snowballing forward* para extrair uma nova seleção de artigos. Esses artigos passam por um processo de seleção que se baseia em critérios de inovação e contribuição científica. Ao fim desse processo, os artigos selecionados são adicionados ao conjunto final, e enquanto for necessário, o processo é repetido para esses novos artigos selecionados.

Um dos principais desafios para o processo de *snowballing* é definir o critério de parada. Segundo Silva (2017), a tarefa de definir quando o conjunto de estudos encontrado é suficiente para a pesquisa pode ser complexa e depende do tema de pesquisa escolhido. Para o presente estudo, o critério de parada foi a inovação e o resultado dos artigos. Ao chegar próximo de 30 artigos selecionados, os novos trabalhos apresentavam poucas mudanças ou resultados inferiores aos já encontrados. Além disso, uma versalidade da técnica *snowballing* é possibilitar atualizar

[1] O número de citações foi retirado do mecanismo de busca Google Scholar no dia 2 de dezembro de 2021.

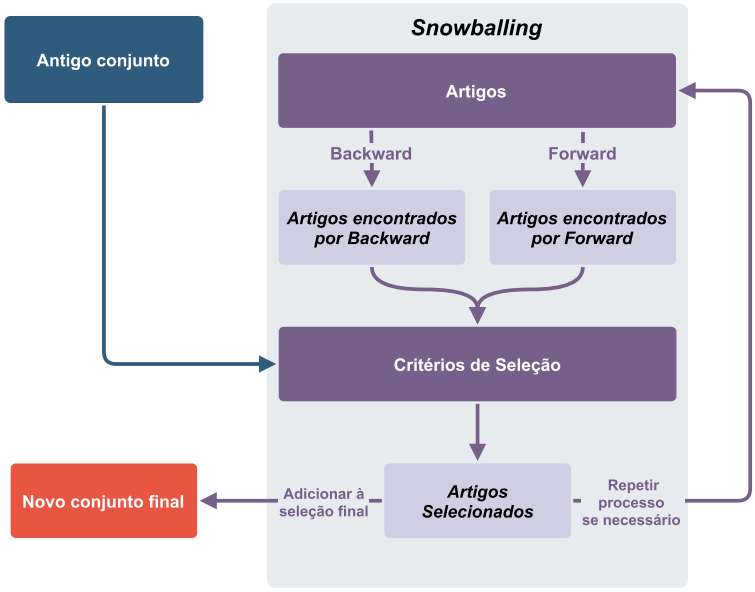
o conjunto final de artigos facilmente, como apresentado no trabalho de Felizardo et al. (2016). Na Figura 9, o fluxo de atualização dos artigos por meio da técnica de *snowballing* é apresentado. Neste fluxo o antigo conjunto de dados passa primeiro por uma seleção para confirmar se os artigos ainda se encaixam nos critérios definidos.

Figura 8: Fluxo de seleção de artigos pela técnica de *snowballing*.



Fonte: Elaborada pelo autor.

Figura 9: Fluxo de atualização de artigos pela técnica de *snowballing*.



Fonte: Elaborada pelo autor.

3.1.2 Discussão

Utilizando a técnica *snowballing*, foram selecionados 30 trabalhos relevantes e próximos aos objetivos a serem atingidos por essa proposta. As próximas seções apresentam uma breve discussão sobre cada trabalho, considerando suas principais contribuições e limitações. Assim como a divisão de artigos para o conjunto base, as discussões apresentadas nas próximas seções são divididas entre trabalhos com foco apenas no controle semafórico adaptativo (ATSC) e trabalhos de explicabilidade para esse controle semafórico como explicado no início do capítulo.

3.2 Controle Semafórico Adaptativo

A ideia de controlar os sinais de tráfego para reduzir o congestionamento tem uma longa história. Alguns dos primeiros passos foram baseados em modelos matemáticos complexos, como o sistema de trânsito adaptativo coordenado de Sydney, do inglês *Sydney Coordinated Adaptive Traffic System* (SCATS) (SIMS; DOBINSON, 1980) e a técnica de otimização do deslocamento do ciclo dividido, do inglês *Split Cycle Offset Optimization Technique* (SCOOT). No entanto, esses trabalhos não conseguiram lidar com a adaptabilidade e flexibilidade em tempo real. Abordagens posteriores conseguiram lidar com condições de tráfego em tempo real, tais como o *Real-Time Hierarchical Optimized Distributed Effective System* (RHODES) (SEN; HEAD, 1997; MIRCHANDANI; HEAD, 2001). No entanto, segundo Dujardin, Vanderpooten e Boillot (2015), o RHODES é melhor classificado como um sistema semi-adaptativo, visto que não pode se adaptar rapidamente às condições de tráfego.

De modo a lidar com um grande número de veículos, Wunderlich et al. (2008) introduziram o algoritmo *Longest Queue First* (LQF), que busca reduzir o tamanho das filas. No entanto, o LQF enfrenta dificuldades ao lidar com diferentes fluxos de veículos e se torna injusto ao lidar simultaneamente com filas pequenas e grandes, em que geralmente favorece o último caso (WU et al., 2017). Cools, Gershenson e D'Hooghe (2013) apresentaram o *Self-Organizing Traffic Lights Control* (SOTL), que regula os semáforos ao estabelecer um limite pré-definido para o número de veículos em espera, sendo então utilizado para alterar as fases. Vários trabalhos na literatura utilizam o SOTL como base de comparação para seus modelos na área de controle semafórico adaptativo (WEI et al., 2018; XIONG et al., 2019; ZHENG et al., 2019). Varaiya (2013) apresenta o atual estado da arte na área para o controle semafórico, o *MaxPressure*. Esse que escolhe gananciosamente a fase com a maior pressão (número de carros parados/entrando do cruzamento - número de carros saindo do cruzamento). Todavia, tal abordagem simplifica condição de tráfego e não garante resultados ótimos no mundo real (CHEN et al., 2020).

Abordagens posteriores utilizam cenários mais realistas e começaram a considerar agentes de tomada de decisão mais robustos, capazes de se adaptar prontamente à dinâmica do tráfego. Nesta seção, seguimos essa direção e discutimos trabalhos representativos na área de aprendizado por reforço para o controle semafórico adaptativo. Para uma revisão mais abrangente da

literatura sobre estes tópicos, sugerem-se os trabalhos de Papageorgiou et al. (2003), Bazzan (2009), Yau et al. (2017), e Wei et al. (2019a, 2021).

O número de trabalhos que utilizam aprendizado por reforço (RL) no controle de tráfego tem crescido constantemente nas últimas décadas. Thorpe e Anderson (1996) utilizaram o algoritmo SARSA e testaram diferentes representações de estado para otimizar o tráfego, mostrando melhorias significativas em relação aos planos de sinais fixos. Abdulhai, Pringle e Karakoulas (2003) introduziram o Q-learning com espaço de estados discreto para encontrar uma política de controle ótima para situações de tráfego com congestionamento em uma rede rodoviária com dois acessos. Bingham (2001) utilizou um controle semafórico neuro-fuzzy, onde aplicou aprendizado por reforço (RL) para o treinamento da rede neural. Seu trabalho também foi testado em uma interseção com dois acessos simples. Embora tenham obtidos resultados promissores, estas abordagens utilizaram aprendizado por reforço tabular, que geralmente se mostra ineficaz para lidar com um grande espaço de ações e estados (SUTTON; BARTO, 2018).

O aprendizado por reforço profundo, do inglês *Deep Reinforcement Learning* (DRL), tem sido muito utilizado para lidar com cenários maiores e mais realistas. Wei et al. (2018) desenvolveram um controlador de sinais de tráfego baseado no DQN executado no simulador SUMO. Sua abordagem foi treinada usando *features* numéricas e imagens representando o estado do cruzamento. Zheng et al. (2019) desenvolveram um modelo chamado FRAP (baseado no Ape-X DQN (HORGAN et al., 2018)) que utiliza características numéricas de vários movimentos distintos do tráfego.

Outras abordagens baseadas em aprendizado por reforço profundo na literatura também propuseram o uso de diversas técnicas diferentes para o ATSC. Gu et al. (2020) implementaram um agente utilizando Double DQN. Nishi et al. (2018) aplicaram graph convolutional neural networks no simulador SUMO. Mousavi, Schukat e Howley (2017) desenvolveram duas abordagens distintas, deep policy-gradient e value-function-based RL, também simulados usando o SUMO. Casas (2017) aplicou deep deterministic policy gradient. Ma e Wu (2020) utilizaram o algoritmo de aprendizado por reforço profundo Feudal Multi-Agent. Aslani, Mesgari e Wiering (2017) usaram um método de *actor-critic* com uma rede de tráfego real com diversos fluxos de tráfego no simulador Aimsun. Apesar das diferentes abordagens, nenhuma delas apresenta uma explicabilidade do modelo ou das políticas aprendidas. Além disso, nenhum dos trabalhos citados acima consegue superar o melhor resultado do estado da arte até o momento, conforme apresentado a seguir.

Chen et al. (2020) apresenta o modelo chamado MPLight, que se baseia em uma coordenação implícita apresentada por Wei et al. (2019b). Além disso, unifica a abordagem com o modelo FRAP elaborado por Zheng et al. (2019). Atualmente, o MPLight pode ser considerado o estado da arte para controle semafórico adaptativo. Chen et al. (2020) destacam três fatores fundamentais para a utilização do modelo no mundo real: escalabilidade, coordenação e viabilidade dos dados. O MPLight é o único modelo que conseguiu controlar com eficiência mais de 2800 semáforos simultaneamente. O modelo também apresenta resultados significativos so-

bre o compartilhamento de tuplas de experiências entre as interseções. Todavia, o modelo não utiliza um sistema de cooperação. Além disso, segue a mesma linha da maioria das abordagens descritas acima e não apresenta nenhuma explicabilidade do modelo.

3.3 Explicabilidade no Controle Semafórico Adaptativo

Pesquisas recentes se concentraram em promover a explicabilidade das políticas aprendidas e em obter resultados significativos no controle semafórico adaptativo (ATSC).

Wei et al. (2018) investigaram a semelhança entre as políticas aprendidas e os padrões de tráfego de uma interseção. Uma direção semelhante foi seguida por Zheng et al. (2019). Comparando as políticas aprendidas e os padrões de tráfego, estes trabalhos visavam mostrar que o agente consegue ajustar adequadamente o plano semafórico às condições de tráfego. Apesar dos resultados interessantes, estas abordagens não permitem que as políticas sejam explicadas globalmente ou localmente, o que poderia ser útil para entender por que determinada ação deve ser escolhida em cada estado. Este é justamente um dos objetivos do presente trabalho.

Wollenstein-Betech et al. (2020) apresentaram uma abordagem de compilação de conhecimento para explicar as probabilidades de ação em função das características do estado. Essa abordagem constrói uma representação hierárquica de percepções interpretáveis, voltado para desenvolver um modelo com explicabilidade implícita. Ault, Hanna e Sharon (2020) propuseram um algoritmo baseado em DQN com uma função de controle regulável. Tal função é utilizada para otimizar os parâmetros da política, que podem então ser interpretados. Entretanto, sua abordagem tem se mostrado mais lenta e menos eficiente do que o DQN. Ao contrário dos trabalhos descritos acima, a ideia do nosso trabalho é utilizar um método post-hoc, permitindo primeiro otimizar a política usando DQN (obtendo assim bom desempenho) e depois usar meios de investigação para explicar a política aprendida.

Recentemente, como um esforço para melhorar a explicabilidade da política de algoritmos, pesquisas têm sido feita usando SHapley Additive exPlanations (SHAP) (LUNDBERG; LEE, 2017). No caso do ATSC uma das primeiras pesquisas a utilizar SHAP foi Rizzo, Vantini e Chawla (2019), cujo foco foi explicar as decisões tomadas por um agente DRL. Como resultado preliminar, esse trabalho conseguiu mostrar que o agente reage de forma diferente conforme o tráfego de cada pista. Também demonstrou consistência na lógica do agente ao analisar sensores na pista. Um ponto interessante foi que os autores conseguiram identificar o viés no tráfego e notaram *overfitting* no modelo. Além disso, Rizzo e colegas afirmaram que a relação entre alguns detectores e a fase escolhida não podia ser explicada de forma intuitiva. Todavia, o trabalho utiliza poucas abordagens de comparação com seu modelo. Além disso, segundo Omeiza et al. (2021) e Wang et al. (2019), o trabalho foi classificado como *unvalidated guidelines*, visto que parte das explicações e avaliações não possuem justificativas substanciais e são fornecidas com base na experiência do autor sobre o assunto.

3.4 Discussão

A maioria das abordagens de aprendizado por reforço mencionadas na Seção 3.2 se concentram principalmente na avaliação do desempenho de um modelo treinado. Entretanto, segundo Gunning (2017) e Wollenstein-Betech et al. (2020), as abordagens baseadas no aprendizado profundo, conhecidas por sua falta de explicabilidade, representam uma grande preocupação em cenários do mundo real, visto que as políticas aprendidas não podem ser compreendidas ou reguladas pelo ser humano, limitando assim o uso desses métodos.

Embora Rizzo, Vantini e Chawla (2019), Wollenstein-Betech et al. (2020) e Ault, Hanna e Sharon (2020) tenham feito progressos interessantes, até onde sabemos, nenhum dos trabalhos citados nesse capítulo apresenta uma comparação sobre a ótica de desempenho e explicabilidade para o contexto do controle semafórico inteligente com aprendizado por reforço profundo. Além disso, nenhuma pesquisa utilizou um algoritmo modelado de forma que o desempenho seja similar ao dos melhores algoritmos da área e tenha explicações simples e intuitivas.

Com isso, nesse trabalho seguimos a direção de comparar dois algoritmos de aprendizado por reforço profundo, capazes de otimizar o congestionamento e serem explicáveis. O primeiro deles apresenta explicações post-hoc por meio do TreeSHAP e o outro apresenta explicações post-hoc por meio do DeepSHAP. Além disso, o atual trabalho consegue, através de uma representação unificada de todas as ações de um determinado estado, apresentar uma relação dos valores SHAP com a política aprendida sem a necessidade de diversas figuras como no trabalho do Rizzo, Vantini e Chawla (2019). Mais informações sobre essa representação unificada é dada no Capítulo 6.

4 MÉTODO

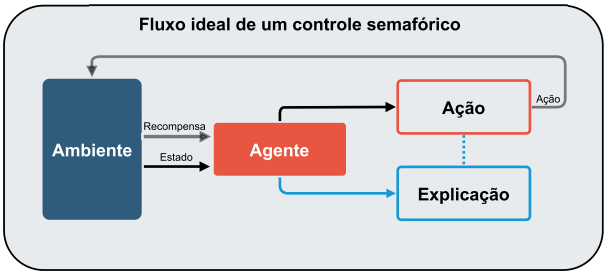
Este capítulo apresenta os modelos de aprendizado por reforço profundo (DRL) idealizados para alcançar o objetivo de controlar com eficiência múltiplos cruzamentos e permitir que os agentes tenham políticas explicáveis. A Seção 4.1 fornece uma visão geral sobre o método proposto e seus principais conceitos. Na Seção 4.2 é apresentada a definição do cenário, do cruzamento e dos agentes. A Seção 4.3 indroduz as modelagens do MDP para esse problema. A Seção 4.4 apresenta a definição dos agentes, algoritmos propostos e seus parâmetros. Na Seção 4.5 uma breve explicação sobre os modelos de explicabilidade é feita. Por fim, na Seção 4.6 temos uma breve discussão com as considerações finais do capítulo.

Vale destacar que um estudo preliminar foi desenvolvido e publicado durante este trabalho (SCHREIBER; RAMOS; BAZZAN, 2021). O mesmo contou com um método similar ao apresentado neste capítulo, embora um pouco mais simples. Em função da limitação de espaço, tais resultados são apresentados de forma independente no Apêndice A.

4.1 Visão Geral

Para realizar o controle semafórico inteligente explicável com aprendizado por reforço (RL), foram definidas dois algoritmos diferentes de aproximadores de função para o Deep Q-Learning. O primeiro deles utiliza o XGBoost, que a partir desse momento será chamado apenas por **XGB**. O segundo utiliza uma rede neural, similar ao artigo original (MNIH et al., 2015), que já é conhecido como **DQN**. Com isso, precisamos estabelecer algumas definições sobre o ambiente, o agente e a explicabilidade para cada algoritmo. Partimos com um exemplo de um possível fluxo ideal para o controle semafórico explicável na Figura 10, onde um agente recebe informações do ambiente por meio dos estados e interage com esse ambiente por meio uma determinada ação e em paralelo permite explicações dessa ação.

Figura 10: Fluxo com visão inicial de uso do agente explicável.

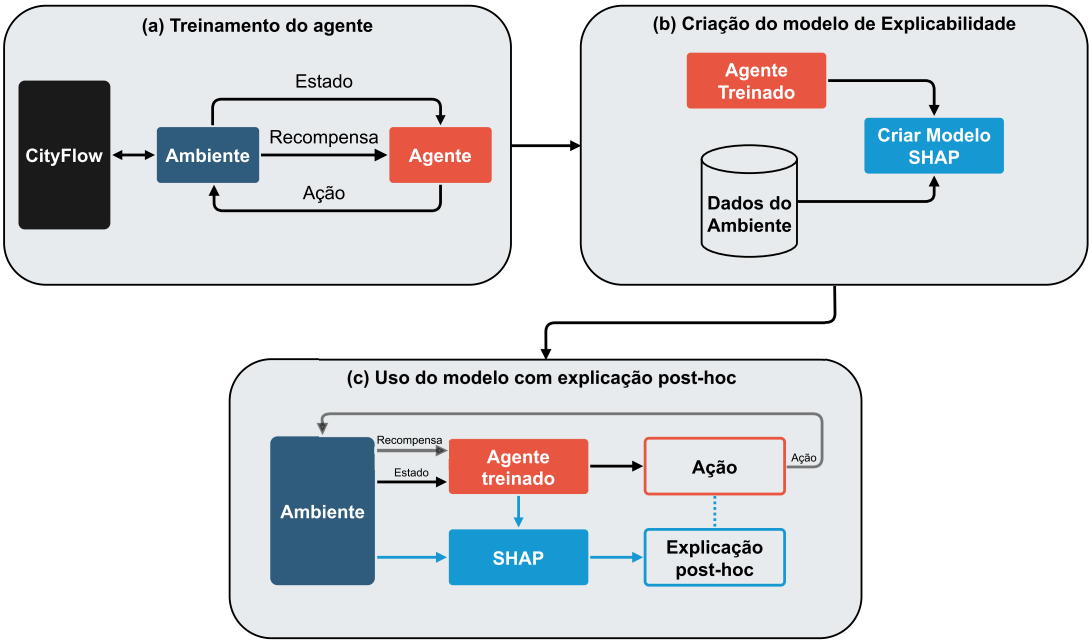


Fonte: Elaborado pelo autor.

Apesar do fluxo estar correto, é evidente que não podemos testar qualquer hipótese ou modelo na prática, visto que há a eventualidade de um possível impacto negativo na vida das pessoas envolvidas, como explicado nas Seções 2.1.1 e 2.1.4. Tendo isso em vista, devemos

adicionar ao fluxo uma etapa de treinamento do agente, onde o mesmo vai interagir com um ambiente simulado e vai extrair informações relevantes para controlar os semáforos. Além disso, o agente estabelecido para esse fluxo deve utilizar um algoritmo transparente, ou um algoritmo que permita explicações intrínsecas, como descrito na Seção 2.5. Nenhum dos dois algoritmos propostos (XGB e DQN) possuem explicações intrínsecas. Além do mais, para ter explicações post-hoc, precisaríamos gerar um modelo SHAP dos agentes. Por isso, devemos adicionar ao fluxo uma etapa de criação de modelo de explicabilidade. A Figura 11 apresenta, de forma geral, uma visão sobre esse novo fluxo descrito. Esse fluxo é dividido em três etapas: (a) treinamento do agente; (b) criação do modelo de explicabilidade; (c) uso do modelo com explicação post-hoc.

Figura 11: Visão geral de uso para algoritmos com explicação post-hoc.



Fonte: Elaborado pelo autor.

Na Etapa (a), como pode-se observar na Figura 11, para realizar o treinamento do modelo consideramos uma instância do simulador CityFlow, o qual é modelado como um ambiente possuindo informações sobre a estrutura da rede viária e o fluxo dos carros. O agente, por sua vez, precisa ter definições claras sobre seus respectivos objetivos, além de uma formulação de estado e ação, permitindo ao mesmo interagir com o ambiente e extrair informações relevantes para controlar o semáforo. Inicialmente, o agente modelado não conhece a dinâmica do ambiente e como suas ações afetam o mesmo. Com o passar do treinamento, informações relevantes são armazenadas e permitem que o agente consiga estipular políticas para as suas ações que maximizam a recompensa recebida.

Na Etapa (b), o agente já treinado e os dados do ambiente são usados para criação de um modelo de explicabilidade. O objetivo dessa etapa é permitir a explicabilidade post-hoc com as informações do ambiente (como os estados) e o comportamento do agente para com essas

informações.

Na Etapa (c) é apresentado o uso do modelo com explicabilidade junto do agente treinado. A interação do agente com o ambiente é similar à do treinamento. Entretanto, a diferença é que enquanto o agente interage com o ambiente, o modelo de explicabilidade assume a tarefa de apresentar explicações do estado atual com as possíveis ações do agente para o estado atual.

Por fim, podemos então confirmar que o fluxo final, apresentado na Figura 11, pode ser utilizado para os algoritmos XGB e DQN. Também vale ressaltar que apesar de usarmos um ambiente simulado para a Etapa (c), não existe mais a necessidade do ambiente ser simulado, dado que nesta etapa o agente já poderia estar em contato com um ambiente real (obviamente deve-se considerar algumas restrições do trânsito não abordadas no atual trabalho).

4.2 Definição do Cenário

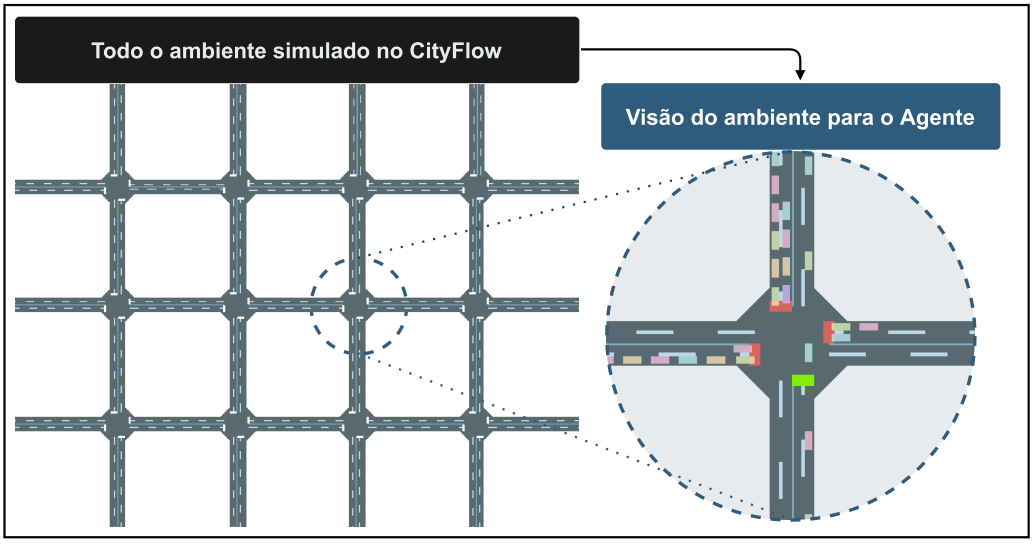
O ambiente de simulação utilizado consiste em múltiplos cruzamentos em formato de rede (*grid*) com 4 acessos cada. Na Figura 12 pode-se observar um exemplo de uma simulação com 12 cruzamentos (no formato 4x3). Além disso, na figura também é possível observar que, independente do número de cruzamentos e agentes dispostos na simulação, cada cruzamento é tratado de forma individual.

Uma vez iniciada a simulação com o simulador CityFlow (ZHANG et al., 2019), cada veículo se desloca de uma origem até um destino, conforme pré-estabelecido no arquivo de fluxo de veículos. Com relação aos semáforos, existe um parâmetro que decide se os semáforos seguem um tempo fixo ou se são controlados por um controlador. Quando controlados, em cada etapa da simulação o simulador fornece o estado do ambiente e executa a ação decidida pelo controlador.

Em cada cruzamento os veículos podem seguir em linha reta, virar à esquerda ou à direita. Ao contrário de outros movimentos, é permitido virar à direita independentemente da fase (mesmo o sinal estando vermelho). Neste sentido, foi definido que o movimento para a direita pode ser ignorado da modelagem do ambiente simulado. Um único agente é responsável pelo controle do tempo de todas as fases de uma interseção. Informações sobre cada simulação, infraestrutura e fluxo de veículos são apresentadas na Seção 5.1.

Os agentes de aprendizado por reforço deste trabalho podem ser visto como controladores isolados com modelagens individuais, Claus e Boutilier (1998) se referem a tais agentes como *independent learners*. Apesar das ações dos agentes afetarem a rede como um todo, assumimos que esse efeito faz parte da dinâmica do ambiente. A ordem das fases e o tempo de duração entre cada fase são definidos conforme a modelagem do ambiente (modelagens descritas na Seção 4.3). As 8 fases e seus movimentos correspondentes são apresentados na Tabela 2.

Figura 12: Exemplo de rede viária simulada no CityFlow. O agente controlador do cruzamento destacado tem a visão apenas desse cruzamento e não do cenário todo.



Fonte: Elaborada pelo autor.

Tabela 2: Fases e movimentos correspondentes. Na tabela, o norte, sul, leste e oeste, são representados pelas letras 'N','S','L' e 'O' respectivamente.

Fase	Movimentos permitidos	Direções permitidas
Fase 0	$O \rightarrow L$ e $L \rightarrow O$	$\rightarrow$ $\leftarrow$
Fase 1	$S \rightarrow N$ e $N \rightarrow S$	$\downarrow$ $\uparrow$
Fase 2	$O \rightarrow N$ e $L \rightarrow S$	$\rightarrow\!\!\uparrow$ $\leftarrow\!\!\uparrow$
Fase 3	$S \rightarrow O$ e $N \rightarrow L$	$\uparrow\!\!\leftarrow$ $\uparrow\!\!\rightarrow$
Fase 4	$O \rightarrow L$ e $O \rightarrow N$	$\rightarrow$ $\rightarrow\!\!\uparrow$
Fase 5	$L \rightarrow O$ e $L \rightarrow S$	$\leftarrow$ $\leftarrow\!\!\uparrow$
Fase 6	$S \rightarrow N$ e $S \rightarrow O$	$\uparrow$ $\uparrow\!\!\leftarrow$
Fase 7	$N \rightarrow L$ e $N \rightarrow S$	$\uparrow\!\!\rightarrow$ $\downarrow$

Fonte: Elaborada pelo autor.

4.3 Modelagem do MDP

A fim de desenvolver um controlador de sinais de tráfego baseado em aprendizado por reforço (RL), caracterizamos o problema como um processo de decisão de Markov (MDP). Em cada etapa, o agente escolhe uma ação baseada no estado em que se encontra. Ao executar essa ação, o agente recebe uma recompensa e avança para o próximo estado. Durante o desenvolvimento deste trabalho, foram escolhidas duas diferentes modelagens para o ambiente. Suas respectivas ações, estados e recompensas são brevemente discutidos nas subseções a seguir.

4.3.1 Modelagem 1: MDP Cíclico

Para a modelagem do MDP Cíclico, o agente de aprendizado por reforço é considerado um controlador isolado de fase fixa com 8 movimentos. Isto significa que a ordem das fases é definida previamente e não muda. As fases seguem a ordem descrita na Tabela 2, iniciando na Fase 0 (com os movimentos $\rightarrow \leftarrow$), depois indo para a Fase 1 (com os movimentos $\downarrow \uparrow$), e assim por diante. Com isso, o agente é responsável apenas por definir a duração da fase atual.

- **Ação**: as ações representam a duração do tempo em que a fase atual ficará em verde. Com isso, podemos definir que as ações possíveis são números inteiros que representam o tempo da fase. Decidimos que cada fase tem um tempo máximo de 60 segundos e mínimo de 10 segundos, com granularidade de 2 segundos, ou seja, o agente pode escolher o tempo de 10, 12, 14 até 60 segundos. Além disso é permitido pular[1] a fase seguinte.

- **Estado**: os estados são modelados como uma tupla com três posições, cada qual representando uma característica (*feature*) diferente do ambiente. A tupla é representada por $S_t = \langle F0,\ F1,\ F2 \rangle$, onde F0, F1 e F2 representam o comprimento da fila: (i) na fase atual; (ii) na fase seguinte; (iii) todas as demais fases[2], respectivamente. A definição de fase atual é relativa ao momento de observação. No momento em que o agente tomará a ação, a fase atual é referente a fase que terá seu tempo definido. Ao fim da duração definida para a fase atual, o agente precisa selecionar outra ação. Dessa vez considerando a nova fase atual (antiga fase seguinte). Esse processo é repetido para cada fase subsequente.

- **Recompensa**: a recompensa é definida como a pressão do cruzamento. A pressão pode ser adquirida pelo número de carros entrando no cruzamento menos o número de carros saindo do cruzamento. Além disso, a recompensa é normalizada no intervalo de $[-1, 1]$, de modo a melhorar o processo de aprendizagem.

4.3.2 Modelagem 2: MDP Seletor

Para a modelagem do MDP Seletor, o agente de aprendizado por reforço é considerado um controlador isolado de tempo fixo com 8 possíveis fases. Isto significa que a ordem das fases é definida pelo agente, mas o tempo é previamente determinado e não muda. O tempo de cada fase foi escolhido inicialmente como 20 segundos[3]. Com isso, o agente é apenas responsável por definir qual fase permanece aberta.

[1] A fase é automaticamente pulada se não houver carros esperando, esse pulo é realizado por um sistema baseado em regras antes da ação do agente.

[2] As demais fases em que os movimentos não estão representados pelas duas primeiras.

[3] Realizamos alguns testes que permitiram determinar empiricamente esse valor.

- **Ação**: as ações representam qual a fase que ficará em verde. Com isso, podemos definir que as ações possíveis são números inteiros que representam cada uma das 8 possíveis fases. Decidimos que a fase selecionada fica 20 segundos em verde. A mesma fase pode ser selecionada quantas vezes o agente decidir, entretanto, essa escolha é feita a cada 20 segundos.

- **Estado**: os estados são modelados como uma tupla com oito posições, cada qual representando o tamanho de um movimento específico. A tupla é representada por $S_t = \langle F0,\ F1,\ F3,\ F4,\ F5,\ F6,\ F7 \rangle$, onde F0 até F7 representam o comprimento da fila do movimento 1 até o movimento 8, conforme a Tabela 3.

- **Recompensa**: a recompensa, assim como o MDP Cíclico, é definida como a pressão do cruzamento. A pressão pode ser adquirida pelo número de carros entrando no cruzamento menos o número de carros saindo do cruzamento. Além disso, a recompensa é normalizada no intervalo de $[-1, 1]$, de modo a melhorar o processo de aprendizagem.

Tabela 3: Número e símbolo dos movimentos. Na tabela, o norte, sul, leste e oeste, são representados pelas letras 'N','S','L' e 'O' respectivamente.

Nome	Movimento	Símbolo/Direção
Movimento 0	$O \rightarrow L$	$\rightarrow$
Movimento 1	$L \rightarrow O$	$\leftarrow$
Movimento 2	$S \rightarrow N$	$\uparrow$
Movimento 3	$N \rightarrow S$	$\downarrow$
Movimento 4	$O \rightarrow N$	$\nearrow$
Movimento 5	$L \rightarrow S$	$\swarrow$
Movimento 6	$S \rightarrow O$	$\curvearrowleft$
Movimento 7	$N \rightarrow L$	$\curvearrowright$

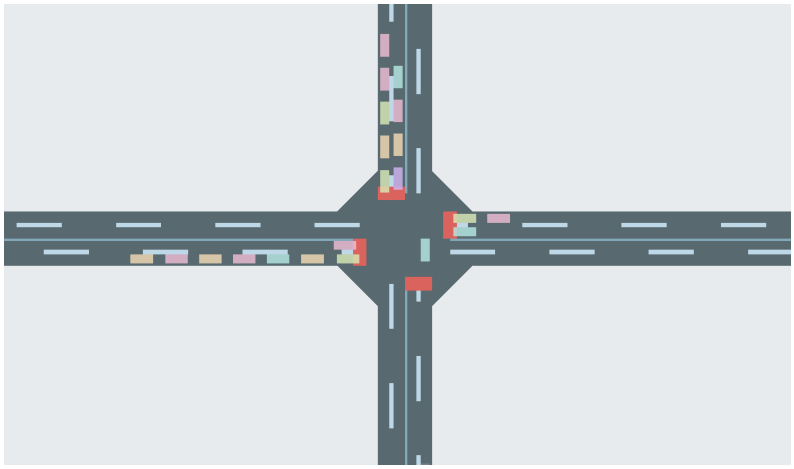
Fonte: Elaborada pelo autor.

4.3.3 Exemplo

Considere o seguinte exemplo para facilitar o entendimento da modelagem dos MDPs. No tempo t o agente deve escolher uma ação. A Figura 13 e a Tabela 4 descrevem as informações do ambiente no momento t.

Considerado a modelagem do **MDP Cíclico** descrita na Seção 4.3.1, as informações descritas na Tabela 2, as informações da Tabela 4, e sabendo que o agente precisa definir a ação para a Fase 0. Podemos montar a tupla de features da seguinte forma: (i) F0 é igual ao tamanho da fila da Fase 0; (ii) F1 é igual ao tamanho da fila da Fase 1; (iii) F2 é igual ao tamanho da fila da

Figura 13: Rede viária com exemplo.



Fonte: Extraida do CityFlow pelo autor.

Tabela 4: Tabela que resume a rede viária com exemplo.

Movimento	Direção	Tamanho da fila
$O \rightarrow L$	$\rightarrow$	7
$L \rightarrow O$	$\leftarrow$	2
$S \rightarrow N$	$\uparrow$	0
$N \rightarrow S$	$\downarrow$	5
$O \rightarrow N$	$\rightharpoonup$	1
$L \rightarrow S$	$\rightharpoondown$	1
$S \rightarrow O$	$\upharpoonleft$	0
$N \rightarrow L$	$\downharpoonright$	4

Fonte: Elaborada pelo autor.

Fase 2 e 3. Além disso, podemos identificar o estado atual da seguinte forma[4]:

$$S_t = \langle F0;\ F1;\ F2 \rangle = \langle \underbrace{7+2}_{\rightarrow + \leftarrow};\ \underbrace{0+5}_{\uparrow + \downarrow};\ \underbrace{1+1+0+4}_{\rightharpoonup + \rightharpoondown + \upharpoonleft + \downharpoonright} \rangle = \langle 9;\ 5;\ 6 \rangle \tag{4.1}$$

Onde S_t indica o estado no tempo t. Podemos dizer que o estado atual é composto pelas três features e seus valores são 9, 5 e 6 respectivamente. A partir desse momento, o agente escolherá uma ação baseada nesses valores. Depois, o agente vai esperar o tempo definido acabar para então receber o próximo estado e escolher o tempo da fase seguinte.

Considerado a modelagem do **MDP Seletor** descrita na seção anterior, as informações descritas na Tabela 2 e as informações da Tabela 4, podemos montar a tupla de features da seguinte forma[5]:

[4]Nessa equação, para facilitar o entendimento das operações envolvidas no cálculo das features, fazemos levemente algumas mudanças na notação e substituímos a ‘,’ pelo ‘;’. Mas isso não afeta os conceitos e nem os resultados, apenas a apresentação.

[5]Assim como na Equação 4.1, para facilitar o entendimento das operações envolvidas no cálculo das features, substituímos a ‘,’ pelo ‘;’.

$$S_t = \langle F0; F1; F3; F4; F5; F6; F7 \rangle = \langle \underbrace{7}_{\rightarrow}; \underbrace{2}_{\leftarrow}; \underbrace{0}_{\uparrow}; \underbrace{5}_{\downarrow}; \underbrace{1}_{\nearrow}; \underbrace{1}_{\nwarrow}; \underbrace{0}_{\upharpoonleft}; \underbrace{4}_{\downharpoonright} \rangle \quad (4.2)$$

Onde S_t indica o estado no tempo t. Podemos dizer que o estado atual é composto pelas oito features. A partir desse momento, o agente escolherá uma fase baseada nesses valores. Depois, vai esperar 20 segundos e então receber o próximo estado e escolher a próxima fase.

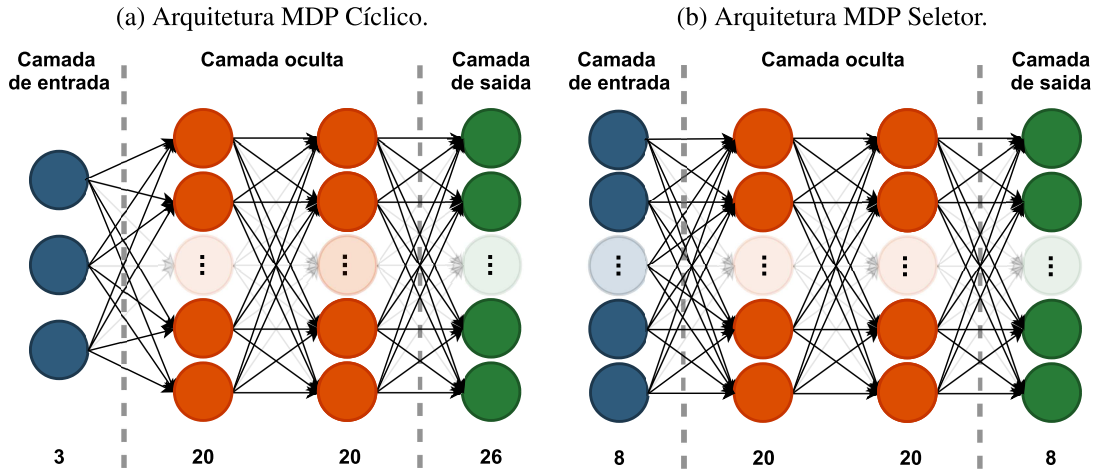
4.4 Definição do Agente

Como dito anteriormente, foram escolhidos dois algoritmos diferentes de aproximadores de função para o Deep Q-Learning. O primeiro deles utiliza o XGBoost e chamamos de **XGB**, o segundo é similar ao artigo original (MNIH et al., 2015) e chamamos de **DQN**. Para ambos os algoritmos, como entrada usamos as features que descrevem o estado, e como saída, ambos estimam o valor Q de cada ação. Com base na saída de cada modelo, cada agente escolhe a ação com o maior valor Q para o estado em questão.

Para o DQN, a Figura 14a apresenta a arquitetura da rede neural para o MDP Cíclico, e a Figura 14b para o MDP Seletor. Nesse trabalho a rede neural tem os neurônios da camada de entrada e de saída equivalente ao número de features do estado e de ações do agente, respectivamente. Por isso, precisamos de duas arquiteturas diferentes para as modelagens do ambiente propostas. Empregamos a função de ativação ReLU para todas as camadas, exceto a camada de saída, para a qual empregamos uma função linear. A rede foi treinada utilizando o otimizador RMSprop com função de loss MSE. Vale destacar que a disposição dos neurônios nas camadas ocultas de ambas as redes foram escolhidas baseadas em uma série de testes, os quais são discutidos de forma detalhada na Subseção 5.1.5.

Para o XGB a situação muda: diferente da rede neural, o algoritmo original do XGBoost só permite uma saída para seus modelos. Por isso, para utilizar o XGBoost como aproximador de função para as modelagens de ambiente descritas na Seção 4.3, temos que combinar ao número de saídas um número equivalente de estimadores. Por exemplo, se temos 8 saídas (como no MDP Seletor), temos que ter 8 estimadores XGBoost combinados, sendo um para cada ação. Para o MDP Cíclico, são combinados 26 estimadores XGBoost. Para realizar esse agrupamento de estimadores, foi utilizado a biblioteca *MultiOutputRegressor* do *scikit-learn* (PEDREGOSA et al., 2011).

Figura 14: Arquitetura das redes neurais para o DQN.



Fonte: Elaborada pelo autor.

Para melhorar o desempenho dos algoritmos, utilizamos duas abordagens, conforme descrito abaixo:

- Usamos o método de repetição proporcional da experiência, do inglês *proportional experience replay* (SCHAUL et al., 2015), que se difere do esquema padrão do DQN (MNIH et al., 2013, 2015) na medida em que as experiências são amostradas. No esquema padrão, as experiências aprendidas são selecionadas uniformemente (com mesma probabilidade). Já no *proportional experience replay*, são selecionadas com uma probabilidade proporcional às suas recompensas.

- Usamos a ideia de ter dois modelos para estimar os valores Q (VAN HASSELT; GUEZ; SILVER, 2016). Assim, com o uso de dois estimadores distintos evitamos viés de maximização nas estimativas do valor Q (demonstrado por Van Hasselt, Guez e Silver (2016)). O segundo modelo copia lentamente os parâmetros do primeiro modelo. No nosso trabalho realizamos essa cópia periodicamente ao fim de cada episódio.

Além disso, tanto o XGB quanto o DQN passam por testes de hiperparâmetros de modo a encontrar o melhor modelo que se encaixa na dinâmica do problema atacado. A Seção 5.1.5 apresenta a metodologia para essa otimização, e a Seção 5.2.2 apresenta os resultados obtidos durantes os testes. Além disso, em especial, separamos os resultados da arquitetura da rede neural utilizada no DQN para a Seção 5.2.3.

4.5 Modelo de Explicabilidade

Apesar de parecer que o algoritmo XGB não precise desenvolver um modelo para extrair explicações do modelo, visto que o mesmo é baseado em árvores de decisão, as quais individualmente já podem ser explicadas de maneira intrínseca, o fato de fazer um *ensemble* delas acaba

limitando essa explicabilidade (ARRIETA et al., 2020). Por isso, assim como para o DQN, como demonstrado na Etapa (b) da Figura 11, para obter o modelo de explicabilidade precisamos calcular os valores SHAP. Esses valores são adquiridos por meio do framework SHAP junto ao agente treinado e algumas informações do ambiente previamente selecionadas e calculadas. Ao obter esses valores, podemos dizer que temos um modelo SHAP, e com ele é possível fazer explicações sobre as ações do modelo durante o uso, conforme é apresentado na Etapa (c) da Figura 11.

4.6 Discussão

Com o método descrito nesse capítulo, no primeiro momento a ideia é treinar cada agente em uma instância simulada utilizando o CityFlow. Depois disso, é necessário criar um modelo de explicabilidade com o agente treinado e informações do ambiente previamente selecionadas. Após isso, é possível utilizar o modelo treinado em um ambiente e também extrair informações para fazer a explicação post-hoc. Para testar ambos os algoritmos, realizamos simulação em dois diferentes ambientes. A descrição desses ambientes e os respectivos resultados são descritos no Capítulo 5. Além disso, cada modelo de explicabilidade e seus respectivos resultados são detalhados no Capítulo 6.

5 RESULTADOS

Neste capítulo apresentamos os resultados do controle semafórico inteligente em duas diferentes simulações. A Seção 5.1 descreve a metodologia dos experimentos, os cenários simulados, as métricas utilizadas e os *baselines*, além do processo de otimização dos hiperparâmetros dos algoritmos. Na Seção 5.2 são apresentados os resultados do processo de otimização dos hiperparâmetros. Na Seção 5.3 são mostrados os resultados das simulações com os melhores parâmetros e todos os *baselines* para cada métrica. Por fim, na Seção 5.4 é adicionada uma breve discussão sobre os resultados obtidos.

5.1 Metodologia dos experimentos

Foram definidos dois diferentes cenários para serem simulados. Informações adicionais de cada simulação podem ser observadas nas Seções 5.1.1 e 5.1.2. A Seção 5.1.3 apresentada as métricas escolhidas para avaliar e comparar os algoritmos propostos. A Seção 5.1.4 detalha cada algoritmo escolhido de comparação. E por fim, na Seção 5.1.5 é apresentada a otimização hiperparamétrica.

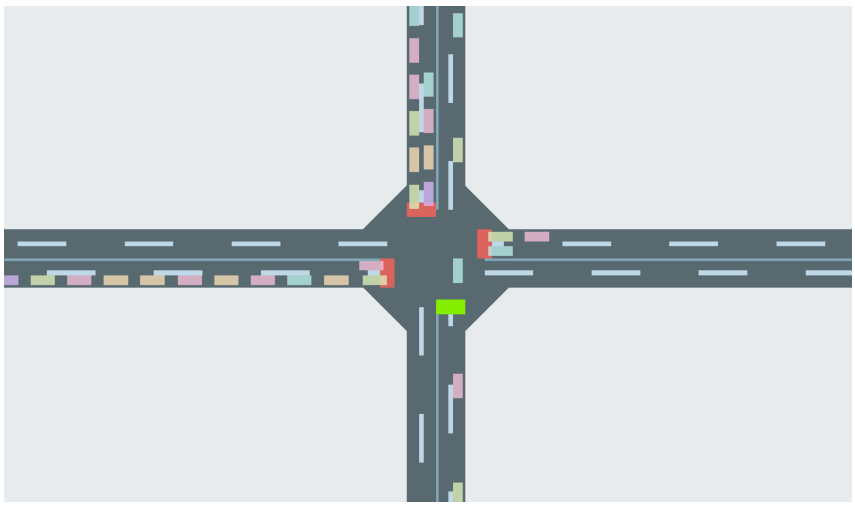
5.1.1 Definição do Cenário 1

Utilizamos uma instância de tráfego do mundo real de Hangzhou, China (ZHENG et al., 2019; WEI et al., 2019a,c). A instância original compreende seis interseções. Entretanto, neste cenário usamos apenas um cruzamento. A Figura 15 apresenta o cruzamento utilizado, o qual tem quatro vias com um limite de velocidade igual a 11,11 metros por segundo (ou seja, 40 quilômetros por hora). Cada acesso tem 300 metros de comprimento. Neste sentido, podemos estimar que o tempo mínimo de viagem é de aproximadamente 55 segundos. Além disso, por definição, cada sinal verde é seguido por um sinal amarelo de 3 segundos e por um sinal vermelho de 2 segundos para todos os semáforos.

O fluxo de veículos foi especificado pelo conjunto de dados *hangzhou_1x1_bc-tyc_18041608_1h*[1]. Esse conjunto de dados foi formado usando câmeras de vigilância entre 01-Abr-2018 e 30-Abr-2018. No conjunto de fluxo de dados, as relações entre dobrar ou seguir reto são fixas, onde 10% dos veículos dobram à esquerda, 60% vão em linha reta, e 30% dobram à direita. Como o movimento em direção à direita não é considerado em nosso trabalho, os dados correspondentes são descartados. A Figura 17a apresenta a matriz de origens e destinos contabilizando cada veículo para o Cenário 1.

[1] Obtido em https://traffic-signal-control.github.io/

Figura 15: Único cruzamento da simulação do Cenário 1.

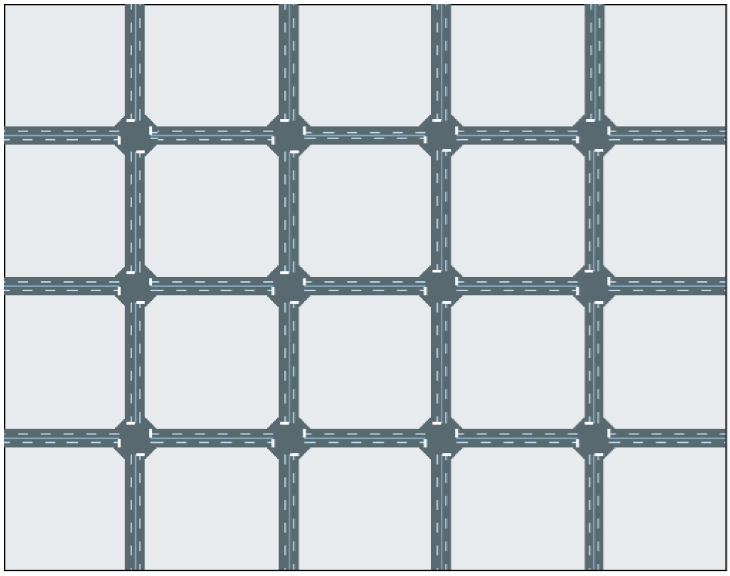


Fonte: Elaborado pelo autor.

5.1.2 Definição do Cenário 2

Utilizamos uma instância de tráfego do mundo real de Jinan, China (ZHENG et al., 2019; WEI et al., 2019a,c). A instância original compreende doze interseções no formato de *grid* 4x3. A Figura 16 apresenta os doze cruzamentos simulados, cada um tem quatro vias com um limite de velocidade igual a 11,11 metros por segundo (ou seja, 40 quilômetros por hora). Cada acesso tem os mesmos 300 metros de comprimento. Neste sentido, o tempo mínimo de viagem em cada cruzamento é de cerca de 55 segundos. Além disso, por definição, cada sinal verde é seguido por um sinal amarelo de 3 segundos e por um sinal vermelho de 2 segundos para todos os semáforos.

Figura 16: Disposição dos doze cruzamentos utilizados na simulação do Cenário 2.

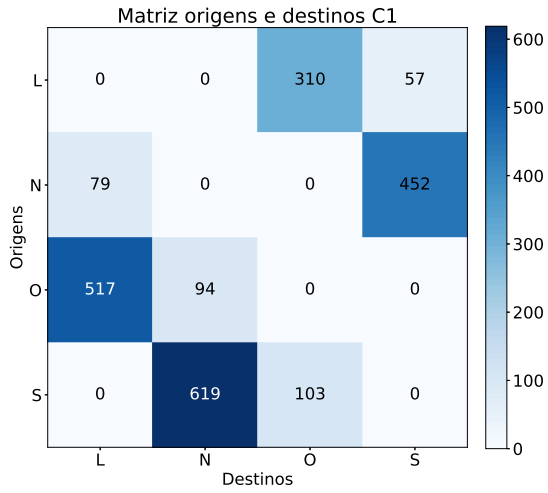


Fonte: Elaborado pelo autor.

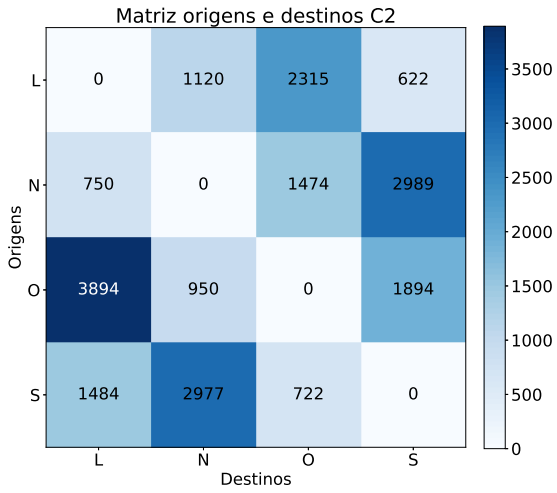
O fluxo de veículos foi especificado pelo conjunto de dados *jinan_3x4_hongqi_16X..X_1h*[2]. Esse conjunto de dados foi formado usando câmeras de vigilância. No conjunto de fluxo de dados, as relações entre dobrar ou seguir reto são fixas, dada uma simplificação feita no *dataset*. Diferente do Cenário 1, os dados de dobrar a direita são considerados nessa simulação; todavia, os semáforos correspondentes a esses movimentos sempre ficam em verde, visto que é sempre permitido dobrar à direita. A Figura 17b apresenta a matriz de origens e destinos contabilizando cada veículo para cada cruzamento do Cenário 2.

Figura 17: Matriz de origens e destinos dos cenários simulados.

(a) Matriz de origens e destinos do Cenário 1.



(b) Matriz de origens e destinos do Cenário 2.



Fonte: Elaborado pelo autor.

5.1.3 Métricas

Para medir o desempenho do agente e compará-lo com outros algoritmos, escolhemos três métricas diferentes disponíveis na literatura (MANNION; DUGGAN; HOWLEY, 2016; YAU et al., 2017; WEI et al., 2019a): tempo médio de viagem, tempo médio de espera e pontuação média de velocidade. Além disso, também adicionamos o tempo de treino, que é uma métrica relacionada ao algoritmo e não diretamente ao controle semafórico. Cada métrica é discutida a seguir. A fim de avaliar o real desempenho, o modelo foi treinado dez vezes. Na Tabela 5 resumimos as métricas baseadas nas suas definições apresentadas abaixo.

[2]Disponível em: https://traffic-signal-control.github.io/ .

Tabela 5: Resumo das informações das métricas utilizadas.

Equação	Nome	Intervalo	Objetivo
5.1	Tempo Médio de Viagem	$[0, +\infty]$	Minimizar ↓
5.2	Pontuação Média de Velocidade	$[0, 1]$	Maximizar ↑
-	*Throughput*	$[0, +\infty]$	Maximizar ↑
-	Tempo Médio de Treino	$[0, +\infty]$	Minimizar ↓

↓ Menor melhor. ↑ Maior melhor.

Fonte: Elaborada pelo autor.

- **Tempo Médio de Viagem**

  Para o controle do sinal de trânsito, o tempo de viagem é a diferença de tempo (em segundos) entre o veículo entrar e sair do cruzamento, ou seja, o tempo necessário para atravessar o cruzamento. Essa métrica é uma das mais utilizadas na literatura. O cálculo da métrica pode ser observado na Equação 5.1.

$$T_{viagem} = \frac{1}{n} \sum_{i=0}^{n} \tau_i \tag{5.1}$$

  Onde n é o número de veículos, τ_i representa o tempo individual de viagem de cada veículo e pode ser calculado por $\tau = t_s - t_e$, t_e é o tempo de entrada e t_s é o tempo de saída. T_{viagem} é medido em segundos e o objetivo é minimizar seu valor, ou seja, quanto menor, melhor.

- **Pontuação Média de Velocidade**

  A métrica de **velocidade** é usada para identificar a velocidade com que um veículo atravessa o cruzamento. Entretanto, os resultados são limitados pelo valor definido no limite de velocidade da estrada (por exemplo, 40 km/h). Portanto, a pontuação de velocidade foi considerada na literatura, calculada como a velocidade do veículo dividida pelo limite de velocidade da rodovia. O cálculo da métrica pode ser observado na Equação 5.2.

$$s_{score} = \frac{1}{n} \sum_{i=0}^{n} \frac{d_i}{v_{max} \cdot \tau_i} \tag{5.2}$$

  Onde n é o número de veículos, d_i é a distância percorrida pelo veículo, τ_i representa o tempo individual de viagem de cada veículo e pode ser calculado por $\tau = t_s - t_e$, t_e é o tempo de entrada e t_s é o tempo de saída. Além disso, v_{max} é a velocidade máxima da pista em segundos. s_{score} é medido em porcentagem e o objetivo é maximizar seu valor, ou seja, quanto maior, melhor.

- ***Throughput***

  A métrica ***Throughput*** é usada para identificar o número de veículos que completaram suas viagens. No caso desse trabalho, a métrica apresenta o número de etapas completas pelos veículos em suas rotas. Uma rota normal de um veículo que deseja atravessar um cruzamento é dividida em duas partes, a etapa de entrada e a de saída. Sabendo disso, para a métrica throughput somamos 1 para cada uma dessas etapas completas, e ao atravessar por completo um cruzamento cada veículo terá somado 2 valores na métrica.

  Se pegarmos como exemplo o fluxo de veículos do cenário 1, apresentado na Figura 17a. Temos um total de 2240 rotas e sabendo serem 2 etapas para cada rota, totalizaria no máximo 4480 de throughput para esse cenário.

- ***Tempo Médio de Treino***

  Essa métrica é usada para medir o tempo de treinamento médio por episódio para cada algoritmo utilizado. Vale destacar que os algoritmos baseados em regras (todos os *baselines* apresentados na Seção 5.1.4) não possuem tempo de treinamento.

5.1.4 Baselines

Como *baselines* para comparar a eficiência de nosso modelo, escolhemos o esquema de tempo fixo (FixedTime) (MILLER, 1963; KOONCE; RODEGERDTS, 2008), o SOTL (COOLS; GERSHENSON; D'HOOGHE, 2013) e o MaxPressure (VARAIYA, 2013). Os três algoritmos têm suas características descritas abaixo.

- **FixedTime** (KOONCE; RODEGERDTS, 2008): tal esquema define um tempo fixo para cada fases e tem um ciclo de fases pré-determinado, ou seja, as fases seguem uma ordem cíclica constante e não mudam. O tempo escolhido foi de 20 segundos. Essa escolha foi baseada no MDP Seletor, que tem a seleção das fases a cada 20 segundos, como descrito na Subseção 4.3.2.

- **SOTL** (COOLS; GERSHENSON; D'HOOGHE, 2013): o *Self-Organizing Traffic Lights Control* (SOTL) regula os semáforos ao estabelecer um limite pré-definido para o número de veículos esperando e um tempo mínimo de verde para a fase atual. Ao atingir tais limites, as fases são alteradas. Não segue uma ordem fixa para as fases. Como dito na Seção 3.2, diversos trabalhos utilizam o SOTL como comparação para seus modelos.

  O SOTL possui 3 diferentes parâmetros, o primeiro deles é o tempo mínimo de verde para a fase, similar ao tempo do FixedTime, então foi fixado em 20 segundos. Os outros dois são limiares baseado no número de veículos na fase atual e nas demais fases. Um estudo desses dois parâmetros é apresentado na Seção 5.2.1.

- **MaxPressure** (VARAIYA, 2013): o MaxPressure escolhe gananciosamente a fase com a maior pressão, definida como o número de carros parados/entrando menos o número de carros saindo do cruzamento. É atualmente o estado da arte na área para o controle semafórico. O único parâmetro é o tempo de decisão para a troca (ou manter) a fase. Assim como para o FixedTime, escolhemos o tempo de 20 segundos.

Ao contrário dos agentes descritos neste trabalho, os *baselines* não precisam ser testados mais de uma vez. Isto porque esses algoritmos têm um comportamento determinístico e também porque o fluxo de veículos é pré-definido (os horários de partida dos veículos não mudam de uma simulação para outra) em vez de serem gerados dinamicamente por uma determinada distribuição. Em outras palavras, sempre que simulamos os veículos partem ao mesmo tempo, do mesmo lugar, e têm comportamentos semelhantes.

5.1.5 Processo de Otimização

De modo a obter o melhor modelo com os melhores resultados para ambos os algoritmos, realizamos uma otimização dos hiperparâmetros. A seguir apresentamos todos os hiperparâmetros testados e seus valores correspondentes. Os principais testes e os melhores modelos são apresentados na Seção 5.2. Vale destacar que alguns valores são alterados quando utilizamos o algoritmo na modelagem de MDP Cíclico e MDP Seletor. Além disso, os hiperparâmetros chamados gerais são os que todos os algoritmos têm, independente da modelagem ou do aproximador de função.

Hiperparâmetros Gerais:

- **Fator de desconto**: testamos os seguintes valores $\gamma \in \{0.2, 0.5, 0.7, 0.8, 0.9, 0.99\}$.
- **Taxa de exploração inicial**: nós testamos $\epsilon \in \{0.5, 1.0\}$.
- **Decaimento da taxa de exploração**: a taxa de decaimento foi variada de duas maneiras. Em primeiro lugar, definimos como linear ou exponencial. Em segundo lugar, quando testado o decaimento exponencial, definimos um valor de decaimento entre 0,95 e 0,99, com um passo de 0,01.
- **Taxa mínima de exploração**: testamos uma taxa mínima de exploração de 0,1 e também uma exploração zero.

Hiperparâmetros para o DQN:

- ***Buffer size***: testamos de 512 até 16384, dobrando a cada teste (começamos com 512, depois 1024 etc.).
- ***Batch size***: ajustamos para 32 tuplas de informação, e depois dobramos até chegar a 1024 tuplas de experiência.

- **Frequência do *replay buffer***: parâmetro para definir a frequência com a qual executaremos o *experience replay*. Foi testado a cada 1, 20, 50, 100, 200, 500, 1000 ações.

- **Atualização da rede *target***: parâmetro para definir a frequência com a qual copiamos os pesos do primeiro modelo para o segundo, como discutido na Seção 4.4. Foi testado a cada 2, 40, 100, 400, 1000 ações.

- **Estrutura da rede neural**: para esse parâmetro, testamos 6 diferentes arquiteturas após a escolha dos hiperparâmetros descritos acima. Por isso, os resultados dessa otimização são separados dos demais e apresentados na Seção 5.2.3. Seguindo o padrão teste(número de camadas ocultas, número de neurônios), os 6 testes foram: teste 1 (1, 20), teste 2 (1, 100), teste 3 (2, 20), teste 4 (2, 100), teste 5 (5, 100), teste 6 (5, 500).

Hiperparâmetros para o XGBoost:

- **Número de estimadores**: número de árvores (*weak learners*). De 50 a 200 de 25 em 25.

- **Profundidade Máxima**: profundidade máxima de uma árvore. Partimos de 3, e vamos até 20 com passo de 1.

5.2 Resultados das Otimizações

Nessa seção, apresentamos os resultados dos testes descritos anteriormente. Os testes do DQN e do XGB foram efetuados com o auxílio da biblioteca Hyperopt (BERGSTRA; YAMINS; COX, 2013). Essa biblioteca ajudou a reduzir a dimensão dos testes, visto que, só para o DQN, o total de possibilidades de testes é mais de 1 milhão.

5.2.1 Resultado dos diferentes parâmetros do SOTL

Como descrito na Subseção 5.1.4, o SOTL possui três parâmetros, o primeiro deles (que corresponde ao tempo para decidir se trocamos de fase ou não) foi fixo em 20 segundos. Os demais parâmetros são baseados no número de veículos, onde temos um parâmetro indicando o número mínimo de veículos (definimos como *mínimo no verde*) na fase atual, e o número máximo de veículos (definimos como *máximo no vermelho*) nas demais fases. Foram então simulados no Cenário 1, 25 testes com esses parâmetros. A Tabela 6 apresenta os 5 melhores resultados para o SOTL. Com os resultados apresentados anteriormente, definimos para o SOTL os seguintes parâmetros: tempo para seleção igual a 20 segundos, numero mínimo de veículos no verde de 10 e numero máximo de veículos no vermelho igual a 20.

Tabela 6: Resultado dos principais testes para o SOTL. Os melhores resultados estão destacados em negrito.

Parâmetros		Métricas (médias)		
Mínimo no Verde	Máximo no Vermelho	Tempo de Viagem (s) ↓	Pontuação de Velocidade ↑	Througput ↑
10	**20**	**394,5**	**0,252**	**3660**
10	30	394,6	0,251	3659
10	10	397,4	0,250	3656
15	30	426,56	0,243	3584
15	20	433,30	0,241	3574

↓ Menor melhor. ↑ Maior melhor.

Fonte: Elaborada pelo autor.

5.2.2 Resultados da Otimização de Hiperparâmetros

Considerando a metodologia de otimização de hiperparâmetros descrita na Seção 5.1.5, detalharemos agora o modelo com melhor desempenho. O melhor modelo para o DQN Cíclico foi aquele treinado com um *buffer size* de 10240 tuplas de experiência, um tamanho de lote de 2048, o *replay buffer* é feito a cada 180 ações e a atualização da rede *target* a cada 360 ações. O fator de desconto foi $\gamma = 0,9$ e a taxa de exploração inicial igual $\epsilon = 1$, com uma taxa de decaimento exponencial de $0,97$ e com valor mínimo de $0,01$. Para o DQN Seletor, o *buffer size* foi de 1000 tuplas de experiência, um tamanho de lote de 32, o *replay buffer* é feito a cada 10 ações e a atualização da rede *target* a cada 40 ações. O fator de desconto foi $\gamma = 0,9$ e a taxa de exploração inicial igual $\epsilon = 1$, com uma taxa de decaimento exponencial de $0,99$ e com valor mínimo de $0,01$.

Para o XGB Cíclico e o XGB Seletor, ambos tiveram o número de estimadores igual a 100 e a profundidade máxima igual a 15. O fator de desconto foi $\gamma = 0,9$ e a taxa de exploração inicial igual $\epsilon = 1$, com uma taxa de decaimento exponencial de $0,99$ e com valor mínimo de $0,01$. A curva de aprendizado obtida pelos nossos modelos são apresentadas na Seção 5.3, onde são apresentados os resultados por cada um dos cenários de simulação, sendo discutido a comparação com os *baselines* para cada métrica apresentada. Em função da quantidade de testes realizados para todos os algoritmos e a limitação de espaço, os resultados foram organizados e apresentados no repositório digital do autor[3].

5.2.3 Resultado dos Testes de Arquitetura do DQN

Os 6 testes descritos Subseção 5.1.5 foram simulados no Cenário 1 para ambos os MDPs (Cíclico e Seletor). Os resultados são apresentados na Tabela 7 e na Figura 18 apenas para o

[3]Disponível em: https://github.com/LincolnVS/resultados-dos-testes-mestrado

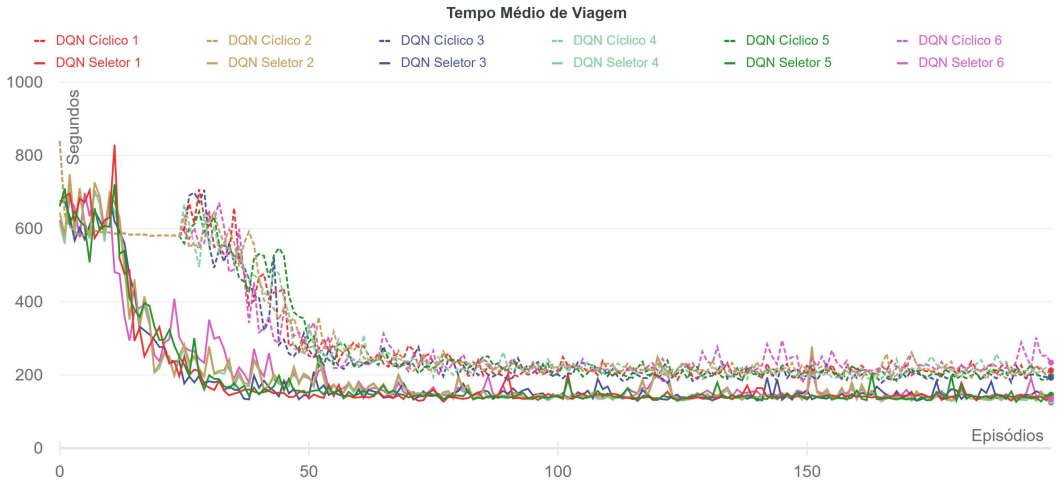
tempo médio de viagem. Com esses resultados podemos definir que as arquiteturas testadas não variaram tanto o resultado, com exceção do teste 6 que se demonstrou um pouco instável. A arquitetura que melhor se encaixou com a dinâmica do ambiente foi a do teste 3 com 2 camadas ocultas e 20 neurônios em cada camada, para ambos os MDPs.

Tabela 7: Resultado dos 7 testes da arquitetura da rede neural. Os melhores resultados estão destacados em negrito.

	Parâmetros		Tempo médio de viagem(s) ↓	
Teste	Número de Camadas Ocultas	Número de Neurônios por Camada	MDP Cíclico	MDP Seletor
1	1	20	187,15	127,02
2	1	100	188,69	130,71
3	**2**	**20**	**180,79**	**125,41**
4	2	100	186,96	127,46
5	5	100	183,96	129,15
6	5	500	181,11	134,14

↓ Menor melhor.
Fonte: Elaborado pelo autor.

Figura 18: Tempo Médio de viagem para os testes de arquitetura da rede neural.



Fonte: Elaborado pelo autor.

5.3 Resultados

5.3.1 Cenário 1

Nesta subseção apresentamos os resultados obtidos na simulação do Cenário 1. A Tabela 8 apresenta os resultados gerais do nosso método e os *baselines* relativos a todas as métricas consideradas. As Figuras 19, 20 e 21 mostram, respectivamente, o Tempo Médio de Viagem, a Pontuação Média de Velocidade e o Throughput ao longo dos episódios para os métodos e para os *baselines*. Como cada experimento foi repetido 10 vezes, na figura também é possível ver áreas sombreadas para representar o desvio padrão.

Tabela 8: Resultados obtidos pela simulação no Cenário 1 com os 2 algoritmos em cada uma das duas modelagens e os três *baselines*. Os melhores resultados estão destacados em negrito.

Algoritmo	Tempo de viagem(s) ↓	Pontuação de velocidade ↑	Throughput ↑	Tempo de treino por episódio (s) ↓
FixedTime	621,79	0,205	3075	-*
SOTL	394,54	0,252	3660	-*
MaxPressure	131,91	0,425	**4327**	-*
DQN Cíclico	196,51 ± 9,05	0,345 ± 0,119	4158 ± 23	17,1
DQN Seletor	132,47 ± 4,88	0,431 ± 0,018	4280 ± 22	**3,9**
XGB Cíclico	178,50 ± 5,61	0,366 ± 0,012	4108 ± 33	306,1
XGB Seletor	$\mathbf{124,96 \pm 4,80}$	$\mathbf{0,468 \pm 0,015}$	**4323 ± 11**	96,6

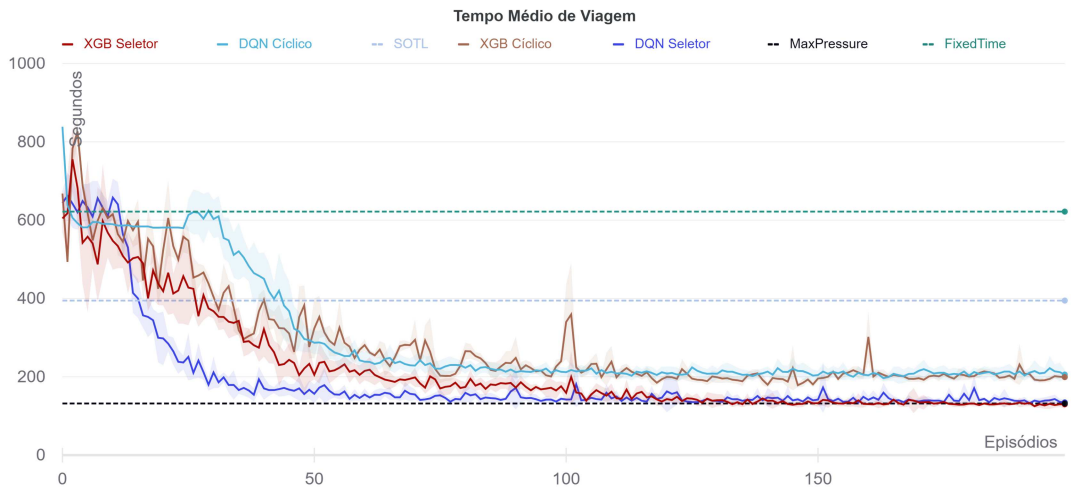
↓ Menor melhor. ↑ Maior melhor.

*Algoritmos baseados em regras não possuem tempo de treino.

Fonte: Elaborada pelo autor.

Como visto na Figura 19, todos os algoritmos conseguem reduzir consideravelmente o tempo médio de viagem ao longo dos episódios. Inicialmente, o tempo de viagem para todos os nossos métodos era por volta de 600, similar ao tempo de viagem do FixedTime. Com o progresso de treinamento, dois agentes se tornaram melhores que todos os *baselines*, o DQN Seletor alcançou o valor de 132,47 ± 4,88 e o XGB Seletor alcançou 124,96 ± 4,8. O DQN Cíclico e o XGB Cíclico também conseguiram reduzir o tempo de viagem, mas não conseguiram ultrapassar o MaxPressure, ficando com 196,51 ± 9,05 e 178,50 ± 5,61.

Figura 19: Tempo médio de viagem em segundos ao longo dos episódios. A área sombreada representa o desvio padrão.

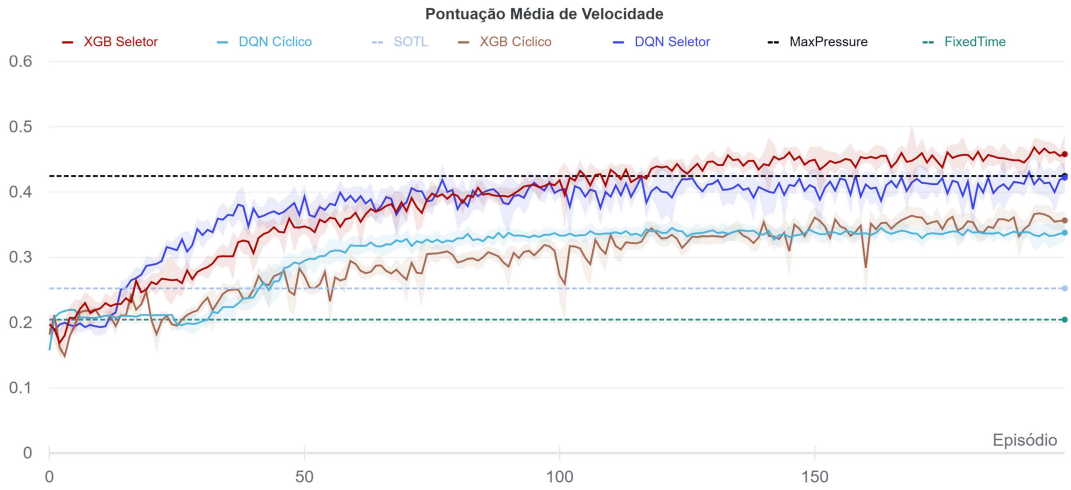


Fonte: Elaborada pelo autor.

Na Figura 20, podemos observar uma melhoria significativa em termos de pontuação de velocidade, o XGB Seletor conseguiu 0,468 ± 0,015. Isso representa mais de 10% de melhoria

em cima do MaxPressure. O comportamento dos algoritmos se assemelha ao observado para o tempo médio de viagem. Inicialmente, o tempo médio está próximo do FixedTime, mas conforme o treinamento continua, nossos métodos vão melhorando na métrica. O DQN Seletor também ficou superior ao MaxPressure, com 0,431 $\pm$ 0,018. Tanto o DQN Cíclico quanto o XGB Cíclico superaram o FixedTime e o SOTL.

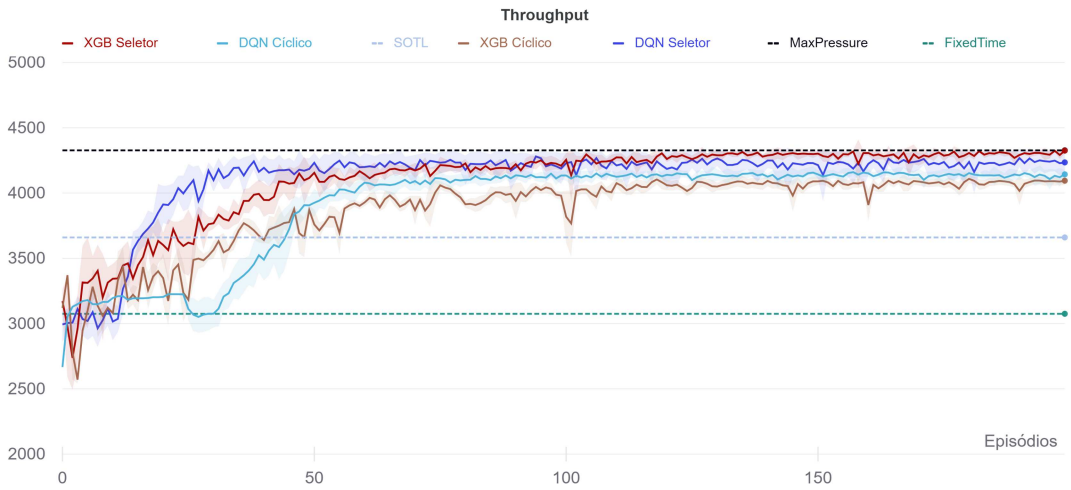
Figura 20: Tempo médio de espera em segundos ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

No caso do Throughput na Figura 21, nossos métodos não conseguiram ultrapassar o MaxPressure, mas ficaram próximos. O XGB Seletor obteve um Throughput próximo a 4323, enquanto os outros esquemas obtiveram pontuações em torno de 4280, 4158 e 4100, para o DQN Seletor, DQN Cíclico e XGB Cíclico, respectivamente.

Figura 21: Pontuação média de velocidade ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

Apesar dos modelos terem gerado bons resultados e terem ultrapassado em desempenho os *baselines*, tanto o XGB Cíclico quanto o DQN Cíclico não conseguiram ter um desempenho superior ao do MaxPressure. Todavia, o XGB Seletor e o DQN Seletor conseguiram ultrapassar em 2 das 3 métricas, e na última métrica ficaram com uma diferença menor de 1,1%. Outro ponto a ser considerado é o tempo de treino entre os dois algoritmos mais eficientes: o DQN Seletor ficou em torno de 24 vezes mais rápido que o XGB Seletor.

5.3.2 Cenário 2

Nesta subseção apresentamos os resultados obtidos na simulação do Cenário 2. A Tabela 9 apresenta os resultados gerais do nosso método e os *baselines* relativos a todas as métricas consideradas. As Figuras 22, 23 e 24 mostram, respectivamente, o Tempo Médio de Viagem, a Pontuação Média de Velocidade e o Throughput ao longo dos episódios para os métodos e para os *baselines*. Assim como no Cenário 1, cada experimento foi repetido 10 vezes e nas figuras também é possível ver áreas sombreadas para representar o desvio padrão. Nessa simulação o algoritmo XGB Cíclico e o DQN Cíclico não conseguiram escalar para múltiplos cruzamentos.

Tabela 9: Resultados obtidos pela simulação no Cenário 2 com os 2 algoritmos em cada uma das duas modelagens e os três *baselines*. Os melhores resultados estão destacados em negrito.

Algoritmo	Tempo de viagem(s) ↓	Pontuação de velocidade ↑	Throughput ↑	Tempo de treino por episódio (s) ↓
FixedTime	473,15	0,491	22591	-*
SOTL	383,27	0,578	24846	-*
MaxPressure	315,99	0,7154	25526	-*
DQN Cíclico**	-	-	-	-
DQN Seletor	$\mathbf{301,65 \pm 1,86}$	$\mathbf{0,745 \pm 0,017}$	$\mathbf{25709 \pm 90}$	**40**
XGB Cíclico**	-	-	-	-
XGB Seletor	372,08 ± 19,11	0,602 ± 0,024	25110 ± 82	396

↓ Menor melhor. ↑ Maior melhor.

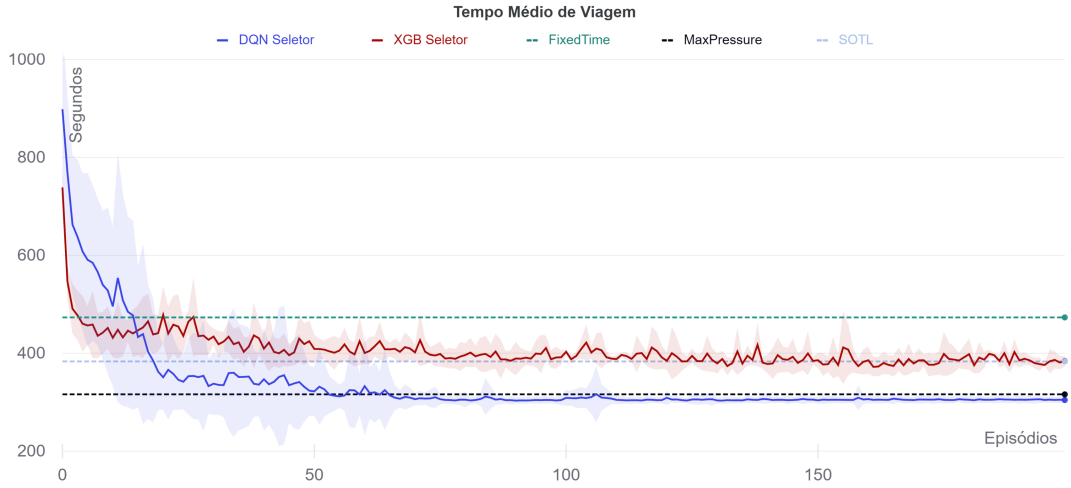
*Algoritmos baseado em regras não possuem tempo de treino.

**O DQN Cíclico e o XGB Cíclico não conseguiram ser escalados para múltiplos cruzamentos.

Fonte: Elaborada pelo autor.

Como visto na Figura 22, todos os algoritmos conseguem reduzir consideravelmente o tempo médio de viagem ao longo dos episódios. Inicialmente, o tempo de viagem, para todos os nossos métodos, era por volta de 800, muito acima do tempo de viagem do FixedTime. Com o progresso de treinamento, o DQN Seletor superou o MaxPressure (315,99) e se tornou o melhor resultado, alçando o valor de 301,65 ± 1,86. O XGB Seletor alcançou 372,08 ± 19,11 ficando próximo do valor do SOTL (383,27 segundos).

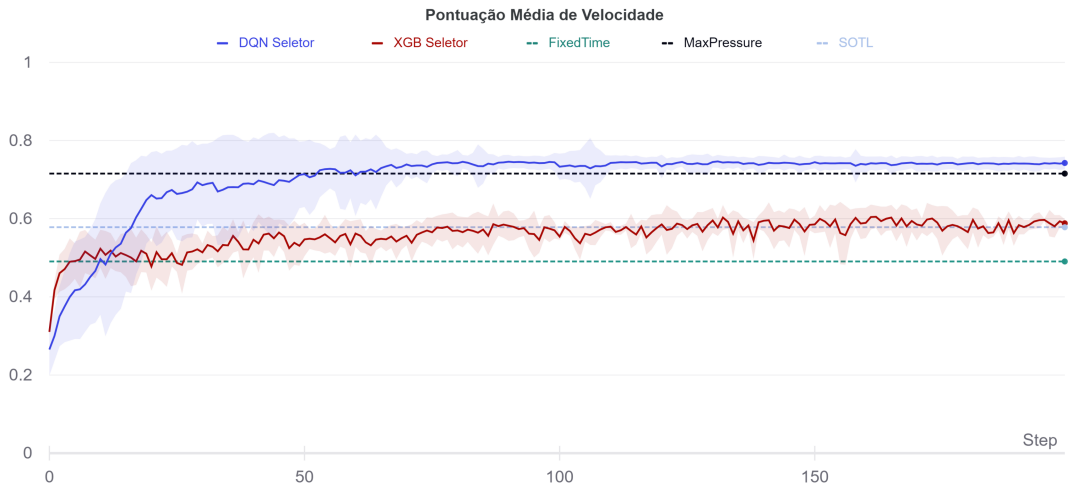
Figura 22: Tempo médio de viagem em segundos ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

Na Figura 23, podemos observar uma melhoria significativa em termos de pontuação de velocidade, o XGB Seletor conseguiu 0,602 $\pm$ 0,024, similar ao SOTL (0.578). Similar ao Tempo de Viagem, inicialmente, a Pontuação de Velocidade estava inferior ao FixedTime, mas conforme o treinamento continua, nossos métodos vão melhorando na métrica. O DQN Seletor foi o único algoritmo que ficou superior ao MaxPressure, com 0,745 $\pm$ 0,017.

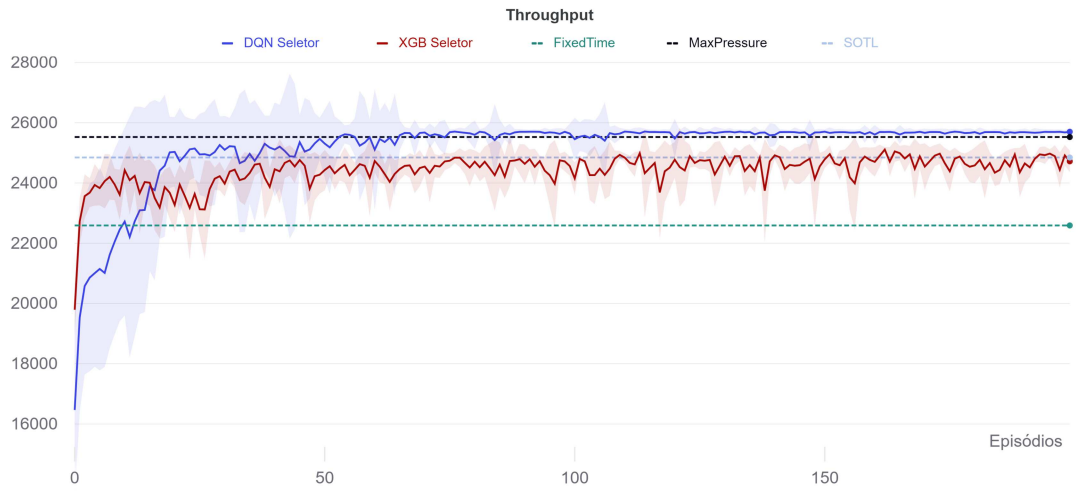
Figura 23: Tempo médio de espera em segundos ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

No caso do Throughput na Figura 24, novamente apenas o DQN Seletor conseguiu ultrapassar o MaxPressure (25526) com 25709 $\pm$ 90. O XGB Seletor obteve um Throughput próximo a 25110, similar ao SOTL (24846).

Figura 24: Pontuação média de velocidade ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

Apesar dos modelos DQN Cíclico não conseguir ser escalado para o Cenário 2, o DQN Seletor conseguiu ultrapassar em todas as 3 métricas o MaxPressure e os demais algoritmos. Além disso, o XGB Cíclico também não conseguiu ser escalado para o Cenário 2 e o XGB Seletor não conseguiu ter um bom desempenho como no Cenário 1. Outro ponto a ser considerado é o tempo de treino: o DQN Seletor ficou em torno de 10 vezes mais rápido que o XGB Seletor.

5.4 Discussão

Como apresentado neste capítulo, podemos perceber que os algoritmos obtiveram êxito ao controlar os semáforos de maneira otimizada, conseguindo assim reduzir o congestionamento. Na simulação do Cenário 1, o melhor algoritmo em questão de desempenho foi o XGB Seletor, que conseguiu ultrapassar o MaxPressure em 2 das 3 métricas utilizadas. Todavia, o algoritmo DQN Seletor também obteve resultados próximos, além de realizar o controle semafórico em torno de 25 vezes mais rápido. Na simulação do Cenário 2, o melhor algoritmo foi o DQN Seletor, que conseguiu ultrapassar o MaxPressure em todas as métricas. O XGB Seletor não conseguiu ser eficiente como no Cenário 1 e não obteve bons resultados.

6 RESULTADOS DA EXPLICABILIDADE

Neste capítulo, apresentamos os resultados de explicabilidade para os modelos treinados. A Seção 6.1 apresenta as explicações post-hoc do algoritmo DQN por meio do Deep SHAP. A Seção 6.2 demonstra todas as informações necessárias para entender como funciona a explicabilidade post-hoc para o algoritmo XGB. A Seção 6.3 explica como foi possível desenvolver essa imagem utilizada para explicabilidade e a diferença com o estado da arte. Por fim, a Seção 6.4 descreve a comparação entre os resultados e uma discussão sobre as limitações e uso.

Vale destacar que no Capítulo 4 apresentamos o algoritmo XGB e o DQN, e duas diferentes modelagens do ambiente. Cada combinação de modelagem e algoritmo possibilitaria extrair informações diferentes. Porém, pelo fato de tanto o XGB Cíclico quanto o DQN Cíclico apresentarem problemas de escalabilidade para o Cenário 2, optamos por não incluir os mesmos nos estudos de explicabilidade apresentadas neste capítulo.

6.1 Explicabilidade para o DQN

Para entender o funcionamento da explicabilidade para o DQN, começamos primeiro com algumas definições. O objetivo da explicabilidade post-hoc para esse algoritmo é investigar como cada feature do estado afeta o valor Q das possíveis ações. Para isso, deve-se entender o que temos de entrada e saída para o modelo. Para o DQN Seletor, por exemplo, temos 8 features que compõem o estado, e 8 ações. Vale destacar que a saída do modelo não é diretamente a ação, mas sim um valor que pode indicar o quanto essa ação traria de retorno para o agente (dinâmica explicada na Seção 2.2.1). Por se tratar de um algoritmo guloso, a ação escolhida é aquela que possui o maior valor Q estimado.

Como discutido na Seção 4.3.2, um estado para o DQN Seletor é representado por uma tupla com 8 features, onde cada uma corresponde ao número de carros esperando na fila de um determinado movimento. O conjunto de movimentos possíveis é apresentado na Tabela 3. Então, para facilitar o entendimento, utilizaremos como referência o seguinte caso, extraído de uma de nossas simulações. Consideramos o estado representado pelas características descritas abaixo e observadas na Figura 25.

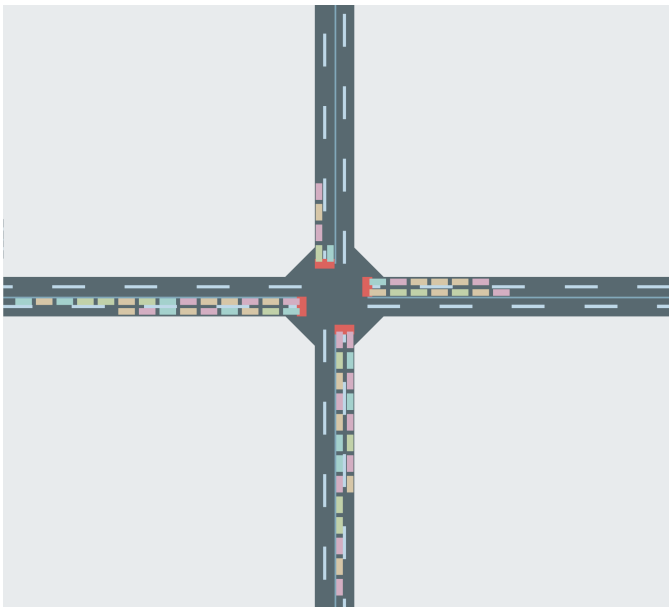
$$S_t = \langle F0; F1; F3; F4; F5; F6; F7 \rangle = \langle \underbrace{9}_{\rightarrow}; \underbrace{6}_{\leftarrow}; \underbrace{8}_{\uparrow}; \underbrace{4}_{\downarrow}; \underbrace{14}_{\rightarrow\!\uparrow}; \underbrace{7}_{\leftarrow\!\downarrow}; \underbrace{13}_{\uparrow\!\leftarrow}; \underbrace{1}_{\downarrow\!\rightarrow} \rangle \quad (6.1)$$

Dado o estado descrito acima, o agente optou por escolher a ação 2, o que significa que a fase atual agora será a Fase 2, a qual permanecerá aberta por 20 segundos. Seguindo a Tabela 2, essa fase é composta pelos movimentos 4 e 5. De modo a entender por que nosso modelo selecionou essa ação, a Figura 26 apresenta o impacto de cada característica (eixo vertical) sobre todas as ações possíveis. Como visto, a figura tem 8 linhas com cores representando as diferentes ações,

o eixo horizontal indica o valor de Q dessas ações. Na barra de cores de cada fase, temos os movimentos que compõem a fase, por exemplo na Fase 2, os movimentos F4 e F5 são as filas de carros nessa fase. Essas definições são generalizadas na Explicação 1.

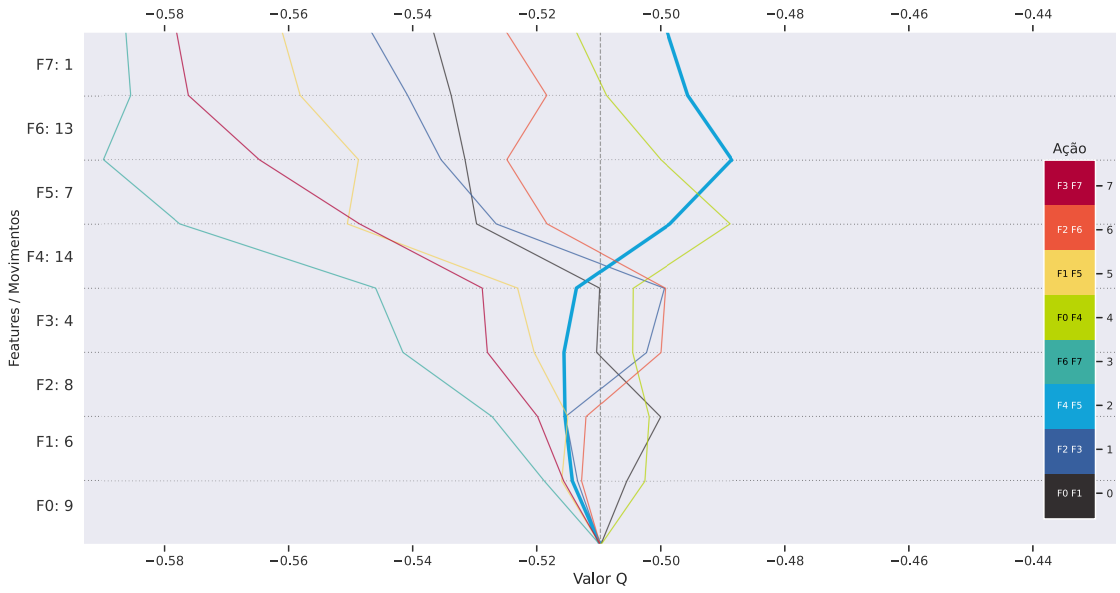
Explicação 1. *As linhas representam as ações (saída) do modelo, o eixo vertical indica as features (entrada ou estados) do mesmo. A barra de cores indica qual o índice da ação e dentro dessa barra podemos observar os movimentos que compõem essa ação.*

Figura 25: Estado utilizado como exemplo.



Fonte: Elaborado pelo autor.

Figura 26: A contribuição de todas as features sobre as ações no estado S_t representado pela Figura 25 para o algoritmo DQN.



Fonte: Elaborada pelo autor.

A partir da Figura 26, podemos observar uma explicação local das ações no que diz respeito às features do estado. Isto significa que cada ação tem seu próprio valor SHAP correspondente à contribuição de cada feature. O valor Q de uma ação é então baseado no impacto dessas oito features que compõem o estado. Neste sentido, a ação cuja linha termina mais à direita é aquela com os maiores valores Q, sendo então escolhida pelo nosso modelo (como discutido nas seções anteriores). Além disso, para facilitar o entendimento, consideramos compartilhar a mesma nomenclatura de F0 até F7 para features do estado, e os níveis de impacto na figura. Daqui para frente, sempre que nos referirmos ao impacto no nível F0, estaremos nos referindo tanto ao eixo vertical da figura, onde F0 corresponde ao primeiro nível de impactos, quanto ao valor da feature F0, descrito na Equação 6.1.

Para analisar a Figura 26, deve-se começar olhando de baixo para cima. O primeiro nível é F0, onde a ação começa a partir do *valor base*[1] sendo agregado à contribuição de F0. No nível de F1, o valor parte do valor anterior (valor de F0) e soma com o impacto de F1. O mesmo é realizado para os restantes dos níveis. Esse comportamento é formalizado na Explicação 2. No topo do gráfico os impactos de todas as features já foram calculados. Na figura, a inclinação para a direita representa um impacto positivo. Por outro lado, a inclinação para a esquerda representa um impacto negativo. Esse comportamento é formalizado através da Explicação 3.

Explicação 2. *A figura é interpretada de baixo para cima, partindo do valor base e agregando o impacto de cada feature na ação analisada. Ao final da soma dos impactos, temos o valor de saída do modelo, marcado no topo da figura para cada ação.*

Explicação 3. *O impacto de uma feature sobre uma saída do modelo é calculado pela inclinação da reta. Impactos positivos geram inclinações para a direita, impactos negativos para a esquerda. O ângulo da inclinação indica a grandeza do impacto.*

Para entender como o modelo pode ser explicado, considere a linha azul-claro destacada na Figura 26, que representa a ação (fase) 2. A partir do valor base, esta linha está inclinada para a esquerda no nível F0, o que significa que F0 tem um impacto negativo sobre o valor Q da ação. No próximo nível (F1), a linha continua inclinada para a esquerda, mas menos acentuada. O mesmo se repete em F2, mas em F3 a linha inclina um pouco para a direita. Em F4 e F5 é onde a maior inclinação para a direita acontece, isso indica que essas features tem um grande impacto positivo sobre o valor Q da ação. F6 e F7 voltam a inclinar para a esquerda.

O estado atual possui mais veículos na Fase 2 do que na maioria das outras fases, o que tornaria justificável escolher essa ação. Observando a Figura 26, podemos extrair a seguinte informação: escolher fases que representam filas com mais carros, também (em grande maioria) corresponde a recompensas maiores. Intuitivamente, isto significa que o agente, nesse estado, prefere escolher uma fila grande para evitar que um número maior de carros fique esperando. Além disso, devemos lembrar que a recompensa do DQN Seletor é o *pressure*, que também

[1] Do inglês *base value*. Segundo Lundberg e Lee (2017), este é o valor que seria previsto se não soubéssemos nenhuma das features do estado atual.

foca no número de carros entrando (não apenas nos carros que sairiam). Isso indica que nem sempre filas grandes vão representar o maior valor Q, mas sim filas que também têm um grande número de carros entrando. Percebe-se a relevância das características do agente e do ambiente para determinar tal comportamento. Essa interpretação é formalizada através da Explicação 4.

Explicação 4. *A relação entre os estados e as ações, além da função de recompensa, deve ser considerada para extrair informações da imagem de explicabilidade. Apenas a imagem não consegue representar toda a dinâmica envolvida entre a interação do agente e o ambiente.*

Considerando então outro exemplo, escolhemos a linha em laranja, que representa a Fase 6. Como visto, a feature F0 tem um impacto negativo sobre esta ação. Já F1, F2 e F3 têm uma inclinação para a direita, indicando um impacto positivo. F4 e F5 impacto negativo, F6 positivo e F7 negativo. Como evidenciado pelo comportamento de inclinação das linhas, para a ação 6 não é apenas os movimentos que compõem a fase que impactam positivamente. Isso pode indicar uma lógica mais complexa entendida pelo modelo, em alguns casos, uma relação indireta com os cruzamentos vizinhos.

Partindo da discussão das duas ações usadas de exemplo (azul-claro e laranja), ambas as fases possuem um número significativo de carros esperando (21 carros). Todavia, a soma do impacto de cada fila sobre a Fase 2 resultou em uma recompensa esperada (valor Q) maior que na Fase 6. Não obstante, ao observar o comportamento de outras fases (como a Fase 7), torna-se claro que a quantidade de carros esperando em suas filas influencia na escolha da ação.

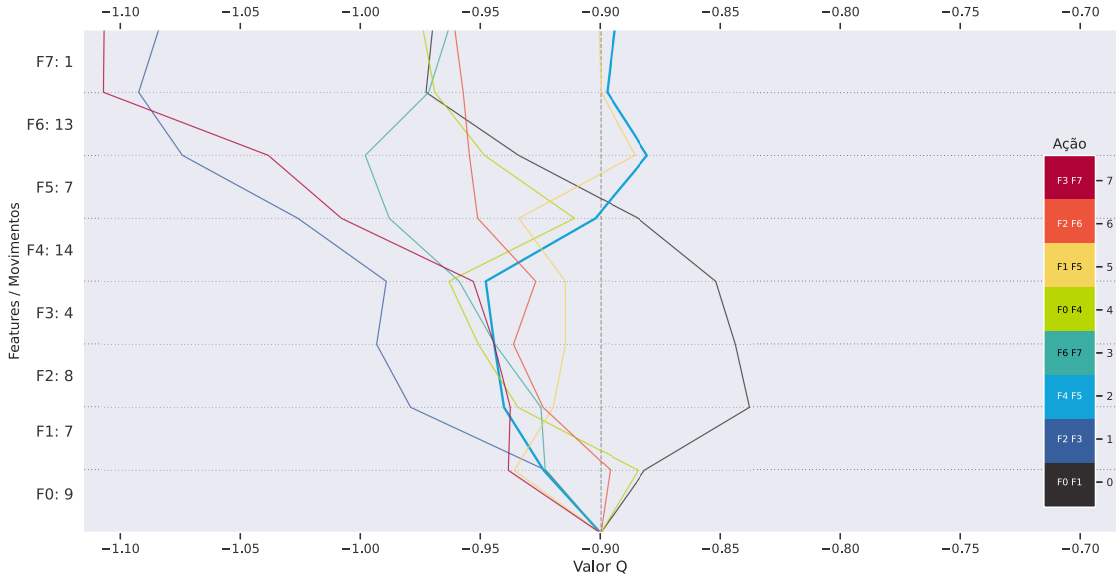
6.2 Explicabilidade para o XGB

Como explicado na Seção 4.5, apesar do algoritmo XGB aparentar não precisar de um modelo para extrair explicações (visto que é baseado em árvores de decisão), o mesmo não é considerado um algoritmo transparente (ou com explicações intrínsecas), pois ao fazer o *ensemble* de árvores acabamos limitando sua explicabilidade (ARRIETA et al., 2020). Com isso, algumas abordagens existentes na literatura podem ser usadas para tornar o modelo explicável. Uma forma inicial de se pensar em extrair explicações do XGB seria apresentar todas as árvores geradas pelo algoritmo e acompanhar cada decisão dessas árvores. Todavia, nota-se que isso é impraticável quando se observa que temos uma árvore para cada estimador. Além do mais, conforme apresentado na Subseção 5.2.2, temos 100 estimadores para cada ação, o que daria um total de 800 árvores para analisar e diversas decisões para tentar extrair informações. Considerando todos esses aspectos, se torna evidente a necessidade de um modelo post-hoc para explicabilidade. Nesse trabalho utilizamos o Tree SHAP para fazer isso.

O objetivo da explicação post-hoc para o XGB é investigar como cada feature do estado afeta o valor Q das ações do agente. Para isso, deve-se entender que para o XGB Seletor temos 8 features que compõem o estado e 8 ações que seriam as saídas do agente. Além disso, a saída do modelo é o valor Q das ações. Por ser um algoritmo guloso, a ação tomada é a que possui o maior valor Q.

Consideramos o mesmo estudo de caso do DQN, representado pelas features descritas na Equação 6.1 e observadas na Figura 25. Como apresentado na Explicação 1, as linhas da figura representam as ações do agente, o eixo vertical representa as features, a barra de cores indica a ação e dentro da barra temos os movimentos que correspondem cada fase. Seguindo as explicações generalizadas durante o processo descrito do DQN (Seção 6.1), a figura deve ser interpretada de baixo para cima, partindo do valor base e somando o impacto de cada feature na ação. Segundo a Explicação 3, conseguimos dizer que o impacto de uma feature sobre uma saída do modelo é calculado pela inclinação da reta, onde inclinações para direita e esquerda indicam impacto positivo e negativo, respectivamente. Apresentado o estudo de caso e a forma de interpretar a explicabilidade gerada pelo framework SHAP, podemos analisar a Figura 25.

Figura 27: A contribuição de todas as features sobre as ações no estado S_t representado pela Figura 25 para o algoritmo XGB.



Fonte: Elaborada pelo autor.

Para entender como o modelo pode ser explicado, considere a ação escolhida pelo agente indicada pela linha azul-claro, a qual representa a ação (fase) 2. A partir do valor base, esta linha é inclinada para a esquerda nos primeiros 4 níveis de features, o que significa que F0, F1, F2 e F3 têm uma influência negativa sobre o valor Q da ação. Nos próximos níveis (F4 e F5), a linha está inclinada para a direita, mas F4 está mais íngreme à direita, indicando que essa feature impacta mais o valor Q dessa ação. F6 novamente inclina a linha para a esquerda e por fim, F7 quase não impacta a linha e apresenta uma breve inclinação para a direita.

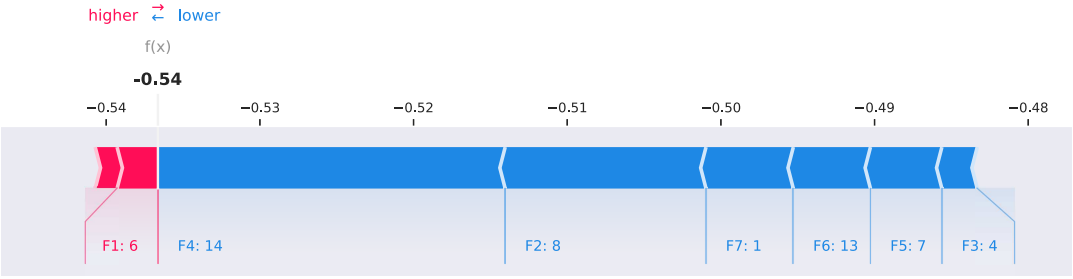
6.3 Diferencial de Explicabilidade

Rizzo, Vantini e Chawla (2019) utilizam um algoritmo de aprendizado por reforço profundo para o controle semafórico e também utilizam o framework SHAP para explicabilidade. To-

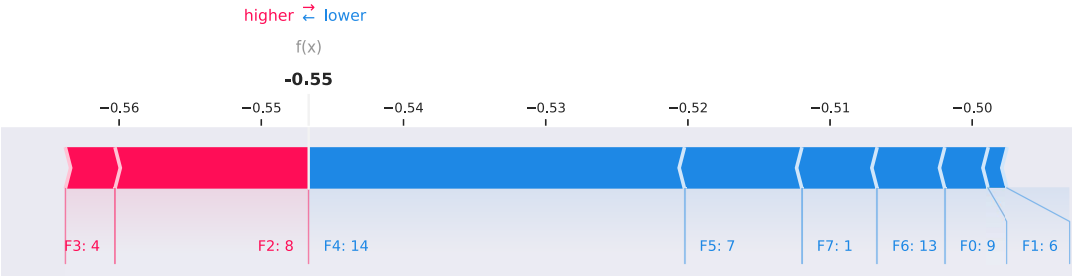
davia, como resultado do trabalho, é necessário ter uma figura de explicabilidade para cada ação do modelo. Considerando o mesmo exemplo de estado apresentado nas seções anteriores (Equação 6.1), as Figuras 28a, 28b e 28c apresentam resultados para as ações 0, 1 e 2 seguindo a mesma metodologia de explicabilidade aplicada por eles (apresentada no framework SHAP como *force plot*). Vale destacar que ainda seria necessário apresentar mais 5 figuras (representando as ações 3, 4, 5, 6 e 7) para conseguir observar e comparar o impacto de todas as features em todas as ações. Como um diferencial, ao utilizar a metodologia empregada nesse trabalho, uma única figura consegue apresentar o impacto dessas features em todas as ações e também consegue destacar a ação escolhida e os movimentos que a compõem (como demonstrado nas Seções 6.1 e 6.2).

Figura 28: Outra forma de mostrar o impacto dos estados em cada ação.

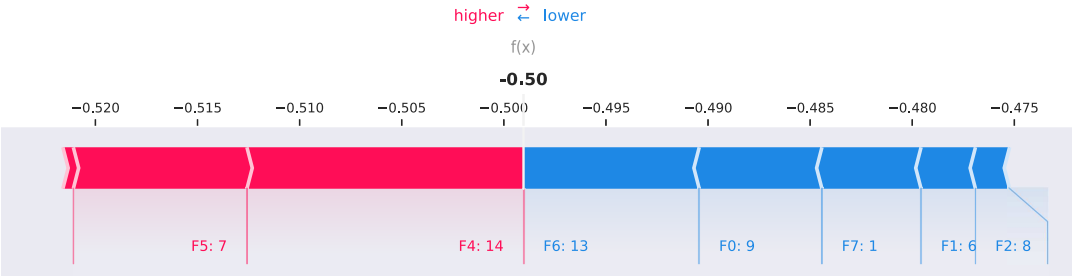
(a) Figura com impacto das features para ação 0.



(b) Figura com impacto das features para ação 1.



(c) Figura com impacto das features para ação 2.

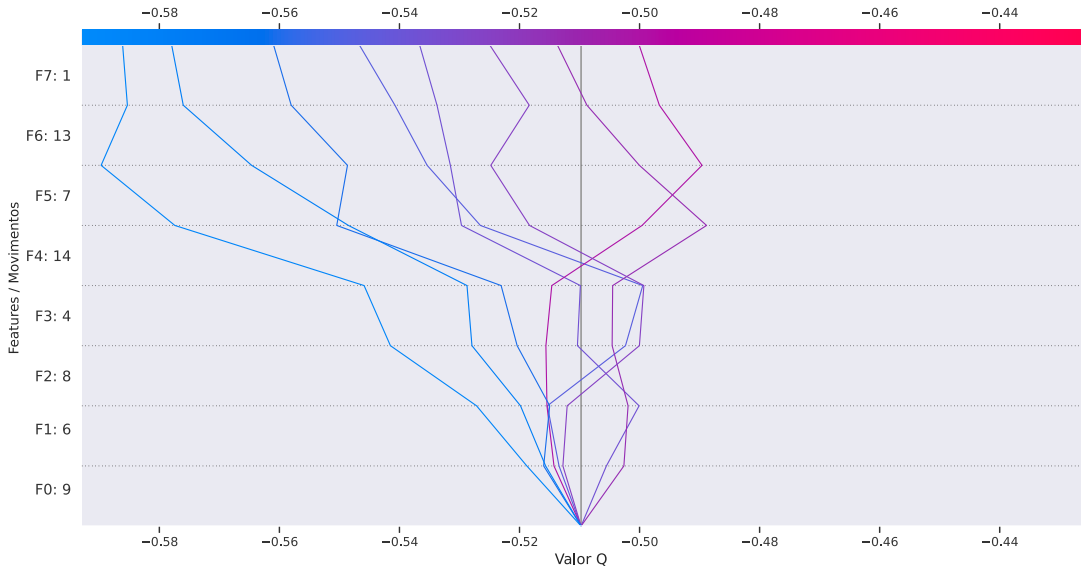


Fonte: Elaborada pelo autor.

De fato, o framework SHAP já possui uma figura que demonstra todos os impactos dos estados nas ações (chamada *multioutput decision plot*). Para o mesmo exemplo de estado com o algortimo DQN Seletor apresentado na Seção 6.1, temos a Figura 29. Entretanto, como pode

ser visto na figura, para esse caso em que estamos, não conseguimos identificar as ações e os movimentos, apenas os valores Q e as ações coloridas de acordo com esses valores. Seria necessário indicar, através de outra visualização, que cor representa que ação e que movimento compõe a mesma. Buscando unificar as vantagens do *force plot* e do *multioutput decision plot*, se estabeleceu a proposta utilizada para explicações nesse capítulo (apresentada nas Figuras 26 e 27).

Figura 29: A contribuição de todas as features sobre as ações no estado S_t representado pela Figura 25.



Fonte: Elaborada pelo autor.

Para gerar a figura apresentada nesse trabalho a partir do *multioutput decision plot* original do framework SHAP, as seguintes mudanças devem ser realizadas:

- Deve-se remover a barra de cores horizontal que representa o valor Q.
- Deve-se adicionar uma barra de cores vertical que representa as ações possíveis do modelo.
- Deve-se colorir cada linha com as cores indicadas na nova barra de cores vertical.
- Na barra de cores, deve-se indicar as features que compõem aquela ação (para o caso do trânsito, os movimentos que compõem a fase).
- (Opcional) Destacar a ação com maior valor Q (aumentar espessura ou mudar estilo), indicando assim a ação escolhida pelo modelo.
- Observação: para um número elevado de ações, optar por utilizar um conjunto de cores gradiente para a barra de cores vertical, como utilizado na Figura 34, que apresenta o resultado de explicabilidade do estudo preliminar descrito no Apêndice A.

6.4 Discussão e Limitações

Como visto na Figura 26 e descrito na Seção 6.1, as ações para o estado atual com mais veículos em suas filas tendem a ter uma recompensa maior que as ações com menos veículos. Além disso, quase todas as ações são impactadas positivamente por alguma feature que não as compõem. A partir disto, podemos observar como o modelo lida com as features para obter o melhor resultado possível. Seguindo os exemplos dados, para otimizar este caso em particular, precisaríamos escolher uma ação que represente um número significativo de veículos para reduzir as filas na fase escolhida, mas também considerar possíveis fluxos de entrada, visto que a recompensa é calculada com base na quantidade de carros que saíram e entraram durante os 20 segundos de ação.

Para o algoritmo XGB, como visto na Figura 27 e descrito na Seção 6.2, a ação escolhida incluiu os movimentos que tinham mais carros esperando, assim como para o algoritmo DQN. Uma diferença encontrada entre os dois algoritmos se dá pelos valores Q de cada ação. Além disso, a ordem das 4 melhores ações (se considerarmos as que possuem maior valor Q) é diferente. Para o algoritmo XGB, as melhores ações são 2, 5, 6 e 3. Para o algoritmo DQN, as melhores ações são 2, 4, 6 e 0. Com isso, percebemos que os modelos identificaram determinadas dinâmicas diferenciadas (alguns impactos de features em determinadas ações não são os mesmos para os algoritmos), onde o DQN obteve melhor resultado (baseado na simulação do Cenário 2), mas nesse caso específico, ambos escolheram a mesma ação a ser tomada.

Todavia, o fato de uma determinada característica ter um impacto positivo ou negativo sobre as ações em qualquer um dos algoritmos, como descrito acima, não significa que ela terá um impacto igual em todos os estados. Na verdade, tudo o que podemos explicar é que quando estamos em um determinado estado, as features têm um impacto nas ações com base em seus valores, e com isso, podemos entender a importância dessa característica localmente. No entanto, no presente trabalho, não podemos justificar um aumento ou diminuição em outro estado pelo valor que a característica tem nesse estado. Além disso, nem todos os estados apresentam comportamento lógico em suas ações, e mesmo considerando que o modelo está convergindo e obtendo excelentes resultados nas métricas, pode-se identificar outliers na explicabilidade e, por consequência, no comportamento do agente.

Além disso, vale recapitular que a Figura 26 é gerada por uma composição de valores e figuras geradas pelo framework SHAP e o agente, e por padrão deve-se seguir alguns passos para permitir extrair e identificar informações de todas as features e suas relações com as ações em uma única imagem. Em nosso conhecimento, não existe representação dessa forma na literatura para o contexto de controle semafórico inteligente utilizando um algoritmo de aprendizado por reforço profundo.

7 CONCLUSÃO

Esse trabalho apresentou dois agentes de aprendizado por reforço profundo para o controle semafórico inteligente explicável. Ambos foram capazes de otimizar o congestionamento de cruzamentos simulados com dados reais e obter políticas explicáveis através de dois diferentes métodos do framework SHAP. O problema de controle semafórico foi caracterizado como um processo de decisão de Markov (MDP) e duas diferentes modelagens do ambiente foram escolhidas, o MDP Cíclico e o MDP Seletor. Cada uma dessas modelagens permitiu aos agentes escolher diferentes ações e ter representações diferenciadas dos estados. Essas características distintas geraram diferentes resultados para os algoritmos.

O primeiro algoritmo utilizado foi o XGB, que utilizou um método *ensemble* baseado em árvores chamado XGBoost como aproximador de função para o Q-learning. O segundo algoritmo utilizado foi o DQN, que utiliza uma rede neural como aproximador de função para o Q-learning. Ambos os algoritmos passaram por um processo de estudo e otimização de seus hiperparâmetros. Dois cenários simulados foram definidos, ambos utilizando instâncias de tráfego real. O Cenário 1 considerou apenas um cruzamento, o Cenário 2 considerou doze interseções no formato de *grid* 4x3.

Durante a simulação do Cenário 1, o XGB utilizando o MDP Seletor encontrou políticas que reduziram o tempo médio de viagem, enquanto aumentaram a velocidade média dos veículos e a quantidade de veículos que conseguiu atravessar o cruzamento (*throughput*). Os resultados da simulação provaram que o agente teve o melhor desempenho que os baselines de tempo fixo, SOTL e MaxPressure utilizados para comparação. O DQN utilizando o MDP Seletor encontrou políticas que permitiram obter bons resultados para ambas as métricas, ficando próximo do XGB Seletor e do MaxPressure. Apesar dos modelos gerarem bons resultados na simulação do Cenário 1, tanto o XGB Cíclico quanto o DQN Cíclico não conseguiram ter um desempenho próximo ao dos demais algoritmos. Em relação ao tempo de treinamento, entre os dois algoritmos que foram mais eficientes, o DQN Seletor ficou em torno de 24 vezes mais rápido que o XGB Seletor.

Na simulação do Cenário 2, os modelos XGB Cíclico e o DQN Cíclico não foram capazes de apresentar um desempenho minimamente aceitável, tendo sido descartados. Por outro lado, o XGB Seletor conseguiu otimizar o congestionamento e ultrapassou o *baseline* SOTL. Todavia, ele ficou abaixo do desempenho do DQN Seletor e do MaxPressure. O DQN Seletor conseguiu ultrapassar todos os outros algoritmos em todas as 3 métricas e em critérios de tempo de treino ficou em torno de 10 vezes mais rápido que o XGB Seletor nessa simulação.

Utilizando o framework SHAP foi possível obter os impactos (contribuição) de cada feature nas ações do agente. O algoritmo XGB obteve suas explicações post-hoc por meio do Tree SHAP e o algoritmo DQN por meio do Deep SHAP. Apesar de cada modelagem do ambiente possibilitar extrair informações de forma diferente, optou-se por não incluir o XGB Cíclico e o DQN Cíclico nos estudos de explicabilidade, dado que estes não apresentaram bons resultados

no Cenário 1 e também por não serem escaláveis para o Cenário 2, o que inviabilizaria seu uso no mundo real. Diferente dos demais trabalhos que utilizam aprendizado por reforço profundo e o framework SHAP aplicado no controle semafórico, conseguimos apresentar o impacto das features por uma representação unificada de todas as ações. Também foi possível destacar a ação escolhida e os movimentos que a compõem. Em nosso conhecimento, não existe uma representação unificada na literatura nesse contexto.

Tanto para o DQN quanto para o XGB foi possível compreender a importância das features do estado localmente e discutir sobre a coerência na lógica dos modelos, mesmo sob algumas limitações. Os resultados obtidos pelo atual trabalho mostraram que podemos utilizar a metodologia descrita para melhorar a compreensão das políticas e aumentar sua confiabilidade e segurança. No geral, a ação escolhida pelos agentes incluiu os movimentos que tinham mais carros esperando, apesar de uma notória diferença entre os impactos sobre as ações em cada explicação. Além disso, quase todas as ações são impactadas positivamente por alguma feature que não as compõe, indicando que o agente leva em consideração dinâmicas do ambiente que não foram diretamente descritas com a modelagem do ambiente. Com a dinâmica do ambiente, a modelagem do MDP, as propriedades dos algoritmos e as explicações adquiridas pela metodologia descrita, foi possível observar que para otimizar um determinado estado o agente precisa escolher uma ação que representa um número significativo de veículos. Além disso, o agente também precisa considerar possíveis fluxos de veículos nos demais movimentos, mesmo que eles não sejam diretamente ligados à ação escolhida.

Apesar das discussões com os atuais resultados, o fato de uma determinada característica ter um impacto positivo ou negativo sobre as ações em qualquer um dos algoritmos não significa que ela terá um impacto igual em todos os estados. Na verdade, tudo o que podemos explicar é que quando estamos em um determinado estado, as features têm um impacto nas ações com base em seus valores e, com isso, podemos entender a importância dessa característica localmente. No entanto, no presente trabalho, não podemos justificar um aumento ou diminuição em outro estado pelo valor que a característica tem nesse estado. Além disso, nem todos os estados apresentam comportamento lógico em suas ações. Mesmo considerando que o modelo está convergindo e obtendo excelentes resultados nas métricas, pode-se identificar *outliers* na explicabilidade e, por consequência, falhas na explicação do comportamento do agente.

7.1 Trabalhos Futuros

Como trabalho futuro, delineamos as seguintes direções de pesquisa:

- Embora o MDP Seletor tenha um desempenho superior ao MDP Cíclico (para ambos algoritmos testados), é relevante explorar diferentes modelagens do ambiente e a possibilidade de obter diferentes explicações que agregam mais informações para o usuário.

- Pela dinâmica do algoritmo de aprendizado por reforço, nem sempre a melhor ação é a escolhida para um determinado estado (por exemplo, utilizando o algoritmo $\epsilon - greedy$). Essa informação não é levada em consideração na atual metodologia de explicabilidade. De forma geral, em um trabalho futuro pode-se estudar sobre formas de demonstrar qual seria a melhor ação, qual foi a ação tomada e o que isso pode ter causado de impacto no congestionamento.

- Existem trabalhos na literatura que apresentam uma simplificação de algoritmos similares ao XGBoost (WELLECK, 2015). Essa simplificação permite gerar uma árvore de decisão baseada em todos os estimadores (*weak learners*) do algoritmo. Com isso, seria possível extrair explicações de uma única árvore simplificada. Essa técnica não foi abordada no atual trabalho, e seria interessante trazer uma relação e comparação das explicações extraídas por essa técnica e pelas apresentadas nesse trabalho (Tree SHAP).

REFERÊNCIAS

ABDULHAI, B.; PRINGLE, R.; KARAKOULAS, G. J. Reinforcement learning for true adaptive traffic signal control. **Journal of Transportation Engineering**, [S.l.], v. 129, n. 3, p. 278–285, 2003.

ADADI, A.; BERRADA, M. Peeking inside the black-box: a survey on explainable artificial intelligence (xai). **IEEE access**, [S.l.], v. 6, p. 52138–52160, 2018.

ARRIETA, A. B. et al. Explainable artificial intelligence (xai): concepts, taxonomies, opportunities and challenges toward responsible ai. **Information Fusion**, [S.l.], v. 58, p. 82–115, 2020.

ASLANI, M.; MESGARI, M. S.; WIERING, M. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. **Transportation Research Part C: Emerging Technologies**, [S.l.], v. 85, p. 732–752, 2017.

AULT, J.; HANNA, J.; SHARON, G. Learning an interpretable traffic signal control policy. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS 2020), 19., 2020. **Proceedings...** [S.l.: s.n.], 2020.

BACH, S. et al. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. **PloS one**, [S.l.], v. 10, n. 7, p. e0130140, 2015.

BADAMPUDI, D.; WOHLIN, C.; PETERSEN, K. Experiences from using snowballing and database searches in systematic literature studies. In: INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 19., 2015. **Proceedings...** [S.l.: s.n.], 2015. p. 1–10.

BARCELÓ, J. et al. **Fundamentals of traffic simulation**. [S.l.]: Springer, 2010. v. 145.

BAZZAN, A. L. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. **Autonomous Agents and Multi-Agent Systems**, [S.l.], v. 18, n. 3, p. 342, 2009.

BAZZAN, A. L.; KLÜGL, F. **Introduction to intelligent systems in traffic and transportation**. [S.l.]: Morgan & Claypool Publishers, 2013. 1–137 p. v. 7, n. 3.

BAZZAN, A. L.; KLÜGL, F. A review on agent-based technology for traffic and transportation. **The Knowledge Engineering Review**, [S.l.], v. 29, n. 3, p. 375–403, 2014.

BELLMAN, R. A markovian decision process. **Journal of mathematics and mechanics**, [S.l.], v. 6, n. 5, p. 679–684, 1957.

BENNETOT, A. et al. Towards explainable neural-symbolic visual reasoning. **arXiv preprint arXiv:1909.09065**, [S.l.], 2019.

BERGSTRA, J.; YAMINS, D.; COX, D. Making a science of model search: hyperparameter optimization in hundreds of dimensions for vision architectures. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2013. **Anais...** [S.l.: s.n.], 2013. p. 115–123.

BINGHAM, E. Reinforcement learning in neurofuzzy traffic signal control. **European Journal of Operational Research**, [S.l.], v. 131, n. 2, p. 232–241, 2001.

BREIMAN, L. et al. **Classification and regression trees.** [S.l.]: Wadsworth, 1984.

BURGHOUT, W.; KOUTSOPOULOS, H. N.; ANDREASSON, I. Hybrid mesoscopic–microscopic traffic simulation. **Transportation Research Record**, [S.l.], v. 1934, n. 1, p. 218–225, 2005.

CARVALHO, A. et al. Inteligência artificial–uma abordagem de aprendizado de máquina. **Rio de Janeiro: LTC**, [S.l.], p. 45, 2011.

CASAS, N. Deep deterministic policy gradient for urban traffic light control. **ArXiv preprint ArXiv:1703.09035**, [S.l.], 2017.

CESTNIK, B.; KONONENKO, I.; BRATKO, I. Assistant 86: a knowledge-elicitation tool for sophisticated users. In: EWSL, 1987. **Anais...** Sigma Press: Wilmslow, 1987. p. 31–45.

CHEN, C. et al. Toward a thousand lights: decentralized deep reinforcement learning for large-scale traffic signal control. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2020. **Proceedings...** [S.l.: s.n.], 2020. v. 34, n. 04, p. 3414–3421.

CHEN, T.; GUESTRIN, C. Xgboost: a scalable tree boosting system. In: OF THE 22ND ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 2016. **Proceedings...** [S.l.: s.n.], 2016. p. 785–794.

CHOONG, M. K. et al. Automatic evidence retrieval for systematic reviews. **Journal of medical Internet research**, [S.l.], v. 16, n. 10, p. e223, 2014.

CLAUS, C.; BOUTILIER, C. The dynamics of reinforcement learning in cooperative multiagent systems. **AAAI/IAAI**, [S.l.], v. 1998, n. 746-752, p. 2, 1998.

COOLS, S.-B.; GERSHENSON, C.; D'HOOGHE, B. Self-organizing traffic lights: a realistic simulation. In: **Advances in applied self-organizing systems**. [S.l.]: Springer, 2013. p. 45–55.

DATTA, A.; SEN, S.; ZICK, Y. Algorithmic transparency via quantitative input influence: theory and experiments with learning systems. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY (SP), 2016., 2016. **Anais...** [S.l.: s.n.], 2016. p. 598–617.

DU, M.; LIU, N.; HU, X. Techniques for interpretable machine learning. **Communications of the ACM**, [S.l.], v. 63, n. 1, p. 68–77, 2019.

DUJARDIN, Y.; VANDERPOOTEN, D.; BOILLOT, F. A multi-objective interactive system for adaptive traffic control. **European Journal of Operational Research**, [S.l.], v. 244, n. 2, p. 601–610, 2015.

EL-TANTAWY, S.; ABDULHAI, B.; ABDELGAWAD, H. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. **IEEE Transactions on Intelligent Transportation Systems**, [S.l.], v. 14, n. 3, p. 1140–1150, 2013.

EOM, M.; KIM, B.-I. The traffic signal control problem for intersections: a review. **European transport research review**, [S.l.], v. 12, n. 1, p. 1–20, 2020.

FELIZARDO, K. R. et al. Using forward snowballing to update systematic reviews in software engineering. In: ACM/IEEE INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 10., 2016. **Proceedings...** [S.l.: s.n.], 2016. p. 1–6.

GARCÍA, M. V.; AZNARTE, J. L. Shapley additive explanations for no2 forecasting. **Ecological Informatics**, [S.l.], v. 56, p. 101039, 2020.

GENDERS, W.; RAZAVI, S. Using a deep reinforcement learning agent for traffic signal control. **ArXiv preprint ArXiv:1611.01142**, [S.l.], 2016.

GREENHALGH, T.; PEACOCK, R. Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources. **Bmj**, [S.l.], v. 331, n. 7524, p. 1064–1065, 2005.

GU, J. et al. Double deep q-network with a dual-agent for traffic signal control. **Applied Sciences**, [S.l.], v. 10, n. 5, p. 1622, 2020.

GUNNING, D. **Explainable artificial intelligence (xai)**. 2017. v. 2, n. 2.

HAMAN, I. T. et al. Towards an multilevel agent-based model for traffic simulation. **Procedia Computer Science**, [S.l.], v. 109, p. 887–892, 2017.

HARRI, J.; FILALI, F.; BONNET, C. Mobility models for vehicular ad hoc networks: a survey and taxonomy. **IEEE Communications Surveys & Tutorials**, [S.l.], v. 11, n. 4, p. 19–41, 2009.

HAYKIN, S. **Redes neurais**: princípios e prática. [S.l.]: Bookman Editora, 2007.

HORGAN, D. et al. Distributed prioritized experience replay. **arXiv preprint arXiv:1803.00933**, [S.l.], 2018.

KOONCE, P.; RODEGERDTS, L. **Traffic signal timing manual.** [S.l.]: United States. Federal Highway Administration, 2008.

LIPOVETSKY, S.; CONKLIN, M. Analysis of regression in game theory approach. **Applied Stochastic Models in Business and Industry**, [S.l.], v. 17, n. 4, p. 319–330, 2001.

LUNDBERG, S. M.; ERION, G. G.; LEE, S.-I. Consistent individualized feature attribution for tree ensembles. **arXiv preprint arXiv:1802.03888**, [S.l.], 2018.

LUNDBERG, S. M. et al. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. **Nature biomedical engineering**, [S.l.], v. 2, n. 10, p. 749–760, 2018.

LUNDBERG, S. M. et al. From local explanations to global understanding with explainable ai for trees. **Nature machine intelligence**, [S.l.], v. 2, n. 1, p. 56–67, 2020.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. In: GUYON, I. et al. (Ed.). **Advances in neural information processing systems 30**. [S.l.]: Curran Associates, Inc., 2017. p. 4765–4774.

MA, J.; WU, F. Feudal multi-agent deep reinforcement learning for traffic signal control. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS), 19., 2020. **Proceedings...** [S.l.: s.n.], 2020.

MANNION, P.; DUGGAN, J.; HOWLEY, E. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In: **Autonomic road transport support systems**. [S.l.]: Springer, 2016. p. 47–66.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, [S.l.], v. 5, n. 4, p. 115–133, 1943.

MILLER, A. J. Settings for fixed-cycle traffic signals. **Journal of the Operational Research Society**, [S.l.], v. 14, n. 4, p. 373–386, 1963.

MIRCHANDANI, P.; HEAD, L. A real-time traffic signal control system: architecture, algorithms, and analysis. **Transportation Research Part C: Emerging Technologies**, [S.l.], v. 9, n. 6, p. 415–432, 2001.

MNIH, V. et al. **Playing atari with deep reinforcement learning**. 2013.

MNIH, V. et al. **Human-level control through deep reinforcement learning**. [S.l.]: Nature Publishing Group, 2015. 529–533 p. v. 518, n. 7540.

MOLNAR, C. **Interpretable machine learning**. [S.l.]: Lulu. com, 2020.

MOUSAVI, S. S.; SCHUKAT, M.; HOWLEY, E. Traffic light control using deep policy-gradient and value-function-based reinforcement learning. **IET Intelligent Transport Systems**, [S.l.], v. 11, n. 7, p. 417–423, 2017.

NISHI, T. et al. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS (ITSC), 2018., 2018. **Anais...** [S.l.: s.n.], 2018. p. 877–883.

OMEIZA, D. et al. Explanations in autonomous driving: a survey. **ArXiv**, [S.l.], v. abs/2103.05154, 2021.

PAPAGEORGIOU, M. et al. Review of road traffic control strategies. **Proceedings of the IEEE**, [S.l.], v. 91, n. 12, p. 2043–2067, 2003.

PEDREGOSA, F. et al. Scikit-learn: machine learning in Python. **Journal of Machine Learning Research**, [S.l.], v. 12, p. 2825–2830, 2011.

PUIUTTA, E.; VEITH, E. M. Explainable reinforcement learning: a survey. In: INTERNATIONAL CROSS-DOMAIN CONFERENCE FOR MACHINE LEARNING AND KNOWLEDGE EXTRACTION, 2020. **Anais...** [S.l.: s.n.], 2020. p. 77–95.

QUINLAN, J. R. Discovering rules by induction from large collections of examples. **Expert systems in the micro electronics age**, [S.l.], 1979.

QUINLAN, J. R. **C4.5**: programs for machine learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

RAMOS, G. de. O. et al. Toll-based learning for minimising congestion under heterogeneous preferences. In: INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS 2020), 19., 2020, Auckland, New Zealand. **Proceedings...** IFAAMAS, 2020. p. 1098–1106.

RASHEED, F. et al. Deep reinforcement learning for traffic signal control: a review. **IEEE Access**, [S.l.], 2020.

RIBEIRO, M. T.; SINGH, S.; GUESTRIN, C. "why should i trust you?"explaining the predictions of any classifier. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 22., 2016. **Proceedings...** [S.l.: s.n.], 2016. p. 1135–1144.

RIZZO, S. G.; VANTINI, G.; CHAWLA, S. Reinforcement learning with explainability for traffic signal control. In: IEEE INTELLIGENT TRANSPORTATION SYSTEMS CONFERENCE (ITSC), 2019., 2019. **Anais...** [S.l.: s.n.], 2019. p. 3567–3572.

RUSSELL, S.; NORVIG, P. **Artificial intelligence**: a modern approach. [S.l.]: Prentice Hall Upper Saddle River, NJ, USA, 2002.

SCHAUL, T. et al. **Prioritized experience replay**. 2015.

SCHRANK, D.; EISELE, B.; LOMAX, T. **Urban mobility report 2019**. [S.l.]: Texas A&M Transportation Institute, 2019.

SCHREIBER, L. V.; RAMOS, G. d. O.; BAZZAN, A. L. Towards explainable deep reinforcement learning for traffic signal control. **International Conference on Machine Learning Conference: LatinX in AI (LXAI) Research Workshop 2021**, [S.l.], 2021.

SEN, S.; HEAD, K. L. Controlled optimization of phases at an intersection. **Transportation science**, [S.l.], v. 31, n. 1, p. 5–17, 1997.

SHRIKUMAR, A.; GREENSIDE, P.; KUNDAJE, A. Learning important features through propagating activation differences. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING, 2017. **Anais...** [S.l.: s.n.], 2017. p. 3145–3153.

SILVA, C. R. Q. **Critérios para priorização de estudos primários identificados por snowballing com conjunto inicial gerado por string de busca**. 2017. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de São Carlos, 2017.

SIMS, A. G.; DOBINSON, K. W. The sydney coordinated adaptive traffic (scat) system philosophy and benefits. **IEEE Transactions on vehicular technology**, [S.l.], v. 29, n. 2, p. 130–137, 1980.

ŠTRUMBELJ, E.; KONONENKO, I. Explaining prediction models and individual predictions with feature contributions. **Knowledge and information systems**, [S.l.], v. 41, n. 3, p. 647–665, 2014.

SUTTON, R. S.; BARTO, A. G. **Introduction to reinforcement learning**. [S.l.]: MIT press Cambridge, 1998. v. 135.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning**: an introduction. 2nd. ed. Cambridge, MA, USA: MIT Press, 2018.

THORPE, T. L.; ANDERSON, C. W. **Traffic light control using sarsa with three state representations**. [S.l.]: Citeseer, 1996.

VAN HASSELT, H.; GUEZ, A.; SILVER, D. Deep reinforcement learning with double q-learning. In: AAAI CONFERENCE ON ARTIFICIAL INTELLIGENCE, 2016. **Proceedings...** [S.l.: s.n.], 2016. v. 30, n. 1.

VARAIYA, P. The max-pressure controller for arbitrary networks of signalized intersections. In: **Advances in dynamic network modeling in complex transportation systems**. [S.l.]: Springer, 2013. p. 27–66.

WANG, D. et al. Designing theory-driven user-centric explainable ai. In: CHI CONFERENCE ON HUMAN FACTORS IN COMPUTING SYSTEMS, 2019., 2019. **Proceedings...** [S.l.: s.n.], 2019. p. 1–15.

WATKINS, C. **Learning from delayed rewards**. 1989. Tese (Doutorado em Ciência da Computação) — University of Cambridge, 1989.

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

WEI, H. et al. Intellilight: a reinforcement learning approach for intelligent traffic light control. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, 24., 2018. **Proceedings...** [S.l.: s.n.], 2018. p. 2496–2505.

WEI, H. et al. A survey on traffic signal control methods. **ArXiv**, [S.l.], v. abs/1904.08117, 2019.

WEI, H. et al. Presslight: learning max pressure control to coordinate traffic signals in arterial network. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, 25., 2019. **Proceedings...** [S.l.: s.n.], 2019. p. 1290–1298.

WEI, H. et al. Colight: learning network-level cooperation for traffic signal control. In: INTL. CONF. ON INFORMATION AND KNOWLEDGE MANAGEMENT, 28., 2019. **Proceedings...** [S.l.: s.n.], 2019. p. 1913–1922.

WEI, H. et al. Recent advances in reinforcement learning for traffic signal control: a survey of models and evaluation. **ACM SIGKDD Explorations Newsletter**, [S.l.], v. 22, n. 2, p. 12–18, 2021.

WELLECK, S. **Peering into the black box**: visualizing lambdamart. **wellecks**, 2015. Disponível em: <https://wellecks.wordpress.com/2015/02/21/peering-into-the-black-box-visualizing-lambdamart/>. Acesso em: 11 de Nov. de 2021.

WIERING, M.; VAN OTTERLO, M. Reinforcement learning: state-of-the-art. **Adaptation, learning, and optimization**, [S.l.], v. 12, n. 3, 2012.

WOHLIN, C. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In: OF THE 18TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING, 2014. **Proceedings...** [S.l.: s.n.], 2014. p. 1–10.

WOLLENSTEIN-BETECH, S. et al. Explainability of intelligent transportation systems using knowledge compilation: a traffic light controller case. In: IEEE 23RD INTERNATIONAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS (ITSC), 2020., 2020. **Anais...** [S.l.: s.n.], 2020. p. 1–6.

WU, J. et al. Delay-based traffic signal control for throughput optimality and fairness at an isolated intersection. **IEEE Transactions on Vehicular Technology**, [S.l.], v. 67, n. 2, p. 896–909, 2017.

WUNDERLICH, R. et al. A novel signal-scheduling algorithm with quality-of-service provisioning for an isolated intersection. **IEEE Transactions on Intelligent Transportation Systems**, [S.l.], v. 9, n. 3, p. 536–547, 2008.

XIONG, Y. et al. Learning traffic signal control from demonstrations. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 28., 2019. **Proceedings...** [S.l.: s.n.], 2019. p. 2289–2292.

YAU, K.-L. A. et al. A survey on reinforcement learning models and algorithms for traffic signal control. **ACM Computing Surveys (CSUR)**, [S.l.], v. 50, n. 3, p. 1–38, 2017.

ZHANG, H. et al. Cityflow: a multi-agent reinforcement learning environment for large scale city traffic scenario. In: THE WORLD WIDE WEB CONFERENCE, 2019. **Anais...** [S.l.: s.n.], 2019. p. 3620–3624.

ZHENG, G. et al. Learning phase competition for traffic signal control. In: INTL. CONF. ON INFORMATION AND KNOWLEDGE MANAGEMENT, 28., 2019. **Proceedings...** [S.l.: s.n.], 2019. p. 1963–1972.

APÊNDICE A – ESTUDO PRELIMINAR

Um estudo preliminar foi realizando antes do estudo final apresentado no trabalho. O método, metodologia e os resultados dessa análise são descritos abaixo. Apesar de servirem de embasamento para algumas escolhas feitas no trabalho, os resultados e algumas limitações encontradas já foram superados, por isso consta apenas como um material complementar. Na Seção A.1 apresentamos o método e a metodologia, descrevemos o ambiente, a modelagem, o algoritmo, as métricas de comparação e os baselines. Na Seção A.2 apresentamos os resultados do estudo preliminar. Na Seção A.3 é apresentada a explicabilidade do modelo. Por fim, na Seção A.4 é feita uma breve discussão sobre os resultados obtidos. Vale destacar que o método, a metodologia e os resultados deste estudo foram publicados (SCHREIBER; RAMOS; BAZZAN, 2021).

A.1 Método e Metodologia

A.1.1 Ambiente

Nosso ambiente de testes consiste em uma única interseção isolada com 4 acessos. Similar ao cruzamento apresentado na visão do ambiente para o agente, na Figura 12. Assim como descrito na Seção 4.2, os veículos podem seguir em linha reta, virar à esquerda ou à direita. Além disso, também decidimos ignorar o movimento para a direita da modelagem do ambiente nesse estudo. O fluxo de veículos é baseado em uma instância real de Hangzhou na China, simulado no Cityflow.

O fluxo de veículos foi especificado pelo conjunto de dados *hangzhou_1x1_bc-tyc_18041608_1h*[1] Esse conjunto de dados foi formado usando câmeras de vigilância entre 01-Abr-2018 e 30-Abr-2018. No conjunto de fluxo de dados, as relações entre dobrar ou seguir reto são fixas, onde 10% dos veículos dobram à esquerda, 60% vão em linha reta, e 30% dobram à direita (descartados). O cruzamento tem quatro vias, com um limite de velocidade igual a 11,11 metros por segundo (ou seja, 40 quilômetros por hora). Cada acesso tem 300 metros de comprimento. Neste sentido, podemos estimar que o tempo mínimo de viagem em todo o cruzamento seja de cerca de 54,10 segundos. Além disso, por definição, cada sinal verde é seguido por um sinal amarelo de 3 segundos e por um sinal vermelho de 2 segundos para todos os semáforos.

A.1.2 Modelagem

Modelamos um único agente RL como um controlador de fase fixa com 8 movimentos. Onde a ordem das fases é configurada previamente e não muda. Então, o agente é apenas responsável pela definição da duração de cada fase. As fases e os movimentos correspondentes são

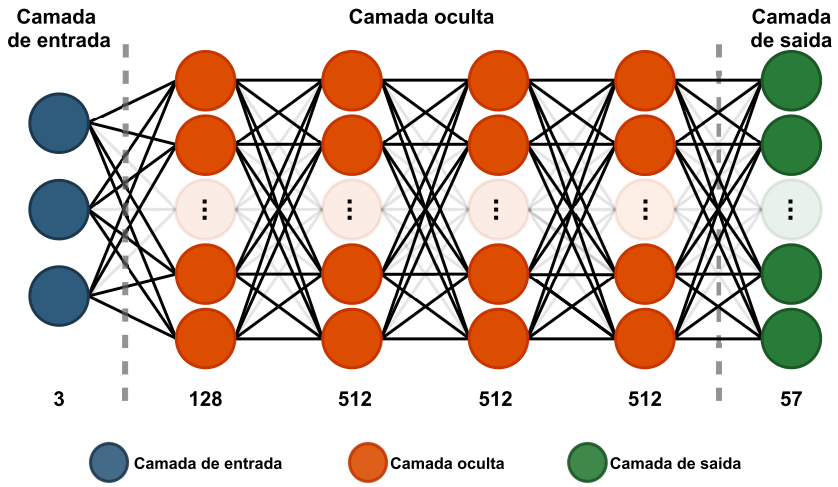
[1] obtido em https://traffic-signal-control.github.io/.

os mesmos apresentados na Tabela 2 na Seção 4.2. A modelagem do ambiente seguiu definições similares do MDP Cíclico, apresentado na Seção 4.3. As ações representam a duração do tempo de verde a ser definido para a fase, podendo variar de 0 a 56 segundos. Os estados possuem 3 features do ambiente, e essas características representam, respectivamente, o comprimento da fila: da fase atual, da fase seguinte, e todas as outras fases não contempladas nas duas primeiras. Por fim, a recompensa é definida pela média do número total de veículos e a diferença entre o número anterior e o atual de veículos esperando.

A.1.3 Algoritmo

A abordagem utilizada foi o Deep Q-Learning (DQN). A Figura 30 apresenta a arquitetura de rede neural proposta. Essa que consiste em seis camadas totalmente conectadas. Onde a camada de entrada tem três nós. As camadas ocultas possuem 128, 512, 512 e 512 nós respectivamente. Por fim, A camada de saída tem 57 nós (nesse caso, o tempo máximo do cruzamento para um semáforo é de 56 segundos, então temos 57 ações possíveis). Quanto às funções de ativação, empregamos ReLU para todas as camadas, exceto a camada de saída, que empregamos uma função linear. A rede foi treinada utilizando o otimizador Adam com função de loss MSE.

Figura 30: Diagrama da rede proposta.



Fonte: Elaborada pelo autor.

Além disso, foi realizado uma otimização de hiperparâmetros com a biblioteca Hyperopt (BERGSTRA; YAMINS; COX, 2013). Testamos o tamanho do *buffer*, tamanho do lote (*batch size*), gamma, epsilon, número de episódios. O melhor modelo foi aquele treinado com 160 episódios, com um *replay buffer* com 10240 tuplas de experiência, e um tamanho de lote de 2048. O fator de desconto foi $\gamma = 0,9$ e a taxa de exploração inicial igual $\epsilon = 1$, com uma taxa de decaimento exponencial de $0,97$ e com valor mínimo de $0,01$.

A.1.4 Métricas

Para medir o desempenho do agente e compará-lo com outros algoritmos, assim como no trabalho final, escolhemos três métricas disponíveis na literatura (MANNION; DUGGAN; HOWLEY, 2016; YAU et al., 2017; WEI et al., 2019a). Nesse estudo escolhemos o tempo médio de viagem, tempo médio de espera e pontuação média de velocidade. O tempo médio de espera é discutido a seguir, as outras métricas seguem a mesma definição apresentada na Seção 5.1.3. A fim de avaliar o real desempenho, o modelo foi treinado dez vezes. Na Tabela 10 resumimos as métricas baseadas nas suas definições apresentadas abaixo.

Tabela 10: Resumo das informações das métricas utilizadas no estudo preliminar.

Equação	Nome	Intervalo	Objetivo
5.1	Tempo Médio de Viagem	$[0, +\infty]$	Minimizar $\downarrow$
5.2	Pontuação Média de Velocidade	$[0, 1]$	Maximizar $\uparrow$
A.1	Tempo Médio de Espera	$[0, +\infty]$	Minimizar $\downarrow$

$\downarrow$ Menor melhor. $\uparrow$ Maior melhor.

Fonte: Elaborado pelo autor.

- **Tempo Médio de Espera**

  O tempo de espera tem definições diferentes na literatura, já que a definição de quando considerar que um veículo está esperando pode ser interpretada de diferentes maneiras. Alguns autores começam a contar o tempo de espera, baseado no tempo de entrada no cruzamento. Outros consideram quando o veículo está parado no cruzamento, esperando o semáforo abrir. Nesse trabalho, consideramos a última opção. O cálculo da métrica pode ser observado na Equação A.1.

$$T_{espera} = \frac{1}{n} \sum_{i=0}^{n} w_i \tag{A.1}$$

  Onde n é o número de veículos, w_i representa o tempo individual de espera de cada veículo e pode ser calculado somando seu tempo esperando no congestionamento. T_{espera} é medido em segundos e o objetivo é minimizar o valor, ou seja, quanto menor, melhor.

A.1.5 Baselines

Como linha de base para comparar a eficiência de nosso modelo, escolhemos o esquema de tempo fixo proposto por Miller (1963). Tal esquema tem um ciclo pré-determinado e um plano de tempo de fase. Entretanto, como esse sistema é um sistema de controle simples, decidimos utilizar três tempos fixos diferentes, com 15, 30, e 45 segundos para cada fase. De agora em

diante, usamos os termos FT15, FT30 e FT45 para representar o esquema de Tempo Fixo com duração de 15, 30 e 45 segundos, respectivamente.

Ao contrário do agente descrito neste trabalho, as linhas de base não precisam ser testadas mais de uma vez. Isto porque estes esquemas têm um comportamento determinístico e também porque o fluxo de veículos é pré-definido (ou seja, os horários de partida dos veículos não mudam de uma simulação para outra) em vez de serem gerados dinamicamente por uma determinada distribuição. Em outras palavras, sempre que simulamos, os veículos partem ao mesmo tempo, no mesmo lugar, e têm comportamento semelhante. Tal comportamento foi empiricamente observado em nossas experiências.

A.2 Resultados

Nesta seção, apresentamos os resultados obtidos pelo nosso método no cenário de simulação proposto. A tabela 11 apresenta os resultados gerais do nosso método e as linhas de base relativas a todas as métricas consideradas. As Figuras 31, 32 e 33 mostram, respectivamente, o tempo médio de viagem, o tempo médio de espera e a pontuação média de velocidade ao longo dos episódios para nosso método e para as linhas de base. Como cada experimento foi repetido 10 vezes, na figura também é possível ver áreas sombreadas para representar o desvio padrão.

Tabela 11: Resultados preliminares obtidos pelo nosso modelo e as três comparações. Os melhores resultados estão destacados em negrito. A última linha apresenta a melhoria média do nosso método em comparação com as linhas de base. Todos os valores são da média por episódio.

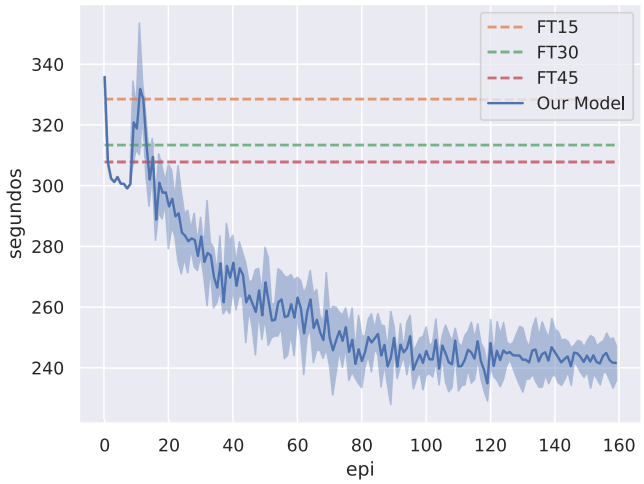
Algoritmo	Tempo de viagem (s) ↓	Tempo de espera (s) ↓	Pontuação de velocidade ↑
FT15	328,44	233,99	0,2482
FT30	313,38	228,64	0,2448
FT45	307,76	229,75	0,2399
Nosso Modelo	$\mathbf{241,62 \pm 5,70}$	$\mathbf{164,70 \pm 6,07}$	$\mathbf{0,30 \pm 0,005}$
Melhoria	23,60%	28,63%	22,82%

↓ Menor melhor. ↑ Maior melhor.

Fonte: Elaborada pelo autor.

Como visto na Figura 31, o tempo médio de viagem diminui consideravelmente ao longo dos episódios. Inicialmente, o tempo de viagem obtido por nosso método era de cerca de 330 segundos, o que é próximo a um obtido pelo esquema FT15. Com o progresso do treinamento, o agente se torna melhor que todas as linhas de base, obtendo um tempo médio de viagem próximo a 240 segundos. Isto representa uma redução de 26,43%, 22,90%, e 21,49%, em comparação com o FT15, FT30, e FT45, respectivamente.

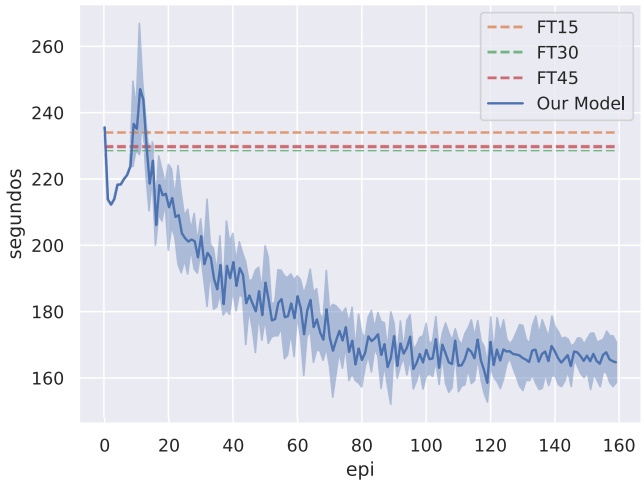
Figura 31: Tempo médio de viagem em segundos ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

Na Figura 32, podemos observar uma melhoria significativa do nosso método em termos de tempo de espera. O comportamento aqui se assemelha ao observado para o tempo médio de viagem. Inicialmente, o tempo médio está próximo do FT15, mas conforme o treinamento continua, nosso método superou as linhas de base e alcançou o valor mínimo médio próximo a 165 segundos. Isto nos dá uma melhora de 29,61%, 27,97% e 28,31% em relação ao FT15, FT30, e FT45, respectivamente.

Figura 32: Tempo médio de espera em segundos ao longo dos episódios. A área sombreada representa o desvio padrão.

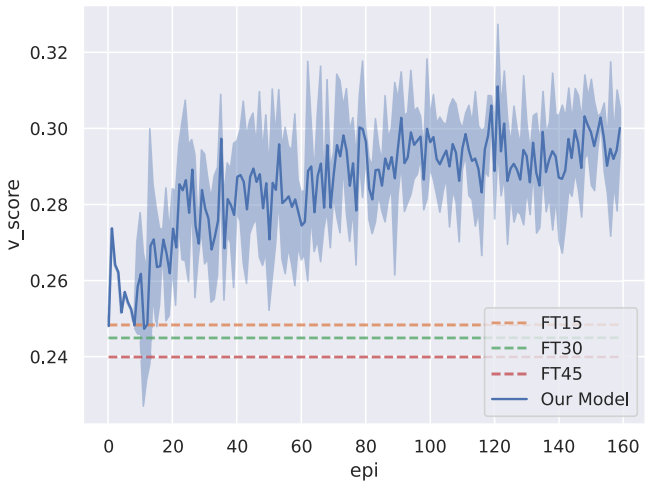


Fonte: Elaborada pelo autor.

No caso da pontuação de velocidade na Figura 33, nosso método, mais uma vez, produziu os

melhores resultados. Especificamente, ele obteve uma pontuação próxima a 30%, enquanto os outros esquemas obtiveram pontuações em torno de 25%. Na verdade, a melhoria da velocidade atingiu 20,87%, 22,55% e 25,05% em comparação com o FT15, FT20 e FT45, respectivamente.

Figura 33: Pontuação média de velocidade ao longo dos episódios. A área sombreada representa o desvio padrão.



Fonte: Elaborada pelo autor.

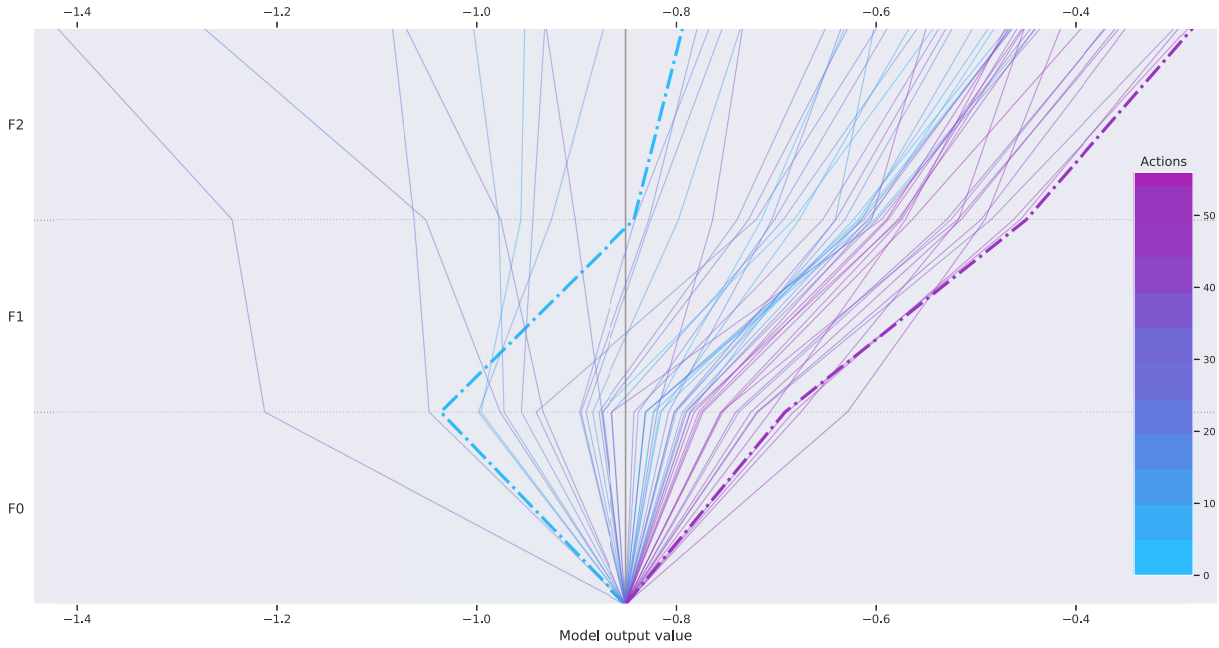
A.3 Resultados Preliminares da Explicabilidade

Nessa seção, o objetivo é investigar como cada feature do estado afeta o valor Q das ações. Para isso, como um estudo de caso, consideramos o estado representado pelas características descritas abaixo. Observamos que, como discutido na Seção 4.2, um estado é representado por uma tupla com três features, onde cada uma é referida como F0, F1 e F2.

- F0 equivale a 58 veículos (ou seja, 58 veículos estão nas filas da fase para a qual o agente atualmente selecionará a ação).
- F1 é igual a 6 veículos (ou seja, 6 veículos estão na fila de espera da próxima fase).
- F2 é igual a 19 veículos (ou seja, 19 veículos estão esperando nas filas das fases restantes).

Dado o estado descrito acima, o agente optou por escolher a ação 43, o que significa que a fase atual permanecerá aberta 43 segundos. A fim de entender porque nosso modelo selecionou essa ação, a Figura 34 apresenta o impacto de cada característica (eixo vertical) sobre todas as ações possíveis. Como visto, a figura tem 57 linhas com cores representando as diferentes ações, desde linhas azuis denotando ações próximas a 0 segundos até linhas roxas representando ações próximas a 56 segundos. O eixo horizontal indica o valor de Q das ações.

Figura 34: A contribuição de todas as features sobre as ações em um estado. Interpretar a figura de baixo para cima.



Fonte: Elaborada pelo autor.

Para entender como o modelo pode ser explicado, considere a linha tracejada roxa, que representa a ação de 43 segundos. A partir do valor base, esta linha está inclinada para a direita no nível F0, o que significa que F0 tem um impacto positivo sobre o valor Q da ação. No próximo nível (F1), a linha inclina-se ainda mais para a direita. E por fim, no nível (F2), a linha se mantém inclinada para a direita, mas menos acentuada.

O estado atual possui mais veículos na fase que vai abrir (58 veículos) do que a próxima fase (6 veículos) ou nas fases restantes (19 veículos). Observando a Figura 34, podemos extrair a seguinte informação: as ações que representam um tempo mais alto, também (em grande maioria) corresponde a recompensas maiores. Intuitivamente, isto significa que o agente, nesse estado, prefere escolher um tempo grande para evitar a longa fila que temos na fase atual.

Como outro exemplo, considere a linha tracejada azul, que representa a ação de 8 segundos. Como visto, a feature F0 tem um impacto negativo sobre esta ação, como evidenciado pelo comportamento de inclinação para a direita das características F1 e F2, essa ação não representa um valor ruim para todas as features. Partindo da discussão das duas linhas tracejadas (roxa e azul), 8 segundos aparenta ser uma janela de tempo pequena para reduzir suficientemente a fila na fase atual, e o impacto de F0 é tão negativo nessa ação, que mesmo com valores positivos das outras features, a ação continua com um baixo valor Q, quando comparado com outras ações. Não obstante, ao observar o comportamento desta linha como um todo, torna-se claro que esta não é uma boa ação dado o estado atual.

A.4 Discussão e Limitações

Apesar do modelo ter gerado bons resultados e ter ultrapassado em desempenho as linhas de base, o algoritmo FixedTime utilizado de comparação (com os diferentes tempos implementados: FT15, FT30 e FT45) serve apenas como uma comparação preliminar, considerando que é um dos algoritmos mais simples para o controle semafórico (TSC) e está longe de ser o estado da arte.

Além disso, na Figura 34 as ações para o estado atual com maior duração (em roxo) tendem a ser melhores para F0 do que ações com menor duração (em azul). Entretanto, quase todas as ações são impactadas positivamente pelas features F1 e F2. Podemos dizer que o modelo interpreta que quanto mais curto for a fase atual, melhor para as fases seguintes. Isto faz sentido dada a lógica do modelo, já que F1 e F2 correspondem às fases que têm que esperar o tempo decidido para a fase atual antes de realmente aliviarem a sua fila, e a recompensa está diretamente ligada ao tamanho da fila.

A partir disto, podemos observar como o modelo lida com as features para obter o melhor resultado possível. Seguindo a Figura 34, para otimizar este caso específico, teríamos que escolher uma ação representando uma duração suficientemente longa para diminuir as filas na fase atual, mas não excessivamente longa para que as filas em outras fases aumentem em demasia. Observando a figura, torna-se claro que esse comportamento é majoritariamente seguido pelo nosso modelo.

Entretanto, o fato de uma determinada característica ter um impacto positivo ou negativo sobre as ações, como descrito acima, não significa que ela terá um impacto igual em todos os estados. Na verdade, tudo o que podemos explicar é que quando estamos em um determinado estado, as features têm um impacto nas ações com base em seus valores, e com isso, podemos entender a importância dessa característica localmente. No entanto, no presente trabalho, não podemos justificar um aumento ou diminuição em outro estado pelo valor que a característica tem nesse estado.

Mesmo considerando que o modelo está convergindo e obtendo excelentes resultados nas métricas, pode-se identificar outliers na explicabilidade, e por consequência, no comportamento do agente. A linha mais à esquerda na Figura 34, corresponde à ação de 45 segundos, que tem seu valor próximo aos 43 segundos, ação escolhida pelo agente. Partido da ideia de que 2 segundos não deveriam ser o suficiente para ter uma disparidade entre uma ação ser a melhor, e outra ser a pior. Podemos considerar essa linha um outlier.