UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Gerenciamento de um cruzamento semaforizado utilizando Reinforcement Learning e Options Framework.

Dimitrius Guilherme Ferreira Borges

Itajubá, Dezembro de 2020

UNIVERSIDADE FEDERAL DE ITAJUBÁ - UNIFEI
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Dimitrius Guilherme Ferreira Borges

Gerenciamento de um cruzamento semaforizado utilizando Reinforcement Learning e Options Framework.

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação.

Área de Concentração: Matemática da Computação

Orientador: Prof. Dr. Edmilson Marmo Moreira
Coorientador: Prof. Dr. João Paulo Reus Rodrigues Leite

Dezembro de 2020
Itajubá

Agradecimentos

Agradeço aos meus pais, Maria e Marcos, por terem me dado toda a oportunidade e incentivo para que eu seguisse na carreira que escolhi. Por jamais terem impedido que eu fizesse as coisas que gosto, estas essenciais para que eu me tornasse o profissional e o estudante que sou hoje.

Agradeço a minha irmã, Rayssa, por ter me ajudado de forma sempre solícita e sincera nos momentos que precisei do auxílio durante o trabalho. Suas opiniões e dicas foram fundamentais para que eu pudesse desenvolver um trabalho tão científico como os dela.

Agradeço a minha esposa, Monique, por sempre ter me apoiado e incentivado nos momentos de desânimo, por ter sido compreensiva com as minhas ausências por conta do trabalho, por acreditar em mim mais do que eu jamais acreditei. Por ter aceitado que nos mudássemos de cidade de forma tão repentina e, com todo o seu otimismo, me mostrou o quanto a mudança foi bem vinda e catalisadora de nossos projetos, este incluso. Sem ela ao meu lado esse trabalho jamais seria o que é.

Agradeço aos meus orientadores, Prof. João Paulo e Prof. Edmilson, por terem sido extremamente solícitos e terem me dado tanta liberdade no desenvolver desse projeto. Por terem entendido minha necessidade de mudança de cidade e terem aceitado, de prontidão, que o trabalho fosse desenvolvido de forma remota, acreditando na minha capacidade e compromisso. Suas orientações, através de e-mails, mensagens, videoconferências ou pessoalmente, sempre foram momentos de grande instrução e me ensinaram como pensar e portar-me como um pesquisador. O resultado desse trabalho é tão de vocês quanto meu.

"Todas as coisas mudam em um ambiente dinâmico.
Seu esforço para manter-se no que você é, é o que te limita."
(Ghost In The Shell)

Resumo

O número de veículos nas ruas de todo o mundo tem crescido rapidamente ao longo da última década, impactando diretamente em como o tráfego urbano é gerenciado. O controle de cruzamentos sinalizados é um problema largamente conhecido e estudado e que, embora cada vez mais tecnologias sejam exploradas e aplicadas, ainda se encontram desafios e oportunidades ao tratar o problema, principalmente quando confronta-se a ineficiência dos já bem difundidos semáforos de tempos fixos, incapazes de lidar com eventos dinâmicos. O objetivo deste trabalho é aplicar *Hierarchical Reiforcement Learning* (HRL) ao controle de um cruzamento veicular semaforizado e, a partir dos resultados obtidos, compará-lo a um semáforo de tempos fixos dimensionado pelo Método de Webster. HRL é uma variação de *Reinforcement Learning* (RL), em que objetivos secundários, representados por sub-políticas, são propostos e organizados em um modelo hierárquico e gerenciados por uma política macro, responsável por selecioná-las quando se espera rendimento máximo das mesmas, sendo que tanto as sub-políticas quanto a principal são regidas pelo *framework Q-learning*. *Herarchical Reinforcement Learning* foi escolhido por aliar a capacidade de aprendizado e tomada de decisão feitos de acordo com observações do ambiente em tempo real, característicos do *Reinforcement Learning*, com um modelo similar ao Dividir para Conquistar, que desmembra o problema principal em sub-problemas. Isso traz ao modelo uma maior dinâmica e poder de adaptabilidade a um problema que exibe, por vezes, variações imprevisíveis, impossíveis de serem levadas em conta em abordagens determinísticas, como o Método de Webster. Os cenários de testes, formados por diversos tipos de fluxo de veículos, aplicados a um cruzamento de duas vias simples, foram construídos através da ferramenta de simulação SUMO. Os modelos HRL, suas sub-políticas isoladas e o Método de Webster são aplicados e avaliados a partir destes cenários onde, de acordo com os resultados obtidos, HRL se mostra superior tanto ao Método de Webster quanto às suas sub-políticas isoladas, mostrando-se uma alternativa simples e eficaz.

Palavras-chaves: Tráfego Veicular, Método Webster, Q-Learning, Hierarchical Reinforcement Learning, SUMO

Abstract

The number of vehicles on the streets across the world has quickly grown in the last decade, directly impacting how urban traffic is managed. The signalized junctions control is a vastly known and studied problem. Although an increasing number of technologies is explored and used to solve it, there still are challenges and opportunities to deal with it, especially when considering the inefficiency of the widely known fixed time traffic controllers, which are incapable of dealing with dynamic events. This study aims to apply Hierarchical Reinforcement Learning (HRL) on the control of a signalized vehicular junction and compare its performance with a fixed time traffic controller, configured using the Webster Method. HRL is a Reinforcement Learning (RL) variation, where secondary objectives, represented by sub-policies, are organized and proposed in a hierarchical model, managed by a macro-policy, responsible for selecting said sub-policies when those are capable of reaching its best results, where The Q-Learning Framework rules both sub and macro policies. Hierarchical Reinforcement Learning was chosen because it combines the ability to learn and make decisions while taking observations from the environment, in real-time, a typical ability from Reinforcement Learning, with a Divide to Conquer approach, where the problem is divided into sub-problems. These capabilities bring to a highly dynamic problem a more significant power of adaptability, which is impossible to be taken into account when using deterministic models like the Webster Method. The test scenarios, composed of several vehicle fluxes applied to a cross of two lanes, were built using the SUMO simulation tool. HRL, its sub-policies and the Webster Method are applied and assessed through these scenarios. According to the obtained results, HRL shows better results than the Webster Method and its isolated sub-policies, indicating a simple and efficient alternative.

Key-words: Vehicular Traffic, Webster Method, Q-Learning, Reinforcement Learning, SUMO

Lista de ilustrações

Figura 1 – Cadeia de Markov representando dinâmicas de clima. 19
Figura 2 – Diagrama de um PDM (SUTTON; BARTO, 1998) 21
Figura 3 – Processo Markoviano de um carrinho sobre trilhos (SUTTON; BARTO, 2018) 22
Figura 4 – Diagrama de estados do carrinho sobre trilhos. 23
Figura 5 – Um robô que deve navegar entre salas (SUTTON; PRECUP; SINGH, 1999) 32
Figura 6 – Fluxograma de um típico modelo de Hierarchical Reinforcement Learning (RIBAS-FERNANDES et al., 2011) 33
Figura 7 – Ambiente do robô com sub-objetivos (SUTTON; PRECUP; SINGH, 1999) 35
Figura 8 – Cruzamento simulado na ferramenta SUMO. a) Semáforo com entreverdes em Vermelho Total. b) Via A (Horizontal) em sinal verde. c) Via B (Vertical) em sinal verde. Os retângulos amarelos são os sensores Laço-Indutivos. 45
Figura 9 – Resultados de fluxo para TF em Fixo-1800 51
Figura 10 – Resultados de tempo de espera para TF em Fixo-1800 51
Figura 11 – Resultados de fluxo para TF em Pico-2700 52
Figura 12 – Resultados de tempo de espera para TF em Pico-2700 52
Figura 13 – Resultados de fluxo para TF em Pico-3600 53
Figura 14 – Resultados de tempo de espera para TF em Pico-3600 53
Figura 15 – Diagrama de estados e ações ao longo do tempo 56
Figura 16 – Relação entre políticas e sub-políticas do agente HRL 57
Figura 17 – Tradução do ambiente para estados do Q-Size 61
Figura 18 – Resultados de fluxo para Q-Size em Fixo-1800 63
Figura 19 – Resultados de tempo de espera para Q-Size em Fixo-1800 63
Figura 20 – Resultados de fluxo para Q-Size em Pico-2700 64
Figura 21 – Resultados de tempo de espera para Q-Size em Pico-2700 65
Figura 22 – Resultados de fluxo para Q-Size em Pico-3600. 65
Figura 23 – Resultados de tempo de espera para Q-Size em Pico-3600. 66
Figura 24 – Tradução do ambiente para estados do W-Size 68
Figura 25 – Resultados de fluxo para W-Size em Fixo-1800. 70
Figura 26 – Resultados de temo de espera para W-Size em Fixo-1800 70
Figura 27 – Resultados de fluxo para W-Size em Pico-2700 71
Figura 28 – Resultados de tempo de espera para W-Size em Pico-2700 72
Figura 29 – Resultados de fluxo para W-Size em Pico-3600 73

Figura 30 – Resultados de tempo de espera para W-Size em Pico-3600 73
Figura 31 – Tradução do ambiente para estados de H-Agent. 76
Figura 32 – Resultados de fluxo para H-Agent em Fixo-1800 79
Figura 33 – Resultados de tempo de espera para H-Agent em Fixo-1800 80
Figura 34 – Resultados de fluxo para H-Agent em Pico-2700 80
Figura 35 – Resultados de tempo de espera para H-Agent em Pico-2700 81
Figura 36 – Resultados de fluxo para H-Agent em Pico-3600 81
Figura 37 – Resultados de tempo de espera para H-Agent em Pico-3600 82
Figura 38 – Resultados de fluxo para H-Agent e TF em Pico-2700 83
Figura 39 – Resultados de tempo de espera para H-Agent e TF em Pico-2700 . . . 84
Figura 40 – Resultados de fluxo para H-Agent e TF em Pico-2700 até esvaziar o cruzamento 84
Figura 41 – Resultados de tempo de espera para H-Agent e TF em Pico-2700 até esvaziar o cruzamento 85
Figura 42 – Resultados de fluxo para H-Agent e TF em Pico-3600 86
Figura 43 – Resultados de tempo de espera para H-Agent e TF em Pico-3600 . . . 86
Figura 44 – Resultados de fluxo para H-Agent e TF em Pico-3600 até esvaziar o cruzamento 87
Figura 45 – Resultados de tempo de espera para H-Agent e TF em Pico-3600 até esvaziar o cruzamento 87

Lista de tabelas

Tabela 1 – Relação entre estado, ação e recompensa. 28
Tabela 2 – Relação entre largura de uma via e a vazão máxima suportada 40
Tabela 3 – Resultados para a aplicação do semáforo de tempo fixo nos cenários de teste 50
Tabela 4 – Resultados da primeira rodada de testes de hiperparâmetros de Q-Size. 61
Tabela 5 – Resultados da segunda rodada de testes de hiperparâmetros de Q-Size. 62
Tabela 6 – Resultados para a aplicação do agente Q-Size e TF aos cenários de teste 62
Tabela 7 – Resultados da primeira rodada de testes de hiperparâmetros de W-Size. 68
Tabela 8 – Resultados da segunda rodada de testes de hiperparâmetros de W-Size. 68
Tabela 9 – Resultados para a aplicação do agente W-Size, Q-Size e TF aos cenários de teste 69
Tabela 10 – Resultados de três categorias da primeira rodada de testes de hiperparâmetros de H-Agent. 77
Tabela 11 – Resultados da segunda rodada de testes de hiperparâmetros de H-Agent, ordenados por rel_h. 77
Tabela 12 – Resultados de três categorias da primeira rodada de testes de β para H-Agent. 78
Tabela 13 – Resultados da segunda rodada de testes de β para H-Agent, ordenados por rel_h. 78
Tabela 14 – Resultados para a aplicação do agente W-Size, Q-Size e TF aos cenários de teste 79

Lista de abreviaturas e siglas

Actor-Critic Adaptive Traffic Signal COntrollers	A-CAT
Actor-Critic	AC
Deep Learning	DL
Deep Q-Network	DQN
Deep Recurrent Q-Network	DRQN
Deep Reinforcement Learning	DRL
Hierarchical Reinforcement Learning	HRL
Multiagent Reinforcement Learning	MARL
Q-Learning	QL
Reinforcement Learning	RL
Simulation of Urban Mobility	SUMO
Temporal-Difference Learning	TD
Agente Inteligente	AI
Algoritmo Genético	GA
Cadeia de Markov	CM
Computação Evolutiva	CE
Dyna-GA	DGA
Função de Base Radial	RBF
Inteligência Artificial	IA
Inteligência de Enxame	IE
Processo Decisório de Markov	PDM
Processo Decisório Semi-Markov	PDSM
Propriedade de Markov	PM
Redes Neurais Artificiais	RNA
Redes Neurais Recorrentes	RNR

Lista de símbolos

A	Conjunto de ações de um processo.	21
A_s	Conjunto de ações disponíveis em um estado de um processo.	21
C_O	Tempo de ciclo ótimo para um semáforo de tempos fixos	38
C_{ef}	Tempo de ciclo efetivo de um semáforo.	40
E_q	Classificação das filas de acordo com a quantidade de veículos.	55
E_w	Classificação do tempo de espera médio das filas de acordo com o tempo de espera de cada veículo inserido nela.	55
E_{tf}	Estado do semáforo, de acordo com a cor atual de cada uma de suas fases.	55
F_n	Taxa máxima de veículos/hora suportada pela via sem que haja constrição.	39
G_n	Tempo de verde efetivo da fase n.	40
G_t	Retorno esperado a partir do estado em t.	23
Gr_n	Tempo de verde real da fase n.	40
I	Conjunto de estados $I \subseteq S$ que podem tomar uma *option* como uma ação.	34
O	Conjunto de *Options.*	54
$P(s', r\|s, a)$	Distribuição de probabilidade das transições de estado e retorno de recompensa, de acordo com a ação tomada no estado anterior.	21
P_a	Forma resumida de $P(s', r\|s, a)$.	21
P_n	Taxa de veículos/hora no pico da via.	39
$Q(s, a)$	Função Valor-Ação. Mapeamento entre um par estado e ação e o retorno esperado.	26
$Q(s, o)$	Função Valor-Ação que faz uso de *Options.*	74
$Q_*(s, a)$	Função Valor-Ação ótima.	26
R	Conjunto de recompensas retornadas em um determinado processo.	21
$R_a(s, s')$	Recompensa recebida ao executar a ação a no estado s e transitar para o estado s'.	21
R_{t+1}	Recompensa que um agente recebe ao realizar uma ação.	22
S	Conjunto de estados de um processo.	21
T_p	Tempo, em segundos, em que não há escoamento de veículos no cruzamento	39
T_{am}	Tempo de amarelo do semáforo.	39
T_{ev}	Tempo entreverdes.	39
T_{p_n}	Tempo, em segundos, em que não há escoamento de veículos na fase n	39

T_{pv}	Tempo, em segundos, em que não há escoamento de veículos durante a cor verde.	39
Tev_o	Tempo entreverdes entre a fase n e a fase que a antecede.	40
Tvd_o	Tempo de verde das fases opostas a fase n.	40
Tvm_n	Tempo de vermelho de uma fase, onde não há escoamento de veículos.	40
V'_f	Vazão de veículos após o término de uma ação ou *option* normalizado com Min-max.	74
V'_w	Tempo de espera após a término de uma ação ou *option* normalizado com Min-max.	74
$V_*(s)$	Função Valor-Estado ótima.	26
V_f	Vazão de veículos após o término de uma ação ou *option*.	74
V_m	Vazão média de veículos por minuto.	49
V_s	Total de veículos que foram atendidos em um cruzamentos.	49
V_w	Tempo de espera após a término de uma ação ou *option*.	74
$V_\pi(s)$	Função Valor-Estado. Mapeamento entre um estado s e o retorno esperado, de acordo com uma política π.	24
W	Tempo de espera médio dos veículos em um cruzamento.	50
Y	Taxa de ocupação crítica do cruzamento.	39
α	Taxa de aprendizagem.	27
β	Critério de parada de uma *Option*.	12
ϵ	Relação *exploit vs explore*.	30
ϵ-greedy	Política π que segue uma lógica de *exploit vs explore*.	30
γ	Fator de desconto.	24
μ	Política que seleciona *options* como ações.	34
π	Política. Mapeamento entre estados e ações, $\pi(a\|s)$.	24
π_*	Política ótima.	26
π_o	Sub-política e uma *option*.	36
σ	*Option*. Ação que pode durar mais de um instante de tempo.	33
avg_q	Média de veículos atendidos dentre as execuções.	12
avg_w	Média do tempo de espera dos veículos dentre as execuções.	12
k	Número de instantes de tempo discretos que a *option* demorou para atingir seu critério de parada β	35
max_q	Maior número de veículos atendidos dentre execuções.	59
max_w	Maior tempo de espera médio dos veículos dentre as execuções.	59
min_q	Menor número de veículos atendidos dentre execuções.	59
min_w	Menor tempo de espera médio dos veículos dentre as execuções.	59
$p(s'\|s, o)$	Probabiliade de a *option* o tomada no estado s se encerrar no estado s'.	36
r_s^o	Recompensa ao selecionar a *option* o no estado s.	36
rel_h	Relação entre avg_q e avg_w.	9

std_q	Desvio padrão de avg_q.	59
std_w	Desvio padrão de avg_w.	59
t	Instante de tempo em um processo.	20

Sumário

1 INTRODUÇÃO 16
1.1 Descrição Geral 16
1.2 Nova abordagem 16
1.3 Organização do trabalho 18

2 FUNDAMENTAÇÃO TEÓRICA 19
2.1 Processo Decisório de Markov 19
2.1.1 Cadeias de Markov 19
2.1.2 Ações e Recompensas 20
2.1.3 Política, Valor-Estado e Equação de Bellman 24
2.1.4 Valor-Ação 26
2.2 Q-learning 27
2.3 Hierarchical Reinforcement Learning 31
2.3.1 Options Framework 32
2.3.2 Algoritmo 37
2.4 Terminologias de um sistema de controle de trânsito 38
2.5 Método Webster para semáforos de tempo Fixo 38
2.6 Trabalhos Relacionados 40
2.7 Considerações Finais 43

3 EXPERIMENTOS E RESULTADOS 44
3.1 Simulador e Modelo de Cruzamento 44
3.1.1 Simulador - SUMO 44
3.1.2 O Cruzamento Sinalizado 44
3.2 Semáforo de Tempos Fixos 48
3.2.1 Resultados 49
3.3 Hierarchical Reinforcement Learning 54
3.3.1 Características gerais 54
3.3.1.1 Execuções de treino e teste 57
3.3.1.2 Configuração dos Hiperparâmetros 58
3.3.2 Sub-política Q-Size 59
3.3.2.1 Definições 59
3.3.2.2 Hiperparâmetros 60
3.3.2.3 Resultados 62
3.3.3 Política W-Size 66
3.3.3.1 Definições 66

3.3.3.2 Hiperparâmetros 67
3.3.3.3 Resultados 69
3.3.4 Política H-Agent 73
3.3.4.1 Definições 73
3.3.4.2 Hiperparâmetros 75
3.3.4.3 Parâmetro Beta 76
3.3.4.4 Resultados 77
3.4 Semáforo de Tempo Fixos e H-Agent 82

4 CONCLUSÃO 88
4.1 Considerações Finais e Trabalhos Futuros 89

REFERÊNCIAS 91

1 Introdução

1.1 Descrição Geral

A frota de veículos brasileira cresceu 21,5% entre 2012 e 2017 (DENATRAN, 2020), e esse crescimento leva a uma constrição dos cruzamentos semaforizados, especialmente nas grandes cidades. O excesso de veículos transforma cada cruzamento em um gargalo, diminuindo a vazão do tráfego e aumentando o tempo médio de espera na fila e a insatisfação dos motoristas. Esse fato é potencializado por uma característica dos semáforos atuais mais comuns: tempos de atuação fixos, incapazes de levar em conta a variação do tráfego, nas vias que compõem o cruzamento.

No entanto, o ecossistema de tráfego veicular é passível de ser descrito, através de dados que podem ser coletados e mapeados, e transpostos para modelos matemáticos. Por exemplo, as dinâmicas do fluxo de veículos nas vias podem ser modeladas através de Teoria de Filas (VANDAELE; WOENSEL; VERBRUGGEN, 2000; EKEOCHA; IHEBOM, 2018), Diagramas Petri (DIFEBBRARO; GIGLIO; SACCO, 2004) ou Teoria dos Grafos (RIEDEL; BRUNNER, 1994). Sendo assim, toda a gama de ferramentas associadas a tais teorias pode ser explorada para solucionar os problemas encontrados no trânsito urbano, especialmente aqueles relacionados aos cruzamentos sinalizados. Posto isto, tem-se, dentro das orientações de trânsito brasileiras, a já estabelecida Fórmula de Webster (DENATRAN, 1984), que tem por objetivo calcular os tempos da cor verde para cada uma das vias de um cruzamento, usando informações pertinentes, como fluxo de pico, vazão máxima da via e largura da rua, tendo como principal objetivo a redução, ou manutenção em valores baixos, do tempo de espera dos veículos envolvidos. No entanto, dimensionar os tempos de um cruzamento de forma definitiva, não leva em consideração uma característica intrínseca ao fluxo de veículos, a estocasticidade e variabilidade dos eventos. Sendo assim, semáforos com tempo fixo perdem desempenho de forma considerável ao lidar com eventos aleatórios ou desconhecidos no momento de suas definições e que exigem mudanças, mesmo que de forma momentânea, nas relações de fluxo e vazão, o que, invariavelmente, gera perda de desempenho.

1.2 Nova abordagem

Uma saída, para problemas com característica estocástica, é a aplicação de modelos de Inteligência Artificial (IA), com sua capacidade de identificar padrões e generalizar a partir da observação de dados históricos (GOODFELLOW; BENGIO; COURVILLE, 2016) e do próprio ambiente, aptos, portanto, a acomodar demandas variáveis, sem pre-

judicar o objetivo de manter o cruzamento com tempos de espera baixos.

O objetivo deste trabalho, portanto, é desenvolver um modelo de IA, do subcampo *Machine Learning*, utilizando agentes inteligentes que adquiram sua base de conhecimento a partir de dados históricos brutos (KUBAT, 2017), e sejam capazes de identificar padrões de trânsito de um cruzamento semaforizado, agindo de forma que melhor se adapte ao cenário e tomando decisões que favoreçam a diminuição do tempo de espera e o aumento do fluxo de veículos. Ainda, com base em estudos prévios (ZHAO; DAI; ZHANG, 2012; YAU et al., 2017), foi escolhido como ponto central o modelo baseado em *Reinforcement Learning* (RL) (SUTTON; BARTO, 2018), definido como *Hierarchical Reinforcement Learning* (HRL) (HENGST; SAMMUT; WEBB, 2010) através do *Options Framework* (SUTTON; BARTO, 1998), onde a abordagem assume aspectos de "Dividir para Conquistar".

Reinforcement Learning é, em linhas gerais, a terceira via dentro do *Machine Learning*, sendo as outras duas o *Supervised Learning* e o *Unsupervised Learning* (ERTEL, 2018). No caso tratado aqui, RL apresenta resultados mais promissores, primeiro por não ter a necessidade de conhecer antecipadamente o modelo do problema ao qual se quer atacar e, segundo, por ser capaz de implementar o modelo de aprendizagem *online* (SUTTON; BARTO, 2018), ou seja, angariar conhecimento para sua base já com o agente em operação, de forma bastante simples, tornando-o capaz de aprender mesmo quando posto a enfrentar situações que diferem muito daquelas observadas no momento do treino. Essas duas características, que não esgotam as vantagens mas as resumem, tornam os agentes de *Reinforcement Learning* ideais para o tratamento do problema de fluxo veicular, já que, justamente por ser estocástico, é virtualmente impossível delimitá-lo em um modelo imutável ou mesmo antecipar todas a situações com as quais o sistema poderá se deparar. Sendo assim, em resumo, o modelo foi escolhido pela compatibilidade com o problema observado.

IA e *Machine Learning* são áreas de estudo já conhecidas e exploradas para solução do gerenciamento de trânsito, com propostas que datam da década de 1980 (BIELLI et al., 1991). Contudo, não se tem conhecimento do uso de *Hierarchical Reinforcement Learning* e *Options Framework* na modelagem e tomada de decisões para cruzamentos veiculares sinalizados. A proposta desta variação de RL demonstra adequar-se ao problema descrito, já que acresce ao modelo básico a possibilidade de alterar o comportamento do agente de acordo com o que observa-se do ambiente em que está inserido. Esta abordagem, como será mostrado no decorrer desta dissertação, é capaz de suplantar a capacidade de controle do fluxo de um cruzamento semaforizado com tempos determinados tradicionalmente pela Fórmula de Webster.

1.3 Organização do trabalho

O trabalho foi dividido em quatro capítulos principais. O Capítulo 2 apresenta a base teórica do trabalho, indo dos alicerces de *Reinforcement Learning*, entre as seções 2.1 e 2.2, até *Hierarchical Reinforcement Learning* na seção 2.3, finalizando com o Método de Webster, na seção 2.5. No capítulo 3, são apresentados o modelo de simulação, na seção 3.1, os resultados observados, tanto para o semáforo com tempos fixos, na seção 3.2, calculados via Método de Webster, quanto para o Agente Inteligente com HRL, na seção 3.3, para finalmente serem comparados, na seção 3.4. Por fim, o Capítulo 4 apresenta as conclusões gerais e indica possibilidade de trabalhos futuros.

2 Fundamentação Teórica

2.1 Processo Decisório de Markov

2.1.1 Cadeias de Markov

Ao observar os mais variados fenômenos, sejam eles biológicos, climáticos, físicos, sociais, etc., e tentar descrevê-los de acordo com sua evolução e relações de causa e efeito, é comum obter uma representação em que estes são divididos em estados, isto é, um conjunto de características que os definem num dado instante de tempo. Estes podem se alterar a partir de algum estímulo, fazendo com que o fenômeno transite para um novo estado.

Por exemplo, ao se observar o clima, é possível definir seu estado como "nublado". A partir dele, caso as condições corretas sejam atingidas, pode começar a chover, passando, então, para o estado "chuvoso". Novamente, se certas condições ocorrerem, a chuva pode parar e voltar para o estado "nublado" ou, caso se esteja em uma cidade litorânea (e isso poderia ser incorporado no estado, como descrição), poderia passar direto para um estado "ensolarado" (Figura 1). Essas condições que devem ser atingidas para transitar de um estado A para um estado B (ou C ou D) não são determinísticas, isto é, possuem probabilidade de ocorrerem e causarem a transição. Sendo assim, se essa probabilidade de transição depender apenas do estado em que o fenômeno se encontra no momento da observação, alheio aos anteriores, e a transição entre os estados ocorrer de forma discreta, esse processo é chamado de Processo de Markov e a sequência de estados é conhecida por Cadeia de Markov (CM) (BOLDRINI, 1986).

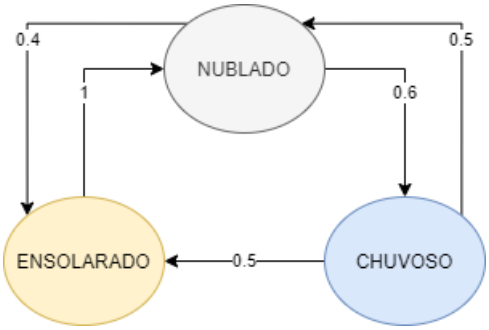

Figura 1 – Cadeia de Markov representando dinâmicas de clima.

Os Processos de Markov foram estabelecidos por Andrei Markov, em 1906 (MARKOV, 1907) e podem ser definidos, de acordo com Levin, Peres e Wilmer (2008), por:

$$P(X_{t+1} = i_{t+1} | X_0 = i_0, ..., X_t = i_t) = P(X_{t+1} = i_{t+1} | X_t = i_t) \tag{2.1}$$

Isto é, para determinar a probabilidade de um estado no instante t+1 ser igual a i_{t+1}, basta conhecer o estado no instante t, i_t. Essa característica é conhecida como Propriedade de Markov (PM) (LEVIN; PERES; WILMER, 2008).

Se houver total conhecimento das transições de probabilidade entre os estados e considerar-se que a mudança só ocorre no dia seguinte (tempo discreto), pode-se, então, estimar como estará o clima em X dias. Considerando que, para a Figura 1, o dia atual esteja nublado, três casos podem ser observados para o clima após dois dias:

- Caso 1: $nublado \rightarrow ensolarado \rightarrow nublado = 0.4 * 1 = 40\%$
- Caso 2: $nublado \rightarrow chuvoso \rightarrow nublado = 0.6 * 0.5 = 30\%$
- Caso 3: $nublado \rightarrow chuvoso \rightarrow ensolarado = 0.6 * 0.5 = 30\%$

Dessa forma, caso um processo seja passível de ser definido com uma CM, ganha-se poder de previsão sobre ele.

2.1.2 Ações e Recompensas

Ao considerar que em dados processos as transições na Cadeia de Markov não são completamente externas, mas sim determinadas por um "tomador de decisões" que possui um objetivo definido, é possível evoluir a noção de transição probabilística para a de ação tomada. Inclui-se no processo um dado agente que não é apenas um observador, mas que também é capaz de interagir com o ambiente, influenciando sua transição e promovendo ações que direcionam qual será o próximo estado dentro do processo, de acordo com seu interesse. Ainda, se cada uma dessas ações obedecer a uma função de *feedback*, que as associa a um valor que indica o quão boa (ou ruim) ela foi em relação ao objetivo do agente, transforma-se então essa Cadeia de Markov em um Processo Decisório de Markov (PDM) (BELLMAN, 1957b). Logo, como ilustrado pela Figura 2, obtem-se então um Agente, que observa o Ambiente, este definido através dos estados S_t, toma uma ação A_t, que gera uma recompensa R_t e faz com que o ambiente transite para um novo estado S_{t+1} e assim sucessivamente.

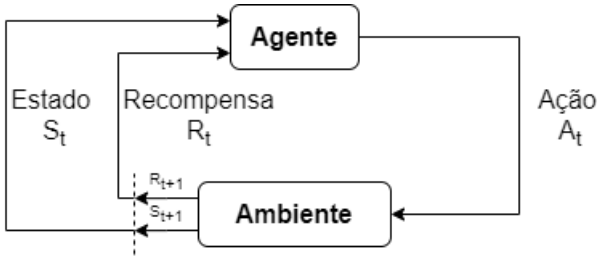

Figura 2 – Diagrama de um PDM (SUTTON; BARTO, 1998)

A Figura 2 expõe três conjuntos distintos:

- S: conjunto de estados que pertencem ao processo;
- A: conjunto de ações que pertencem ao processo. Pode também ser representado por A_s, referindo-se às ações que podem ser tomadas em um dado estado s;
- R: recompensas geradas dentro do processo. Também pode ser representado por $R_a(s, s')$, referindo-se à recompensa recebida ao tomar a ação a, no estado s e transitando para o estado s'.

Sendo assim, é possível definir o PDM através da tupla (S,A,R), sabendo que as distribuições de probabilidade de S e R, de acordo com a Propriedade de Markov, dependem apenas do estado atual e da ação tomada nele, para serem definidas. Ou seja, considerando um estado qualquer $s' \in S$ e uma recompensa $r \in R$, existe a probabilidade de ambos ocorrerem em um instante de tempo t, contanto que se saiba o estado e ação que o precederam, como demonstrado por Sutton e Barto (2018):

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_t = r|S_t = s, A_t = a\} \tag{2.2}$$

Assim, pode-se estender a tupla para (S, A, R_a, P_a), onde P_a, forma resumida de $P(s', r|s, a)$, é a distribuição de probabilidade das transições de estado e retorno de recompensa, de acordo com a ação tomada naquele estado.

Tanto nas CM quanto nos PDM, as probabilidades de transição indicam qual é a dinâmica do processo a partir de um estado observado. A única diferença é que, no caso do PDM, as transições são influenciadas por um agente ativo no sistema, então, para CM, $P(s, s')$ é a probabilidade de um estado s, em t, passar para s' em $t+1$, enquanto em um PDM, $Pa(s, s')$, é a probabilidade de um estado s, em t, passar para s' em $t+1$, caso o agente tenha tomado a ação a.

Observando a Figura 3 (SUTTON; BARTO, 2018), há um carrinho que se move para direita ou esquerda sobre um trilho, e, fixado a ele, um bastão, com uma articulação em sua base, que faz com que ele (o bastão) mova-se também para a direita ou para a

esquerda, de acordo com o movimento do carrinho. Modelando o caso como um PDM, tem-se como objetivo equilibrar o bastão sobre o carrinho o máximo de passos (ações) possíveis, sem deixar que ele caia completamente. As ações são os movimentos do carrinho, direita (D) e esquerda (E). Os estados são definidos pelo ângulo e direção de movimento do bastão, em relação ao carrinho, e as recompensas são +1 para caso a ação mantenha o bastão equilibrado e 0 caso o bastão caia. O agente, então, observa o estado do bastão e decide se deve mover o carrinho para a direita ou para a esquerda. Se o bastão continuar equilibrado, a ação retorna +1 para o agente, se não, +0. Considera-se, também, que o agente conhece o retorno imediato, R_{t+1}, para cada uma das ações possíveis para seu estado atual e que não há limite de passos.

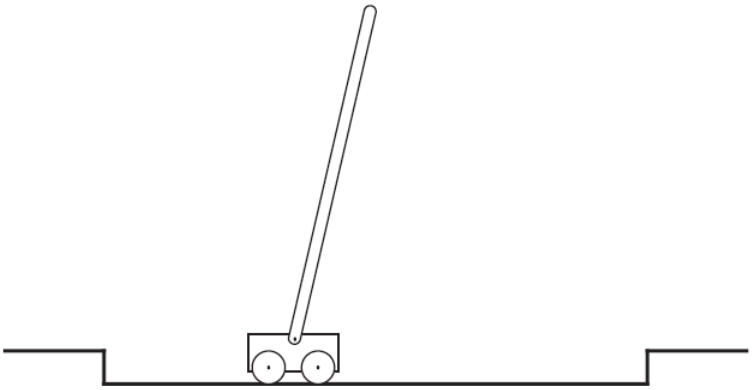

Figura 3 – Processo Markoviano de um carrinho sobre trilhos (SUTTON; BARTO, 2018)

Quando o agente escolhe a ação que quer tomar, baseando-se em R_{t+1}, ele garante apenas que o próximo estado manterá o bastão equilibrado. Mas, e no estado seguinte? Como o agente garante que essa ação tomada agora, que atinge seu objetivo nesse momento, não é, na verdade, uma última iteração antes de o bastão cair? Como foi observado com as CM, e isso estende-se aos PDM, esse tipo de modelagem nos permite calcular previsões dos estados futuros, baseando-se apenas no estado atual, caso se saiba as dinâmicas do processo. Sendo assim, é de interesse do agente saber, tomando como ponto de partida o estado em que ele está, quais são as recompensas futuras, obtidas em cada um dos próximos estados possíveis, como ilustra a Figura 4.

Agora, considerando que o agente só possui dois movimentos restantes, antes do processo terminar, qual a melhor ação a se tomar em $S1$? Com o diagrama em mãos, pode-se observar que a ação E trará uma recompensa maior, +2. No entanto, o agente, por enquanto, só tem visão de sua recompensa imediata e que, nesse caso, é +1, tanto para o caminho da Direita quanto para o da Esquerda, ou seja, em sua visão míope, os dois caminhos são iguais. Para resolver a questão, precisa-se dar a ele um indicativo do

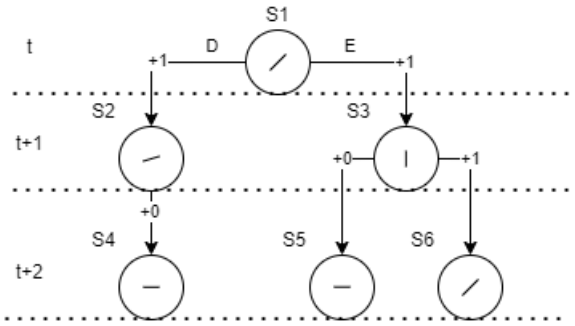

Figura 4 – Diagrama de estados do carrinho sobre trilhos.

que ele pode esperar de recompensa no próximo estado, de acordo com a ação tomada. Defini-se então, *retorno* (SUTTON; BARTO, 2018), G_t, em um PDM:

$$G_t = R_{t+1} + R_{t+2}... + R_T \tag{2.3}$$

$$G_t = \sum_{k=0}^{\infty} R_{t+k+1} \tag{2.4}$$

$$G_t = R_{t+1} + G_{t+1} \tag{2.5}$$

Logo, G_t, uma função naturalmente recursiva (2.5), é a recompensa final *esperada* ao acessar um dado estado.

Para o exemplo da Figura 4, S_2 tem uma recompensa esperada $G_{t,S_2} = 0$, já que só existe um caminho a ser seguido. Para G_{t,S_3}, deve-se considerar ambos os caminhos possíveis, porque não se sabe, antes de o agente estar em S_3 e tomar sua decisão, qual caminho ele irá escolher, sendo assim:

$G_{t,S_3} = 0.5 * 0 + 0.5 * 1 = 0.5$

Agora o agente passou a ter um indicativo não só do sucesso de sua ação imediata, mas também das suas consequências, baseado no estado a que ela garante acesso. Então, ao observar o retorno esperado dos dois possíveis estados, S_2 e S_3, 0 e 0.5, respectivamente, ele é capaz de escolher aquela que vai gerar maior retorno no longo prazo.

O exemplo proposto pela Figura 4 possui um limite estabelecido de dois passos, constituindo uma *tarefa episódica*. No entanto, o problema original estabelece que o agente deve manter o bastão equilibrado pelo máximo de passos possíveis, sem um limite, o que o torna uma *tarefa contínua*. Sendo assim, o melhor resultado possível é aquele com infinitos passos com o bastão equilibrado. Ao se calcular G_t em um processo contínuo, o valor tenderá ao infinito, tornando sua aplicação inútil. Em casos assim, propõe-se definir um horizonte para o retorno, isto é, limitar a visão do agente até certo ponto. Esse horizonte

é configurado por um *fator de desconto*, representado por $\gamma = [0, 1]$. Sendo assim, Sutton e Barto (2018) demonstram:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4}... \tag{2.6}$$

$$G_t = R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4}...) \tag{2.7}$$

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2.8}$$

$$G_t = R_{t+1} + \gamma(G_{t+1}) \tag{2.9}$$

O desconto é aplicado exponencialmente, em função do estado atual. Logo, quanto mais distante for a recompensa, menor a influência (ou importância) ela tem para o agente no instante t. Se $\gamma = 0$, apenas os valores de recompensa imediatos são considerados, se $\gamma = 1$, todos os valores, independente da distância t em relação a atual, são considerados.

2.1.3 Política, Valor-Estado e Equação de Bellman

Ao se mapear cada um dos estados com as respectivas probabilidades de se tomar cada uma das ações disponíveis e, ainda, considerar que essas são escolhidas, isto é, possuem maior ou menor probabilidade de serem tomadas, de acordo com o retorno que geram no longo prazo, pode-se dizer, então, que o agente passou a ter uma linha de pensamento que guia suas decisões e controla seu comportamento. Esse mapeamento é chamado de *política*, representado por π e definido por Bellman (1957b) e representado, de acordo com Sutton e Barto (1998), por:

$$\pi(a|s) = P\{A_t = a|S_t = s\} \tag{2.10}$$

Logo, π indica a probabilidade de se tomar a ação a no estado s.

Aliando π com G_t, da equação 2.8, expande-se o retorno esperado para $V_\pi(s)$:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] \tag{2.11}$$

$$V_\pi(s) = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{2.12}$$

Onde $V_\pi(s)$, conhecido por *função de Valor-Estado* (SUTTON; BARTO, 1998), representa o retorno esperado $\mathbb{E}$ em s ao seguir a política π.

Da mesma forma que G_t, $V_\pi(s)$ também é recursiva.

$$V_\pi(s) = R_{t+1} + \gamma V_\pi(s+1) \tag{2.13}$$

Sabe-se que o valor de cada estado, ou seja, seu retorno esperado, considera todas as ações $a \in A_s$, suas respectivas recompensas e probabilidades de serem tomadas. Sendo assim, Sutton e Barto (1998) definem:

$$V_\pi(s) = \sum_a \pi(a|s)[\sum_{s',r} p(s', r|s, a)(r + \gamma V_\pi(s'))] \tag{2.14}$$

que diz que o valor de um estado seguindo, uma dada política π, $V_\pi(s)$, é a média de todos os Valores-Estado e recompensas possíveis de serem assumidos, ponderados pela probabilidade da ação ser escolhida.

Caso cada ação leve a um único estado e gere uma única recompensa, pode-se resumir a equação 2.14 para:

$$V_\pi(s) = \sum_a \pi(a|s)[r + \gamma V_\pi(s')] \tag{2.15}$$

Ao se expressar $V_\pi(s)$ em função dos valores projetados para os possíveis estados futuros e estes, por sua vez, representados da mesma forma, forma-se então uma lógica recursiva, onde estados atuais são definidos por estados futuros, que também são definidos por estados futuros e assim por diante, demonstrando que 2.14 é, de fato, a *Equação de Bellman* para $V_\pi(s)$ (LAPAN, 2018).

Bellman (1952) estabeleceu a base da Programação Dinâmica ao demonstrar que um problema de otimização pode ser modelado através da relação da função de valor de um estado, num período t, com a função valor do estado seguinte, t+1, de forma recursiva. Essa relação é conhecida por Equação de Bellman. Solucioná-la, então, é resolver o problema de otimização. Para um PDM, resolvê-lo significa maximizar as recompensas recebidas, já que maximizá-las garante que o agente atingirá o objetivo da melhor maneira possível. O caminho para isso é garantir que as melhores ações, isto é, aquelas que retornam o maior $r + V(s')$, tenham maior probabilidade de serem tomadas. Logo, a política ideal é aquela que leva o agente a escolher mais frequentemente tais ações, logo, Sutton e Barto (1998) definem:

$$V_*(s) = \max_\pi v_\pi(s) \tag{2.16}$$

$$V_*(s) = \max_{a \in A_s}[\sum_{s',r} p(s', r|s, a)(r + \gamma V_*(s'))] \tag{2.17}$$

onde $V_*(s)$ é a função de Valor-Estado ótima e $\max_{a \in A}$ é a política ótima, π_*.

2.1.4 Valor-Ação

Na Equação 2.14, observa-se que o Valor-Ação $V_\pi(s)$, de um estado qualquer, nada mais é do que a soma das recompensas e do $V_\pi(s')$ de cada uma das ações $a \in A_s$, multiplicados por suas probabilidades $\pi(a|s)$. Pode-se enxergar cada uma dessas iterações como o *Valor-Ação* de cada ação, $Q(s,a)$, definido por Sutton e Barto (1998):

$$Q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)(r_{t+1} + \gamma V_\pi(s')) \tag{2.18}$$

Se $Q(s,a)$ é a iteração de cada ação, $V_\pi(s)$ pode ser representado por:

$$V_\pi(s) = \sum_a \pi(a|s) Q_\pi(s,a) \tag{2.19}$$

O mesmo para $V_*(s)$:

$$V_*(s) = \max_{a \in A_s} Q_{\pi_*}(s,a) \tag{2.20}$$

Se $Q(s,a)$ é definido em função de $V_\pi(s)$ e $V_\pi(s)$ pode ser definido em função de $Q(s,a)$, é possível então definir Valor-Ação em função dele mesmo. Para o caso ótimo, tem-se:

$$Q_*(s,a) = \sum_{s',r} p(s',r|s,a)(r + \gamma \max_{a' \in A_{s'}} Q(s',a')) \tag{2.21}$$

fazendo com que $Q_*(s,a)$ também seja uma Equação de Bellman.

Nota-se que a diferença entre $V_*(s)$ e $Q_*(s,a)$ é que, para o cálculo do primeiro, são consideradas todas as ações e suas respectivas probabilidades de serem tomadas, enquanto que, para o segundo, calcula-se apenas para a melhor ação possível, independente de sua probabilidade, ou seja, o Valor-Ação não exige que se conheça toda a dinâmica de cada estado para calcular seu respectivo valor. Outra maneira de diferenciá-los é entender que $V_*(s)$ dá a ideia de quão bom é estar em um estado ao seguir a política π, através de um valor esperado que não é, de fato, a recompensa a ser recebida, mas sim uma média ponderada de todos os retornos esperados possíveis. Já $Q_*(s,a)$ indica o quão bom é tomar a ação a no estado s, através da recompensa imediata e das recompensas futuras esperadas ao seguir continuar assumindo a ação ótima.

É importante diferenciá-las porque raramente se tem conhecimento de toda a dinâmica do modelo, isto é, as probabilidades de cada ação ser selecionada, sendo que o

processo para determiná-las pode ser excessivamente custoso ou virtualmente impossível. No presente estudo, o foco é justamente nos algoritmos baseados em $Q(s, a)$, conhecidos por *Q-Learning* (QL).

2.2 Q-learning

Watkins e Dayan (1992) estabeleceram que Q-learning (QL) é uma abordagem de *Reinforcement Learning* que permite que o agente aprenda uma política ótima ao experimentar as consequências de suas ações, sem a necessidade de conhecer toda dinâmica do problema abordado. O fato de não conhecer e não tentar estabelecer o modelo, torna QL uma abordagem *model-free*, oposta à *model-based*, que busca conhecer a dinâmica completa, isto é, as probabilidades de transição e destinos para cada estado e ação do modelo (SUTTON; BARTO, 2018).

Para se entender como QL funciona, é preciso estabelecer algumas definições. Primeiro, por uma questão de simplicidade, serão considerados apenas casos em que cada ação $a \in A_s$ leva a um único estado e gera uma única recompensa:

$$Q_*(s, a) = r + \gamma \max_{a' \in A_{s'}} Q(s', a') \tag{2.22}$$

Isto é, a equação 2.22 é um caso particular da equação 2.21, em que $P(s', r|s, a)$ é 1 para um par (s', r) qualquer ($s' \in S$ e $r \in R$) e 0 para todos os demais.

Também é introduzido o conceito de taxa de aprendizagem, $\alpha = [0, 1]$, indicativo do tamanho do passo, em direção a um novo valor, durante o aprendizado. Se $\alpha = 0$, todo valor novo aprendido é ignorado e apenas o valor já conhecido é levado em conta. Se $\alpha = 1$, o contrário acontece, todo novo valor substitui completamente o atual.

Com essas definições, Watkins e Dayan (1992) estabelecem a lógica central do *Q-learning*:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)) \tag{2.23}$$

Apesar dessa ser a definição formal, pode-se manipula-la para facilitar o entendimento:

$$Q(S_t, A_t) \leftarrow (1 - \alpha)Q(S_t, A_t) + \alpha(R + \gamma \max_a Q(S_{t+1}, a)) \tag{2.24}$$

O que a equação 2.24 demonstra é que o novo valor atribuído a $Q(S_t, A_t)$ é parte seu valor atual (primeiro membro do braço direito da equação) e parte o novo valor obtido para ele nessa iteração t (segundo membro). Sendo assim, o que o algoritmo faz é sempre

levar o Valor-Ação de cada par visitado em direção ao valor ótimo, aqui selecionado por $\max_a$, levando $Q(S_t, A_t)$ para $Q_*(S_t, A_t)$. Como max_a sempre retorna o valor ótimo conhecido, de acordo com o que o agente já explorou, quanto mais se conhece do processo, mais próximo ao valor ótimo $\max_a$ estará.

O algoritmo básico de QL pode ser visto no Algoritmo 1.

Algoritmo 1 Q-learning

0: **Função** ExecutarEpisodio
1: Inicializa Q(s,a) para todo $s \in S$, $a \in A_s$
2: **while** $S_t \neq$ objetivo ou $S_t \neq$ *estado-final* **do**
3: Seleciona A_t de A_{s_t}, seguindo a política π
4: Executa a ação A_t, observa R, S_{t+1}
5: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
6: $S \leftarrow S_{t+1}$
7: **end while**

A execução do algoritmo se dá até um objetivo ser alcançado ou até um estado final ser atingido (Linha 2). Esse tempo de execução é conhecido por *episódio*. É possível notar que há uma inicialização dos valores de $Q(s, a)$ (Linha 1) onde o que acontece, de fato, é uma atribuição de valores aleatórios a todos os pares Estado-Ação que existem no processo. No método tabular, utilizado em todo o estudo e definido mais a frente, essa associação de $Q(s, a)$ a valores se dá por uma tabela propriamente dita, como a Tabela 1.

Tabela 1 – Relação entre estado, ação e recompensa.

$Q(S_t, A_t)$	A_1	A_2	...	A_n
S_1	$R_{A_1}(S_1)$	$R_{A_2}(S_1)$	...	$R_{A_n}(S_1)$
S_2	$R_{A_1}(S_2)$	$R_{A_2}(S_2)$	...	$R_{A_n}(S_2)$
...	...	...	...	...
S_n	$R_{A_1}(S_n)$	$R_{A_2}(S_n)$	...	$R_{A_n}(S_n)$

Essa tabela é a representação da função Valor-Ação do modelo, e é através dela que a política π faz consulta dos valores esperados de $Q(s, a)$ e determina qual a melhor ação a ser tomada. Por usar uma tabela como aproximação/representação da função $Q(s, a)$, o algoritmo também é conhecido por *Tabular Q-learning* (SUTTON; BARTO, 2018).

Através das linhas 3 - 6, é possível definir como o algoritmo de fato funciona:

- Linha 3: O agente escolhe uma ação $A \in A_s$, seguindo a política atual.
- Linha 4: O agente executa a ação no ambiente, observa a recompensa imediata recebida e o estado S_{t+1} alcançado.

- Linha 5: A Equação de Bellman para $Q(s, a)$ é executada. O valor atual do par Estado-Ação que o agente se encontra é atualizado na tabela, respeitando a configuração do passo, α. A recompensa esperada do estado S_{t+1}, também consultada na tabela, é obtida utilizando uma política gulosa ($\max_a$).
- Linha 6: O agente passa então a analisar o estado acessado, S_{t+1}, reiniciando o processo.

É importante notar que a política utilizada para escolher a melhor ação para o estado S_t não é a mesma utilizada para estimar o valor de S_{t+1}. A primeira é a política que o agente busca melhorar, até chegar em π_*, a segunda é uma política gulosa, que escolhe sempre o melhor valor de $Q(S_{t+1}, a)$ dentre os conhecidos. O fato de *Q-learning* utilizar uma política de atualização diferente da de seleção, que é aquela que está sendo otimizada, o torna um método *off-policy* (SUTTON; BARTO, 1998). Métodos que utilizam a mesma política nesses dois momentos são conhecidos como *on-policy*, sendo SARSA (RUMMERY; NIRANJAN, 1994) o exemplo mais conhecido.

Como, através da Equação de Bellman, o agente aprende os valores utilizando estimativas de $Q(s, a)$ futuros e essas estimativas se alteram ao longo de passos sucessivos, isto é, o agente *aprende* ao longo de instantes t sucessivos, diz-se que QL é um modelo de *Temporal-Difference Learning* (TD) (SUTTON, 1988). Outro modelo seria o *Monte Carlo* (OTTERLO; WIERING, 2012), que, ao invés de usar a estimativa do próximo estado imediato, $Q(S_{t+1}, a)$, usa a estimativa final do episódio, G_t.

Obviamente, um número baixo de amostragens para cada par $Q(s, a)$ não é suficiente para estimar um valor confiável, menos ainda o ótimo. O mesmo pode se dizer para a política: um único episódio raramente é capaz de gerar a política ótima. A solução, então, é executar mais de um episódio, com a diferença de que, após o primeiro, a inicialização da tabela de Valor-Ação passa a ser com os valores obtidos no anterior.

No entanto, considerando que, no episódio 1, a tabela tenha sido toda iniciada com zeros. Levando-se em conta que no passo de escolha da ação (Linha 3), a escolha da estimativa de $Q(S_{t+1}, a)$ seja gulosa, ele vai, obviamente, escolher a ação que tiver o maior valor. Se todas forem zero, pode-se supor que a escolha é aleatória, mas, e num segundo momento? Quando todos os valores eram iguais, uma ação qualquer foi tomada e um valor autêntico, retornado pelo ambiente, foi obtido e armazenado, fazendo com que essa ação passe a ser a única com $Q(s, a) > 0$ e, de agora em diante, sempre será a escolhida, mas e se ela não for a ótima? Como fazer com que o algoritmo considere ações diferentes? Esse é o caso clássico do dilema aproveitar versus explorar (*exploit vs explore*), isto é, dar ao agente uma maneira de hora explorar novas ações, para, talvez, encontrar um caminho melhor, tirando proveito das ações que já conhece.

A maneira de se obter equilíbrio entre aproveitar e explorar é através de um outro parâmetro, o $\epsilon = [0, 1]$. Se ϵ= 0, apenas os valores já conhecidos são utilizados, e se $\epsilon = 1$, apenas valores desconhecidos são utilizados. Essa decisão se dá justamente no momento de escolher qual ação tomar no estado S_t. Se o agente possui uma maneira de pensar ao escolher uma ação, determinando se deve explorar em busca de novos pares (s, a) ou aproveitar daqueles que já conhece, ele tem, então, uma política π. Essa política é chamada de ϵ-greedy (LAPAN, 2018).

o Algoritmo 2 apresenta as novas definições incluídas no Algoritmo 1.

Algoritmo 2 Q-learning Tabular

0: **Função** ExecutarEpisodio n
1: **if** $n == 1$ **then**
2: Inicializa Q(s,a) para todo $s \in S$, $a \in A_s$ com 0
3: **else**
4: Inicializa $Q(s, a)$ com $Q(s, a)_{n-1}$
5: **end if**
6: **while** $S_t \neq$ objetivo ou $S_t \neq$ *estado-final* **do**
7: Seleciona A_t de A_{s_t}, seguindo a política ϵ-*greedy*
8: Executa a ação A_t, observa R, S_{t+1}
9: $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t))$
10: $S \leftarrow S_{t+1}$
11: **end while**

É importante observar que, como bem evidencia o *loop* principal, o agente aprende através de melhorias de sua política π, interagindo ativamente com o ambiente e esse aprendizado é disponibilizado imediatamente. Isso torna *Q-learning* um modelo de aprendizado *online* (BEN-DAVID; KUSHILEVITZ; MANSOUR, 1997), no qual, o agente aprende já inserido no ambiente (ou problema ou processo) que ele pretende otimizar, ganhando experiência a medida que novas situações vão se apresentando. Em contra-partida, modelos *offline*, de um modo geral, aprendem com um conjunto de dados fixos e conhecidos e, após essa fase, não são mais capazes de angariar conhecimento (BEN-DAVID; KUSHILEVITZ; MANSOUR, 1997).

No modelo tabular, as consultas feitas pela política (Linha 7) e as atualizações feitas pela Equação de Bellman (Linha 9) são, de fato, realizadas em uma tabela, representada por uma matriz ou um conjunto de *arrays*. No entanto, não é difícil de se imaginar que esse tipo de representação é custoso, já que ocupa memória e exige cada vez mais poder de processamento a medida que os pares s e a aumentam em quantidade. Esse problema é conhecido, tratando-se da "Maldição da Dimensionalidade" (do inglês, *Curse of Dimensionality*), formulado por Bellman (1957a), ainda nos primórdios da Programação Dinâmica. Em suma, o que a "maldição" expõe é que, quanto mais dimensões

o problema possui, isto é, quanto maior o número de variáveis que se relacionam, mais custosa é a representação e mais esparsos ficam os dados disponíveis. Traduzindo para Aprendizado por Reforço e QL, o problema aparece quando, por exemplo, tem-se um processo em que o número de estados é muito grande, como nos modelos que tratam do jogo de Xadrez, tornando virtualmente impossível a representação tabular. Em casos assim, pode-se utilizar *Deep Learning* (DL) (GOODFELLOW; BENGIO; COURVILLE, 2016), modelo de Aprendizado de Máquina que usa Redes Neurais Artificiais (HAYKIN, 2008; SKANSI, 2018) e *Representation Learning* (BENGIO; COURVILLE; VINCENT, 2013; GOODFELLOW; BENGIO; COURVILLE, 2016), com o intuito de aprender uma aproximação que substitua a tabela de $Q(s, a)$, ou seja, no lugar de uma tabela como a demonstrada na Tabela 1, usa-se uma Rede Neural Artificial, que recebe como entrada S_t e A_t e entrega em sua saída o valor de $Q(S_t, A_t)$. Esses modelos, que unem Aprendizado por Reforço e *Deep Learning*, são conhecidos por *Deep Reinforcement Learning* (DRL) (AGGARWAL, 2018) e os que usam especificamente a função Valor-Ação, *Deep Q-Network* (DQN) ou *Deep Q Learning* (LAPAN, 2018; RAVICHANDIRAN, 2018). Com isso, torna-se possível resolver problemas de grandes magnitudes e, de fato, algoritmos de grande notoriedade usam *Deep Learning* e *Deep Reinforcement Learning* como via principal. Por exemplo, o AlphaGo (SILVER et al., 2017), que ficou conhecido por derrotar o campeão mundial do jogo Go em 2016, e Mnih et al. (2013), que apresentou um agente capaz de jogar jogos do Atari.

2.3 Hierarchical Reinforcement Learning

O agente inserido em um PDM toma ações buscando maximizar a recompensa recebida e, ao maximizá-las, garante que atingirá seu objetivo. Imagine um robô que tem a missão de ir de um ponto em uma sala até outro ponto, em outra sala, tomando o menor caminho possível, sendo que a única ligação entre as salas é um pequeno corredor, como mostra a Figura 5, onde as células azuis representam os corredores e a verde o objetivo do robô. Considere que suas ações resumem-se a mover-se nas quatro direções possíveis e que sua recompensa seja +1 ao alcançar seu objetivo. Utilizando $Q(s, a)$, o robô aprenderá o caminho mais curto após um número finito de iterações com o ambiente, sendo que sua exploração se limitará a caminhar pelas células até, eventualmente, encontrar o ponto objetivo. A cada nova iteração, o robô encontrará seu objetivo mais rápido, até aprender a política ótima.

Observando o modelo proposto, é simples concluir que, antes de atingir seu objetivo principal, o agente (o robô) deve ser capaz de sair da sala em que ele se encontra e, depois, sair da sala seguinte, pra só ai atingir seu objetivo principal. No entanto, ele aprende apenas a atingir o ponto verde, já que esse é seu objetivo. Isso quer dizer que, embora ainda esteja na primeira sala, ele já está buscando o ponto verde, sem saber (e ele jamais

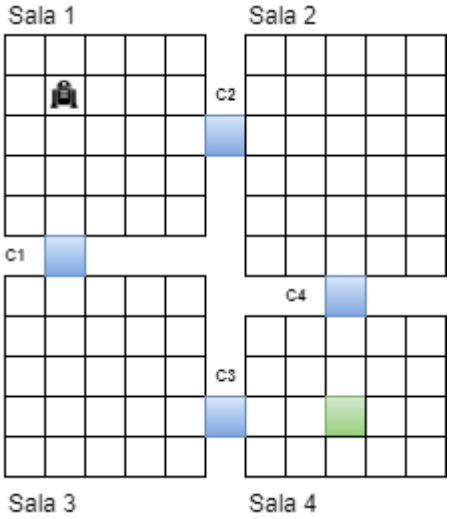

Figura 5 – Um robô que deve navegar entre salas (SUTTON; PRECUP; SINGH, 1999)

terá essa informação de forma explícita) que ele primeiro precisa sair da área em que se encontra. Ao descrever o problema, dizendo que o robô primeiro precisa sair da sala em que se encontra, depois, da sala seguinte para só aí chegar na área de seu objetivo e, então, atingi-lo, já se divide, com naturalidade, o problema em sub-problemas e é, justamente, essa a premissa do *Hierarchical Reinforcement Learning*.

Mais formalmente, diz-se que um modelo de *Hierarchical Reinforcement Learning* decompõe um problema de *Reinforcement Learning* em sub-problemas, ou sub-objetivos, onde o agente, que possui uma tarefa macro, assume sub-objetivos em momentos chave para facilitar e agilizar seu cumprimento (HENGST; SAMMUT; WEBB, 2010; BARTO; MAHADEVAN, 2003). Em termos mais próximos dos utilizados até aqui, o agente possui uma política, que busca atingir o objetivo principal e seleciona políticas parciais (ou sub-políticas), que buscam atingir objetivos locais que fazem parte do todo.

Há, na literatura, diversas abordagens de *Hierarchical Reinforcement Learning*, tais como *Feudal Learning* (DAYAN; HINTON, 1993), *MAXQ* (DIETTERICH, 1999) e *Options* (SUTTON; PRECUP; SINGH, 1999), a última sendo a principal ferramenta utilizada neste estudo e usada para exemplificar os conceitos a seguir.

2.3.1 Options Framework

O diagrama da Figura 6 ilustra como *Hierarchical Reinforcement Learning* funciona, onde tem-se:

- V: Valor-Estado da política principal;
- π: A política principal;
- a: ação comum;

- σ: ação que inicia uma sub-política;
- V_σ: Valor-Estado da sub-política;
- π_σ: sub-política.

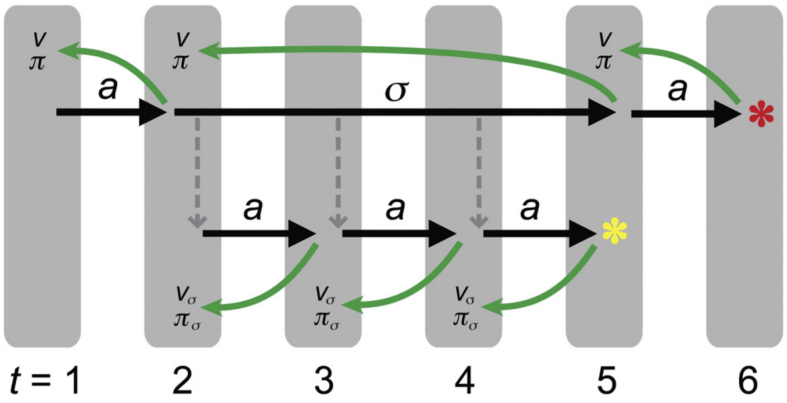

Figura 6 – Fluxograma de um típico modelo de Hierarchical Reinforcement Learning (RIBAS-FERNANDES et al., 2011)

As setas verdes representam quem recebe a atualização em seu Valor-Estado. V, π e a são exatamente como visto anteriormente. A primeira novidade, então, é o σ, um tipo diferente de ação que inicia uma busca por um sub-objetivo e pode durar mais de um instante t de tempo. Essa ação especial recebe o nome de *Option* (SUTTON; PRECUP; SINGH, 1999). Para garantir a diferenciação dos tipos de ação, também pode-se chamar as ações comuns de *ação primitiva*. Dessa maneira, as duas outras definições derivam da mesma ideia, V_σ reúne os Valores-Estado esperados para aquele sub-objetivo e π_σ é a política seguida pelo agente para concluir essa tarefa. A questão principal aqui é que, apesar de uma *Option* ter sido tomada em $t = 2$, para a política principal, é como se tivesse sido uma ação comum. A política *não tem conhecimento prévio de quantas ações (primitivas ou não) foram executadas dentro de* σ, tudo o que se sabe é que foi iniciado o processo para alcançar um sub-objetivo que, ao ser atingido, retorna um estado e as recompensas acumuladas ao longo da execução. O fato de se possuir um conjunto específico de Valor-Estado e política para o sub-objetivo só reforça a ideia de que essa busca é um processo separado do principal. É como se fossem dois agentes, um com a visão macro e um com a visão micro, o primeiro atribui tarefas a serem cumpridas pelo segundo, mas não as supervisiona, apenas espera pelo resultado.

É importante elucidar a necessidade de dois conjuntos de Valor-Estado. Como foi visto anteriormente, ao se falar de recompensas, na seção 2.1.2, os valores de *feedback* gerados são relacionados ao objetivo que se quer atingir, sendo que o mesmo vale para os estados: diferentes agentes, com objetivos variados, interpretam de forma diferente aquilo que o ambiente lhe apresenta. Logo, toda a relação de estado, ação e recompensa se altera

para cada caso e isso também vale para as diferentes camadas dentro *Hierarchical Reinforcement Learning*, já que, ao buscar um sub-objetivo, o agente pode mudar radicalmente de comportamento, gerando, então, diferentes conjuntos de recompensas.

Apesar de o exemplo mostrar apenas duas camadas, a da política principal π e a da sub-política π_σ, isso não implica que o problema não possa ser dividido em mais hierarquias. Mesmo sub-políticas podem selecionar uma *option* para alcançar um outro sub-objetivo.

Foi visto que em um PDM, após uma tomada de decisão, ocorre uma transição do estado presente para o próximo em tempo discreto, movendo-se de t para $t+1$, sendo que o tamanho desse salto é fixo, discreto e conhecido. No entanto, estabeleceu-se que, ao executar uma *option*, o tamanho do salto, no ponto de vista da política macro, é desconhecido no momento da tomada de decisão, podendo durar mais de um instante de tempo. Essa característica, de transições que duram $t = n$ instantes de tempo, é uma das que definem um Processo Decisório Semi-Markov (PDSM) (BRADTKE; DUFF, 1994), que possui seu próprio conjunto de funções para tudo o que foi definido para PDM. Ainda, quando o agente dá início a uma *option* em S_t, a tomada de ação neste estado não depende apenas dele, mas também da *option* (e de sua sub-política) que está sendo seguida. Isso conflita com a definição de Propriedade de Markov, que estabelece que a transição entre dois estados depende apenas S_t. Essa caraterística também faz com que a seleção de *options* seja um PDSM. No entanto, a partir de S_{t+1}, a seleção de ação volta a depender apenas do estado atual, já que sub-política já foi definida. Caso tais ações sejam primitivas, a execução da *option* passa a ser um PDM. Tais características fazem com que, no caso tratado pelo *Options Framework*, o modelo seja definido duplamente: quando se seleciona ações entre o conjunto de *options*, o processo é visto como um PDSM e, quando selecionando ações primitivas, seja dentro de uma sub-política ou na política principal, um PDM.

Sutton, Precup e Singh (1999) define *Options* pela tupla:

$$(I, \mu, \beta) \tag{2.25}$$

onde:

- I: é o conjunto de estados que podem tomar uma *option* como ação. Sendo que $I \subseteq S$
- μ: é a política seguida ao escolher entre as *options*
- β: é o critério de parada da *option*

Os dois primeiros itens, I e μ são representações diferentes de conceitos já conhecidos. Já β é algo novo a ser considerado. Quando um agente inicia uma *option*, ele espera que, em algum momento, haja um retorno e este é definido por β, que pode ser tanto um número de passos (ações primitivas tomadas) k, quanto o acesso a um estado específico. Ainda, é possível interromper uma *option* antes de um dos critérios de parada ser atingido, caso uma outra qualquer se mostre mais vantajosa, isto é, retorne um Valor-Ação ou Valor-Estado maior, de ser seguida.

Tomando, mais uma vez, o robô navegando em uma sala como exemplo e com os já estabelecidos sub-objetivos, de o agente sair da sala em que se encontra e, como cada sala possui duas saídas, tem-se então oito sub-rotinas, que podem ser traduzidas para oito *options*, numeradas de 1 a 8, como pode ser visto na Figura 7.

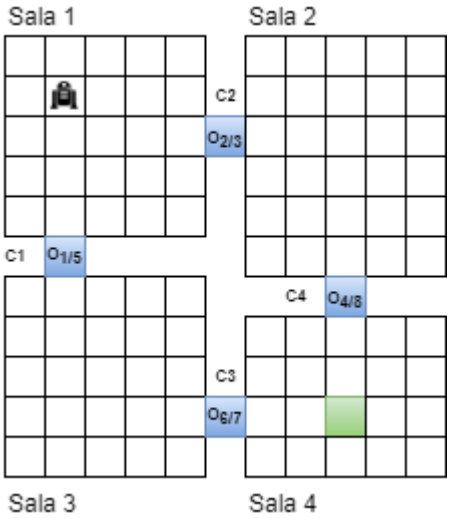

Figura 7 – Ambiente do robô com sub-objetivos (SUTTON; PRECUP; SINGH, 1999)

Usando a Sala 1 de exemplo, qualquer uma das posições (estados) que pertencem a ela podem tomar as *Options* O_1 e O_2 como ação, formando assim I_{O_1,O_2}. Ambas tem por objetivo alcançar os corredores $C1$ e $C2$, respectivamente, isto é, o critério β de cada uma é alcançar sua respectiva passagem. Agora o robô pode tanto tomar uma ação primitiva (cima, esquerda, direita, baixo), quanto assumir uma das duas *options*, o que guia sua decisão é justamente sua política e as relações de recompensa.

No ambiente da Figura 7, o objetivo principal do agente é ir do seu ponto inicial até o ponto final (célula verde) dando o menor número de passos possível. No entanto, ao selecionar uma *option*, o agente pode dar k passos e k pode ser não-ótimo, executando um número maior que o necessário. Para evitar isso, o agente deve recebe uma recompensa ao selecionar *option*, mas essa recompensa deve estar relacionada a k, para garantir que não só o sub-objetivo será cumprido, bem como o caminho tomado é vantajoso para o

objetivo principal, sendo assim, Sutton, Precup e Singh (1999) definem:

$$r_s^o = \mathbb{E}\{r_{t+1} + \gamma r_{t+2} + ... + \gamma^{k-1} r_{t+K} | \mathcal{E}(o, s, t)\} \tag{2.26}$$

onde k é o número de ações tomadas durante a execução da *option* e $\mathcal{E}(o, s, t)$ denota o evento da *option* o ter sido escolhida no estado s no tempo t.

O que a equação 2.26 diz é que a recompensa ao tomar uma *option* em s, r_s^o, é, então, a soma das recompensas descontadas geradas durante sua execução. Logo, o agente dará preferência a sub-políticas π_o que não só atingem seu sub-objetivo, mas o atinge sem prejudicar o objetivo principal.

No entanto, o número de passos dados é variável, isto é, uma mesma *option* vai demorar t instantes de tempo diferente para cada S_t em que for escolhida. Quanto mais perto do sub-objetivo o agente estiver, menos tempo de execução a *option* precisará. Ainda, quanto menos experiente a sub-política for, mais variável será k. Sendo assim, devemos considerar, de acordo com Sutton, Precup e Singh (1999), a probabilidade de o finalizar pra cada k passos:

$$p(s'|s, o) = \sum_{k=1}^{\infty} p(s', k)\gamma^k \tag{2.27}$$

onde $p(s'|s, o)$ é a probabilidade de a *option* o, tomada no estado s, terminar no estado s', já considerando o quão longe de s foi o término, sendo que essa distância é determinada por γ^k.

Com essas duas novas definições, é possível elaborar a Equação de Bellman para Valor-*Option* (SUTTON; PRECUP; SINGH, 1999):

$$Q_\mu(s, o) = r_s^o + \sum_{s'} p(s'|s, o) \sum_{o' \in O_{s'}} \mu(s', o') Q_\mu(s', o') \tag{2.28}$$

O Valor-*Option* é sua recompensa acumulada descontada, r_s^o, somada a média ponderada do Valor-*Option* de cada um dos estados futuros s', seguindo a política μ.

Para o caso ótimo, tem-se:

$$Q_O^* = r_s^o + \sum_{s'} p(s'|s, o) \max_{o' \in O_{s'}} Q_O^*(s', o') \tag{2.29}$$

Sutton, Precup e Singh (1999) também apresentam quais são as vantagens de usar *options* para buscar sub-objetivos, em comparação com o uso padrão de *Reinforcement Learning*. Um dos principais benefícios é o de diminuir o tempo necessário para se atingir o objetivo estabelecido. Isso ocorre porque, ao dividir o problema em sub-problemas, o

agente consegue direcionar seus esforços para aquilo que lhe traz melhores resultados mais rapidamente, já que, ao atingir um sub-objetivo, arquitetado para representar uma parcela mais tangível do problema, o agente garante que esta etapa já o direciona para sua meta principal, eliminando de seu processo a exploração de estados e ações que não trazem resultados relevantes.

2.3.2 Algoritmo

A Equação 2.29 é, na verdade, uma generalização das equações 2.21 e 2.22. Considerando que toda *Option* leva a apenas um estado s' (da mesma forma que foi feito em 2.22) e que sempre ocupa apenas um instante t para ser executada. Dessa maneira, $p(s'|s, o) = 1$ para apenas um estado s', com $k = 1$, e 0 para os demais. Logo, o somatório torna-se desnecessário, ficando:

$$Q^*_O = r^o_s + p(s', 1)\gamma^1 \max_{o' \in O_{s'}} Q^*_O(s', o') \tag{2.30}$$

Sabemos que $p(s', 1) = 1$, logo:

$$Q^*_O = r^o_s + \gamma \max_{o' \in O_{s'}} Q^*_O(s', o') \tag{2.31}$$

Como apenas um passo é tomado, $r^o_s = r_{t+1}$:

$$Q^*_O = r_{t+1} + \gamma \max_{o' \in O_{s'}} Q^*_O(s', o') \tag{2.32}$$

Chegando, então, a mesma estrutura da equação 2.22.

Sendo assim, a adaptação do algoritmo de *Q-learning* para o caso de *options* fica (SUTTON; PRECUP; SINGH, 1999):

$$Q(S_t, O_t) \leftarrow Q(S_t, O_t) + \alpha(r^o_s + \gamma^k \max_{o \in O_{S_{t+1}}} Q(S_{t+1}, o) - Q(S_t, O_t)) \tag{2.33}$$

Ou seja, uma generalização da equação já apresentada em 2.23. As novidades ficam então para a recompensa, r^o_s, que agora é acumulada e descontada ao longo da execução da *option*, e γ^k, que passa a descontar a distância em k passos entre o estado S_t e S_{t+1}. Levando, então ao Algoritmo 3.

Como comprovado nas equações 2.30, 2.31 e 2.32, as ações primitivas são um caso especial de *options*, onde k é sempre igual a 1. Sendo assim, se for considerado que O_s reúne tanto *options* quanto ações primitivas, o algoritmo da Figura **??** passa a ser aplicável para os dois tipos de ações e, portanto, compatível com *Q-learning* clássico e com o *Options Framework*. Essa será, a representação do algoritmo considerada neste estudo.

Algoritmo 3 Q-learning - Options Framework

Função ExecutarEpisodio n
if $n == 1$ **then**
  Inicializa Q(s,o) para todo $s \in S$, $o \in O_s$ com 0
else
  Inicializa $Q(s, o)$ para todo $Q(s, o)_{n-1}$
end if
while $S_t \neq$ objetivo ou $S_t \neq$ *estado-final* **do**
  Seleciona O_t de O_I, seguindo a política ϵ-*greedy*
  Executa a *option* O_t, observa r_s^o, k e S_{t+1}
  $Q(S_t, O_t) \leftarrow Q(S_t, O_t) + \alpha(r_s^o + \gamma^k \max_{o \in O_{S_{t+1}}} Q(S_{t+1}, o) - Q(S_t, O_t))$
  $S \leftarrow S_{t+1}$
end while

2.4 Terminologias de um sistema de controle de trânsito

O Departamento Nacional de Trânsito brasileiro (Denatran) é quem padroniza os termos descritivos de um semáforo e de sistemas de controle de trânsito (DENATRAN, 1984), de acordo com normas internas e internacionais. Dentre eles, os termos mais utilizados nesse trabalho são:

- *Fase*: Execução completa de todas as cores (verde, amarelo e vermelho) de um único semáforo;
- *Ciclo*: Tempo total, em segundos, para a sequência completa de fases em um cruzamento semaforizado;
- *Entreverdes*: Período entre o fim do verde de uma fase e o início do verde de outra;
- *Vermelho Total*: Período em que todas as fases ficam no estado vermelho;
- *Faixa de retenção*: Faixa indicativa de onde o veículo deve parar ao aproximar-se de um semáforo;

2.5 Método Webster para semáforos de tempo Fixo

Seguindo as recomendações do Manual de Semáforos Brasileiro (DENATRAN, 1984), o cálculo dos tempos de verde de cada fase de um semáforo é feito através do Método de Webster, proposto por Webster e Cobbe (1966). O método busca calcular o tempo de verde efetivo total e de ciclo ótimo, C_O, que é aquele que mantém o fluxo do cruzamento de forma a gerar um tempo de espera mínimo para os veículos, e dividi-lo entre as fases do semáforo, de modo que o tempo seja condizente com o fluxo de cada via.

Denatran (1984) define C_O por:

$$C_O = \frac{1,5 \cdot T_p + 5}{1 - Y} \tag{2.34}$$

O tempo perdido total T_p é dado pelo somatório dos tempos perdidos de cada fase:

$$T_p = \sum^{n} T_{p_n} \tag{2.35}$$

sendo n o número de fases do cruzamento em questão e T_{p_n} o tempo perdido de cada fase:

$$T_{p_n} = T_{pv} + T_{ev} - T_{am} \tag{2.36}$$

onde:

- T_{pv}: Tempo em que, mesmo com a cor verde acesa, não há escoamento de veículos;
- T_{ev}: Tempo entreverdes, composto pela soma do tempo de amarelo com o tempo de Vermelho Total;
- T_{am}: Tempo de amarelo.

A taxa de ocupação crítica do cruzamento, Y, é dada pela soma das taxas de ocupação crítica de cada fase:

$$Y = \sum^{n} y_n \tag{2.37}$$

sendo esta a relação entre o fluxo de veículos no pico e a vazão máxima suportada pela via:

$$y_n = \frac{P_n}{F_n} \tag{2.38}$$

onde:

- P_n: Taxa de veículos/hora no pico de fluxo da via;
- F_n: Taxa máxima de vazão veículos/hora suportada pela via sem que haja constrição.

A vazão máxima de veículos em uma dada via é sempre considerada ideal, isto é, a taxa de vazão é a mesma da taxa de entrada. Já F_n é diretamente relacionada com a largura da via, sendo que seus valores estão na Tabela 2 (DENATRAN, 1984).

Tabela 2 – Relação entre largura de uma via e a vazão máxima suportada

Largura (m)	**3**	**3.3**	**3.6**	**3.9**	**4.2**	**4.5**	**4.8**	**5.2**
Vazão (v/h)	1850	1875	1900	1950	2075	2250	2475	2700

O tempo de verde efetivo de cada fase é dado por:

$$G_n = \frac{y_n}{Y} \cdot (C_O - T_p) \tag{2.39}$$

E o tempo de verde real por:

$$Gr_n = G_n + T_{pv_n} - T_{am} \tag{2.40}$$

A diferença entre G_n e Gr_n é que o primeiro considera apenas o tempo de verde estritamente necessário para o fluxo de veículos proporcional para cada via, enquanto o segundo estende o tempo de verde com o valor de tempo perdido esperado, indicando então o que de fato será configurado no semáforo.

Logo, o ciclo efetivo de um semáforo C_{ef} é dado por:

$$C_{ef} = \sum^{n} Gr_n + T_{ev} \tag{2.41}$$

Tem-se também o tempo de vermelho de uma fase, Tvm_n, dado por:

$$Tvm_n = Tvd_o + Tev_o \tag{2.42}$$

onde:

- Tvm_n: Tempo de vermelho da fase n
- Tvd_o: Tempo de verde das fases oposta a fase n
- Tev_o: Tempo entreverdes entre a fase n e a fase antecessora a ela

Sendo assim, C_{ef} também pode ser dado por:

$$C_{ef} = \sum^{n} Tvm_n \tag{2.43}$$

2.6 Trabalhos Relacionados

Dessa maneira, explorando as variadas técnicas mencionadas anteriormente, é possível observar o uso de Computação Evolutiva (CE) e Inteligência de Enxame (IE) no

trabalho de Tian et al. (2018), que faz uso de Otimização Por Enxame de Partículas, aplicada a múltiplas interseções conectadas entre si, cada uma possuindo oito etapas de abertura/fechamento de vias. A seleção de qual via abrir, em cada cruzamento, é considerada uma partícula do sistema, enquanto o complexo de cruzamentos e suas seleções são considerados uma rede de partículas de enxame onde, além de cada partícula possuir uma função correspondente, elas também relacionam-se entre si. Logo, o objetivo global a ser otimizado, melhorar a eficiência do tráfego de todo o sistema, é composto, de fato, pela solução dos objetivos individuais, porém não isolados, devido a interação entre as partículas. O algoritmo proposto é posto em teste em cenários variados, que vão de 1600 a 4800 veículos por hora aplicados a todo o complexo de cruzamentos, onde observou-se que, em casos de fluxo baixo, o semáforo de tempos fixos possui capacidade de gerenciar os cruzamentos mas, quando fluxos mais severos são aplicados, a partir de 4000 veículos por hora, o algoritmo proposto otimiza o tamanho máximo das filas em 26%.

Tan et al. (2018) vão além, ao combinar *Q-Learning* com Algoritmos Genéticos, utilizado o primeiro para extrair os parâmetros do modelo observado de forma *on-line*, isto é, enquanto o processo ocorre e são aplicados a função-objetivo do segundo, sendo que a combinação proposta é conhecida por Dyna-GA (DGA). DGA procura estabelecer tanto o tempo de ciclo do semáforo quanto o tempo de verde de cada fase, sendo que a avaliação de adequação de cada cromossomo é feita através do modelo dinâmico levantado por QL. Mais do que isso, os autores propõem, para a via arterial estudada, onde uma via principal é cruzada por diversas vias secundárias, usar agentes locais em cada um dos cruzamentos, sob responsabilidade de um agente macro, chamado de Supervisor, para garantir que não só o objetivo local mas também o principal sejam atingidos, criando então uma hierarquia entre os agentes. O desempenho do modelo proposto é comparado com um Algoritmo Genético (GA) clássico, sendo que o DGA atingiu tempos de espera 30.8% menores que GA.

O uso de Redes Neurais Artificiais (RNA) pode ser observado principalmente em modelos que aplicam *Deep Reinforcement Learning* (ou *Deep Q-Network*), caso em que RNAs são utilizadas para gerar aproximação da função $Q(s, a)$ (LAPAN, 2018). A proposta de Li Yisheng Lv (2016) aborda justamente essa premissa, usando uma RNN arquitetada como um *Deep Stacked Autoencoder* para estimar a função QL, criando uma *Deep Q-Network*, sendo que a entrada da rede são os estados do cruzamento e a saída são os valores $Q(s, a)$ para cada ação possível. O modelo é aplicado a um cruzamento simples, de duas fases, onde cada via possui duas pistas em ambos o sentidos, totalizando oito pistas. O tamanho da fila em cada uma delas nos últimos quatro instantes t de tempo formam a entrada da RNA, sendo que as saídas são manter a via corrente aberta ou inverter, respeitando um tempo mínimo de verde de 15 segundos. Os resultados são comparados a um modelo que utiliza *Reinforcement Learning* clássico e demonstraram que é possível reduzir em até 14% o tempo de espera. Zeng, Hu e Zhang (2018) também trazem tal

combinação para o modelo, com uma especialização de RNA, Redes Neurais Recorrentes (RNR), tornando o modelo então uma *Deep Recurrent Q-Network* (DRQN). A justificava para o uso é que, ao lidar com um ambiente dinâmico e não estacionário, o agente deve não só considerar a observação atual, mas também o histórico de observações. Tal histórico pode ser abordado através de um modelo baseado em memória e este faz uso de RNR para incluí-los, em especial a que aplica *Long Short-Term Memory*. A entrada da DRQN é formada pelas matrizes de densidade e velocidade do cruzamento e as ações, mais uma vez, são apenas estender o tempo de verde da via corrente ou deixar que o cruzamento passe para a via seguinte. O algoritmo proposto é comparado tanto com um semáforo de tempos fixos quanto com uma DQN, todos aplicados a um cruzamento de duas vias, sendo que ambas possuem quatro pistas em ambos os sentidos, totalizando quatro fases. Embora os resultados finais da DRQN e DQN sejam próximos, a primeira apresenta uma estabilidade e taxa de convergência melhores.

Yau et al. (2017) também nos trazem um *overview* do uso de IA no controle de tráfego, focado nos algoritmos de *Reinforcement Learning*, dividindo em seis categorias: *Multiagent Reinforcement Learning* (MARL), que abrange aqueles que aplicam agentes com capacidade de trocar informações entre si, num sistema complexo e tomam decisões que levam em conta o todo; *Max-plus RL*, agentes orientados a valores *payoff* na seleção de suas ações; *Model-based*, a exemplo do SARSA, onde o agente constrói um modelo do ambiente; *Actor-Critic* (AC), onde o agente é dividido em duas partes, o crítico, que estima o quão perto do objetivo o estado atual está, e o ator, que seleciona a ação, sendo que após cada ação tomada, o crítico avalia o novo estado acessado e determina se as novas condições são melhores ou piores do que o esperado; *Multistep Backup RL*, que considera os valores obtidos em episódios completos ou de uma sequência de instantes de tempo, análogo ao que faz Monte-Carlo; e, por fim, RL com Função de Aproximação, ou seja, aqueles tratados via DQN. Destes, podemos destacar dois, devido a sua complexidade e capacidade, sendo estes o MARL e AC.

El-Tantawy, Abdulhai e Abdelgawad (2013) fazem uso da abordagem MARL, num modelo batizado de MARLIN-ATSC, onde cada agente, responsável por controlar um cruzamento em específico, considera não só o estado deste, mas também os estados e ações dos outros agentes dentro do sistema. A comparação de desempenho é feita entre um sistema em que os agentes comunicam-se entre si, identificado por MARLIN, outro em que não há comunicação entre os agentes, mas estes aplicam MARL para casos isolados, intitulado MARL-I e um caso base, de acrônimo BC, representativo do conjunto de técnicas utilizadas num conjunto de vias da área central de Toronto - Canadá, vias estas emuladas para uso dos modelos propostos. Sendo assim, MARLIN apresentou melhora de 37% no tempo de espera, quando comparado a BC, e 14,41% quando comparado ao MARL-I.

O uso de *Actor-Critic* pode ser observado no trabalho de Aslani, Mesgari e Wiering

(2017), que emprega *Actor-Critic* para Estados Contínuos num modelo intitulado *Actor-Critic Adaptive Traffic Signal COntrollers* (A-CAT), onde, para a função de mapeamento Valor-Estado, é empregado tanto *Tile Coding*, método em que os estados são divididos em partições, quanto Função de Base Radial (RBF), usada para calcular o grau de pertencimento dos estados e ambos são comparados. O modelo é aplicado a uma área com 50 cruzamentos, onde vários cenários são avaliados. Para análise dos resultados, três principais parâmetros são observados: Tempo de viagem (sec/km), Emissão de Gás Carbônico e Combustível gasto, sendo que, para o cenário mais completo, onde são considerados, entre outras coisas, pedestres, área de estacionamento e incidentes no tráfego, RBF alcançou os melhores resultados em todos os parâmetros. Ainda, A-CAT também é comparado tanto com *Q-Learning* Tabular quanto com *Q-Learning* Bayesiano (DEARDEN; FRIEDMAN; RUSSELL, 1998), apresentando resultados melhores no tempo de viagem para o o cenário representando o caso mais básico, onde não há qualquer interferência no fluxo de veículos.

2.7 Considerações Finais

Como destacado por Bielli et al. (1991) e Zhao, Dai e Zhang (2012), Inteligência Artificial é um caminho largamente trilhado ao tentar se resolver o problema imposto pelos cruzamentos de vias sinalizados. Ainda, Zhao, Dai e Zhang (2012) apresentam as principais abordagens adotadas, citando, para o caso de gerenciamento de cruzamentos urbanos, Sistemas Fuzzy, Redes Neurais Artificiais, Computação Evolutiva e Inteligência de Enxame, *Reinforcement Learning* e Teoria dos Jogos. Por fim, Yau et al. (2017) indica que *Reinforcement Learning*, tanto sozinho quanto aliado a outras técnicas, é uma das abordagens mais difundidas e exploradas dentro do problema estudado. Aliando-se a isso a oferta cada vez maior de sistemas computacionais dedicados a IA, como o NVIDIA Jetson Nano (NVIDIA, 2019), computador de pequeno porte direcionado para sistemas embarcados, ou como o TensorFlow (TENSORFLOW, 2019), biblioteca completa para serviços na nuvem destinados a desenvolvimento e treino de IAs, dão a oportunidade de se trazer ainda mais desempenho aos modelos RL, aplicando técnicas de *Deep Reinforcement Learning*, *Actor-Critic*, etc, sem aumento demasiado dos custos do projeto, abrindo então caminho para que essas técnicas ganhem ainda mais popularidade. No entanto, embora a riqueza de modelos seja expressiva, não foi encontrado qualquer trabalho que faça uso de *Hierarchical Reinforcement Learning*, através do *Options Framework*, no controle de um cruzamento semaforizado, identificado então uma possível lacuna de abordagem, o que sustenta a justificativa de explorá-la, então, neste trabalho.

3 Experimento e Resultados

3.1 Simulador e Modelo de Cruzamento

3.1.1 Simulador - SUMO

Para reproduzir um cruzamento semaforizado, foi utilizada a ferramenta de simulação *open-source Simulation of Urban Mobility* (SUMO) (LOPEZ et al., 2018; SUMO, 2020a), capaz de simular tráfego urbano microscópico (riqueza de detalhes) e contínuo em larga escala, permitindo a inclusão de diferentes tipos de veículos, pedestres, semáforos, detectores de veículos, etc. A interface de acesso utilizada foi a TraCI (SUMO, 2020b), em sua versão Python. O simulador é desenvolvido e mantido por colaboradores do *Institute of Transportation Systems*, que faz parte do *German Aerospace Center* e é licenciada sob a EPL 2.0. Seu uso é bem difundido entre trabalhos com foco em gerenciamento de tráfego veicular e a exemplo disto tem-se os trabalhos de Zeng, Hu e Zhang (2018), Castro, Hirakawa e Martini (2017) e Jin e Ma (2015), onde todos usam a ferramenta como meio de simulação, embora cada um explore uma abordagem diferente: Zeng, Hu e Zhang (2018) aplicam *Deep Recurrent Q-learning* e dados em tempo real de GPS para controlar o semáforo, já Castro, Hirakawa e Martini (2017) aplicam Redes Neurais para o controle e comportamento do sistema bem como as fases do semáforo, enquanto Jin e Ma (2015) aplicam tanto *Q-learing* quanto SARSA em um cruzamento e fazem comparações de desempenho através dos dados coletados no simulador.

3.1.2 O Cruzamento Sinalizado

Como o intuito do trabalho é comparar o desempenho de um Agente Inteligente (AI) que aplica *Hierarchical Reinforcement Learning* ao de um semáforo de tempos fixos, com relação ao fluxo total de veículos e seus tempos médios de espera em um cruzamento, optou-se pela elaboração de um cenário que, embora simples, traz a oportunidade de se observar tais valores com a segurança de que não existam variáveis ocultas que os alteram, e é representativo de casos reais frequentes de cruzamentos semaforizados. O cenário utilizado é um modelo de dois sentidos, com mão dupla, totalizando quatro vias, sem conversão, com pista simples e duas fases, uma para a via vertical e outra para a horizontal (Figura 8). Tal composição engloba o caso mais comum de cruzamento de vias com semáforo, que ocorre tanto em cidades de pequeno quanto de grande porte e, embora cenários mais complexos sejam alvos naturais de soluções robustas, casos mais simples por vezes passam despercebidos, ainda que possam gerar impactos severos no trânsito como um todo, já que fazem parte do sistema e podem, em muitos casos, ser os únicos cenários

possíveis, quando limita-se o ambiente observado a cidades pequenas.

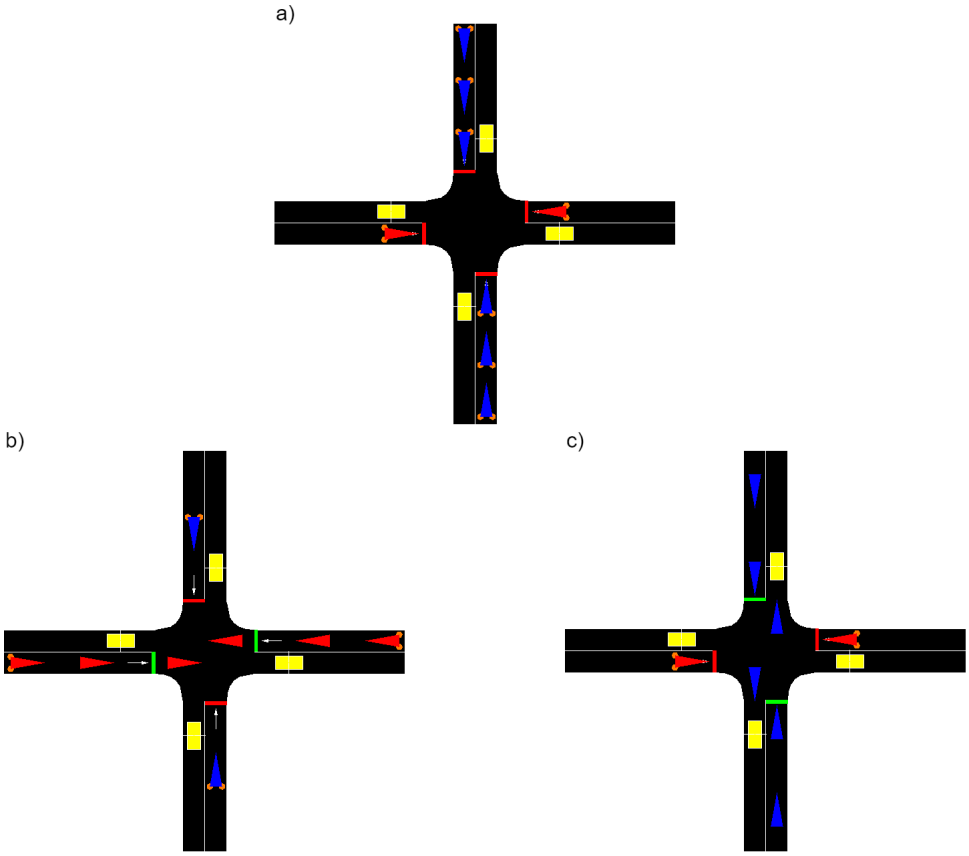

Figura 8 – Cruzamento simulado na ferramenta SUMO. a) Semáforo com entreverdes em Vermelho Total. b) Via A (Horizontal) em sinal verde. c) Via B (Vertical) em sinal verde. Os retângulos amarelos são os sensores Laço-Indutivos.

Casos mais complexos poderiam, por exemplo, considerar um sistema de múltiplas vias nos dois sentidos, onde a diferença seria que se observaria fluxos de entrada e saída maiores para todos os modelos, o que, na verdade, apenas inflaria os valores. Também poderiam ser permitidas conversões, isto é, dar liberdade aos veículos de convergirem a direita ou a esquerda no cruzamento. Entretanto, conversões não teriam impacto no estudo proposto, uma vez que, no momento em que um veículo faz a travessia do cruzamento, ele é tratado como atendido, independente de seu sentido. Aqui é possível citar um caso especial que poderia gerar variações de fluxo: quando houvesse uma demanda excessiva nas vias após o cruzamento, gerando uma fila que chegasse até a saída deste ou, num caso extremo, criando um *deadlock*, que é quando há veículos parados no cruzamento das pistas, impedindo que o trânsito circule. No entanto, como o cenário foi limitado ao de um cruzamento isolado, os efeitos nos semáforos e vias posteriores é irrelevante.

As vias possuem 3,2 metros de largura, 250 metros de comprimento e velocidade máxima de 50 km/h. Tanto a largura da via quanto a velocidade máxima são padrões da ferramenta. Os veículos gerados pelo simulador são todos do tipo *passenger*, que repre-

sentam carros de passeio e possuem as seguintes características:

- Comprimento: 5m;
- Espaçamento entre veículos: 2.5m;
- Aceleração máxima: 2.6 m/s ^{2};
- Desaceleração máxima: 4.5 m/s ^{2};
- Velocidade máxima: 11,11 m/s (40 km/h).

Com exceção da velocidade máxima, ajustada de acordo com o Manual de Semáforos (DENATRAN, 1984), todos os demais valores são padrão da ferramenta. As filas de veículos foram delimitadas através de sensores do tipo Laço-Indutivo, capazes de captar a passagem de um objeto metálico em sua área de ativação. Sendo assim, o detector que identifica um veículo entrando na fila foi posicionado de forma que caibam 25 veículos entre ele e a Faixa de Retenção, totalizando 188 metros. O sensor que identifica a saída de um veículo da fila foi posicionado logo após o fim da travessia do cruzamento.

A simulação de tráfego foi delineada de acordo as limitações do Semáforo de Tempos Fixos. Embora tais sistemas sejam capazes de assumir diferentes configurações de acordo com a data e hora locais (ele pode assumir, por exemplo, um conjunto de tempos de verde para suas fases entre as 8 da manhã e as 17 horas e, então, trocar sua configuração, para suprir horários de pico), TF continua restrito a cenários em que há previsibilidade, em que o operador que o configura sabe de antemão que a partir das 17 horas, por exemplo, haverá um pico de demanda e, então, é capaz de configurar mudanças de plano. Porém, caso a mudança de demanda seja aleatória ou inesperada, o operador não possui meios de antevê-la e o TF torna-se incapaz de alterar sua configuração para adequar-se a nova demanda. Sendo assim, foram elaborados dois cenários de estresse, Pico-2700 e Pico-3600, que possuem surtos de demanda de curta duração, e que buscam simular tais mudanças repentinas, isto é, picos que não seriam levados em conta no momento de calcular os tempos de cada fase para TF.

A duração da simulação é de uma hora, para que haja tempo de se observar os impactos dos surtos de demanda e da gerência do semáforo no ambiente simulado.

O fluxo de veículos gerado no simulador é controlado através de um fator veículo/hora (v/h), por via, e pode ser tanto determinístico, que garante igualdade entre N execuções, quanto probabilístico, obedecendo uma distribuição de Poisson (LARSON, 2015). No entanto, todos os cenários utilizados neste trabalha fizeram uso apenas dos geradores determinísticos, já que assim há garantia de total igualdade nos números e instantes de tempo dos veículos gerados entre N execuções e, caso seja observada qualquer variância entre elas, a responsabilidade será do agente que controla o semáforo.

Além dos cenários Pico-2700 e Pico-3600, dois outros foram delineados: Fixo-1800, que representa o caso base de fluxo esperado, onde não há saturação da via nem mudança brusca de demanda e foi utilizado para realizar os cálculos de tempo para TF; e Fixo-3600, proposto com o intuito de saturar o cruzamento, a ponto de haver constrição, de forma alternada, e utilizado no treino do agente HRL.

- **Fixo-1800**: 1800 veículos gerados em uma hora, dividido igualmente entre todas as vias
- **Fixo-3600 (exclusivo para treino do AI)**: 3600 veículos gerados em uma hora, dividido em 3 etapas:
    1. 30 v/m em cada via, por 20 minutos;
    2. Vias horizontais com 6 v/m cada, vias verticais com 24 v/m cada, por 20 minutos;
    3. Vias horizontais com 24 v/m cada, vias verticais com 6 v/m cada, por 20 minutos;
- **Pico-2700**: 2700 veículos gerados em uma hora, dividido em 6 etapas, sendo:
    1. 15 v/m em cada via, por 20 minutos;
    2. Vias horizontais com 7,5 v/m cada, vias verticais com 15 v/m cada, por 5 minutos;
    3. Vias horizontais com 7,5 v/m cada, vias verticais com 45 v/m cada, por 5 minutos;
    4. 15 v/m em cada via, por 20 minutos;
    5. Vias horizontais com 15 v/m cada, vias verticais com 7,5 v/m cada, por 5 minutos;
    6. Vias horizontais com 45 v/m cada, vias verticais com 7,5 v/m cada, por 5 minutos;
- **Pico-3600**: 3600 veículos gerados em uma hora, dividido em 6 etapas, sendo:
    1. 15 v/m em cada via, por 20 minutos;
    2. Vias horizontais com 7,5 v/m cada, vias verticais com 45 v/m cada, por 5 minutos;
    3. Vias horizontais com 7,5 v/m cada, vias verticais com 60 v/m cada, por 5 minutos;
    4. 15 v/m em cada via, por 20 minutos;

5. Vias horizontais com 45 v/m cada, via vertical com 7,5 v/m cada, por 5 minutos;
6. Vias horizontais com 60 v/m cada, via vertical com 7,5 v/m cada, por 5 minutos;

3.2 Semáforo de Tempos Fixos

Para se calcular os tempos para o Semáforo de Tempos Fixos (TF), é necessário fazer algumas considerações. Primeiro, para fins de teste, os cenários Pico-2700 e Pico-3600 são classificados, do ponto de vista do TF, com um fluxo de 1800 v/h, independente dos períodos de pico e fluxo total. Isto é feito porque, apesar dos períodos de pico serem bem delimitados no ponto de vista de projeto, eles buscam simular surtos de fluxo que não foram previstos na configuração do TF.

O tempo de amarelo, para vias em que os veículos circulam numa velocidade entre 30 e 40 km/h, é convencionalmente configurado em três segundos (DENATRAN, 1984), bem como tempo de Vermelho Total, configurado para dois segundos, completando, assim, um tempo entreverdes T_{ev}= 5. O Manual dos Semáforos (DENATRAN, 1984) estabelece que o tempo de um ciclo deve limitar-se entre 30 e 120 segundos. Sendo assim, qualquer valor fora do intervalo é ajustado. O manual também estabelece que o tempo mínimo de verde não deve ser menor que 10 segundos. Ainda, dado que o cenário proposto para simulação possui valores ideais, sem variação de perfil de condução, o tempo de verde perdido, T_{pv}, foi estabelecido em dois segundos. Todos os valores de tempo obtidos foram arredondados para o inteiro divisível por cinco superior.

Também deve-se considerar:

- Nº de Fases n: Duas fases, uma para a via vertical e uma para a via horizontal.
- Vazão Máxima: Foi adotado, de acordo com a Tabela 2, a vazão máxima de 1875 v/h, referente a vias de largura de 3,3 metros, valor mais próximo a largura da via simulada, de 3,2 metros;
- Fluxo: 1800 v/h em todo o cruzamento e divididos igualmente, logo, 900 v/h para via horizontal e 900 v/h vertical. No entanto, tal fluxo é dividido pelos dois sentidos de cada via (Vertical: Norte-Sul(NS), Sul-Norte(SN) e Horizontal: Oeste-Leste(OL), Leste-Oeste(LO)) e, como ambos sentidos são sempre abertos simultaneamente (NS com SN e OL com LO), o fluxo de pico real para cada via é de 450 v/h.

Sendo assim, o Fluxo de Saturação Crítica, Y, fica:

$$Y = \sum_{n}^{2} y_n = \frac{450}{1875} + \frac{450}{1875} = 0,24 + 0,24 = 0,48$$

Para o tempo perdido total, tem-se:

$$T_p = \sum_{n}^{2} T_{p_n} = (2 + 5 - 3) + (2 + 5 - 3) = 8(s)$$

Sendo assim, para C_O:

$$C_O = \frac{1,5 \cdot 8 + 5}{1 - 0,48} = 32,69 \approx 35(s)$$

Como a distribuição de demanda entre as vias é igual, os cálculos para ambas as fases retornarão os mesmos resultados, sendo assim, para o tempo de verde efetivo G_n, tem-se:

$$G_1 = G_2 = \frac{0,24}{0,48} \cdot (35 - 8) = 13,5 \approx 15(s)$$

E para o tempo de verde real, Gr_n:

$$Gr_1 = Gr_2 = 15 + 2 - 3 = 14 \approx 15(s)$$

O tempo de vermelho de ambas as fases também é igual:

$$Tvm_1 = Tvm_2 = 15 + 5 = 20(s)$$

O tempo de ciclo efetivo, C_{ef}, fica:

$$C_{ef} = (15 + 5) + (15 + 5) = 40(s)$$

TF fica, então, com duas fases, uma para via vertical, outra para a horizontal, cada uma com 15 segundos de verde, três de amarelo e dois de vermelho, levando a um ciclo (soma de tempo de todas as fases) de 40 segundos.

3.2.1 Resultados

Os resultados foram divididos em quatro indicadores:

- V_s: Total de veículos que foram atendidos pelo semáforo do cruzamento
- V_m: Vazão média de veículos por minuto

- W: Tempo de espera médio, em segundos, dos veículos que passaram pelo cruzamento

Como o simulador garante igualdade entre execuções que fazem uso do gerador determinístico e TF possui um comportamento estritamente fixo, sem qualquer possibilidade de alternância, todas as execuções do cenário geram os mesmos resultados. Por este motivo, apenas uma rodada foi necessária para se obter os resultados com TF.

A Tabela 3 reúne os resultados observados.

Tabela 3 – Resultados para a aplicação do semáforo de tempo fixo nos cenários de teste

Cenário	V_s	V_m	W
Fixo-1800	1784	29,73	16,63
Pico-2700	2349	39,15	29,34
Pico-3600	2420	40,33	31,53

Ao se analisar os resultados para o cenário Fixo-1800, é possível observar que o Método de Webster é capaz de entregar uma configuração que gera um tempo de espera, com média de 16,63 segundos, muito próximo do tempo de verde, que é de 15 segundos, o que implica que o tempo configurado é suficiente, na maioria das vezes, para atender todos os veículos presentes na via. Também é possível ver que a vazão aproxima-se muito da absoluta, com 29,73 v/m, contra 30 v/m (15 v/m por via) de fluxo de entrada, um rendimento de 99,1%. As Figuras 9 e 10 apresentam, respectivamente, os resultados de vazão e tempo de espera minuto a minuto. O formato em onda triangular se dá porque o tempo de cada via é de 20 segundos (15 de verde e 5 de entreverdes), cabendo então 3 fases em cada minuto, somado ao fato de que o TF sempre atende as vias de forma alternada. Sendo assim, no primeiro minuto ele atende, por exemplo, na ordem V-H-V e no minuto seguinte, H-V-H, se uma das duas vias possui um fluxo maior, quando ela for atendida com mais frequência no minuto, o fluxo/minuto observado também será maior. Porém, a alternância de vias para o tempo de espera é benéfica, como o fluxo de entrada é aquele usado durante os cálculos, cada via é aberta no momento em que há uma quantidade de veículos condizente com o tempo de verde a ser acionado.

Embora o total de veículos que entram no cruzamento seja, entre os cenários Pico-2700 e Pico-3600, 33% maior, o número total de veículos atendidos aumenta em apenas 3%. Ao se aliar esse dado com o fato de que a vazão manteve-se praticamente a mesma, é possível concluir que o Método de Webster atingiu seu limite de desempenho já no Pico-2700 e, independente do volume de veículos que possa tentar acessar o cruzamento, a vazão deverá se alterar muito pouco, enquanto o tempo de espera continua a degradar-se.

A Figura 11 traz os resultados para o cenário Pico-2700 e já é possível notar como o Método de Webster perde efetividade quando ocorrem surtos não previstos no momento de calcular os tempos do semáforo. A vazão de veículos aumenta a partir dos 20 minutos,

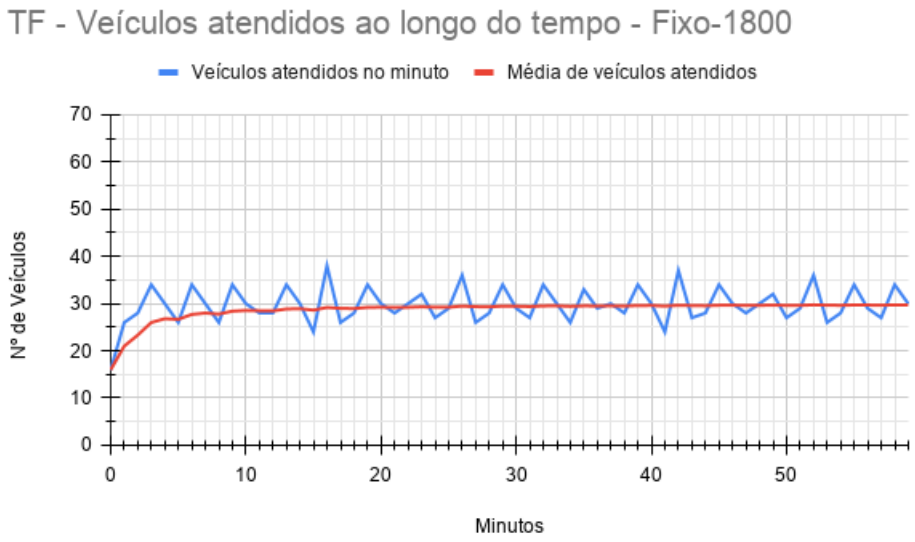

Figura 9 – Resultados de fluxo para TF em Fixo-1800

Figura 10 – Resultados de tempo de espera para TF em Fixo-1800

momento do primeiro surto, saindo de picos de vazão anteriores na casa dos 35 veículos, e indo para próximo de 50, no minuto 25. Embora, como descrito no cenário, o fluxo de entrada total nos surtos seja de 45 v/m nos primeiros 5 minutos e 105 veículos nos 5 seguintes, a média de fluxo de saída move-se lentamente, indo de 29 v/m, aos 20 minutos, para 33 v/m aos 30 e, aos 49 minutos, já registra 37,12 v/m, mesmo antes do surto final. É possível observar também que, mesmo após o fim do primeiro surto, aos 30 minutos, a vazão/minuto mantém-se alta, indicando que o TF não foi capaz de atender essa primeira demanda, que acumulou-se no cruzamento e, mais tarde, se combina com o segundo surto que, apesar de ter os mesmos valores de fluxo do primeiro, gera picos de vazão de saída mais altos, como o registado no minuto 51, com 59 v/m.

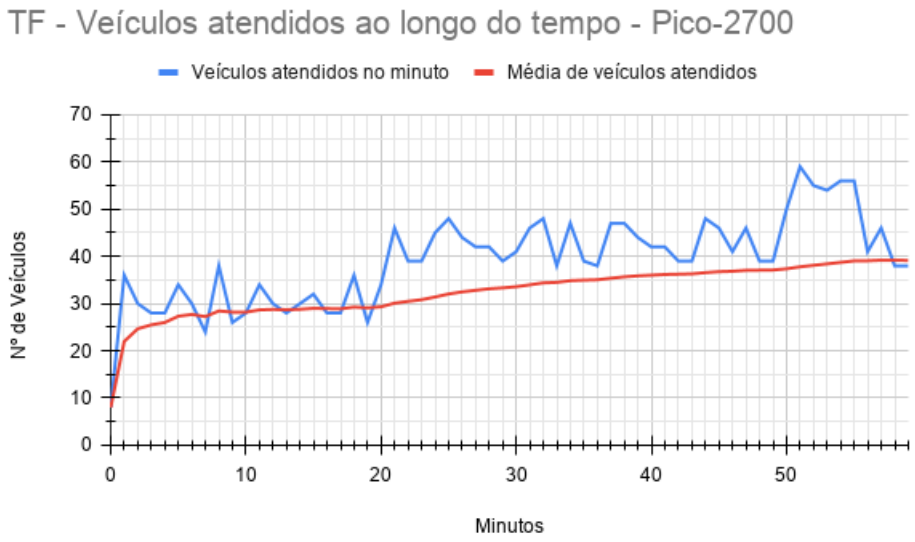

Figura 11 – Resultados de fluxo para TF em Pico-2700

Ao se analisar o tempo de espera, ilustrado na Figura 12, fica ainda mais evidente como o surto influencia no desempenho de TF. No momento do primeiro aumento, ocorre um salto no tempo de espera, que antes chegou a um máximo de 23 segundos, no minuto 7, para 40 segundos, a partir do minuto 27, um aumento de 74%.

Tal impacto considerável é reflexo dos tempos fixos, incapazes de adaptar-se a novas demandas, que não foram previstas no momento de seu planejamento, levando os veículos a ficaram parados por mais de um ciclo. Mais uma vez, o salto de demanda gera consequências mesmo após o seu término e, quando o sistema dá indícios de que normalizaria seus valores, o novo surto ocorre, degradando mais uma vez o desempenho.

Figura 12 – Resultados de tempo de espera para TF em Pico-2700

O cenário Pico-3600, com seus resultados apresentados nas Figuras 13 e 14, traz valores semelhantes aos observados no Pico-2700, reforçando que o TF já havia atingido seu ponto máximo de rendimento. As principais diferenças são que não há indícios de que o TF conseguiria normalizar a vazão no futuro próximo e o tempo de espera que, para o Pico-2700, atingiu 40 segundos no minuto 27 e aqui alcança o mesmo valor quatro minutos antes.

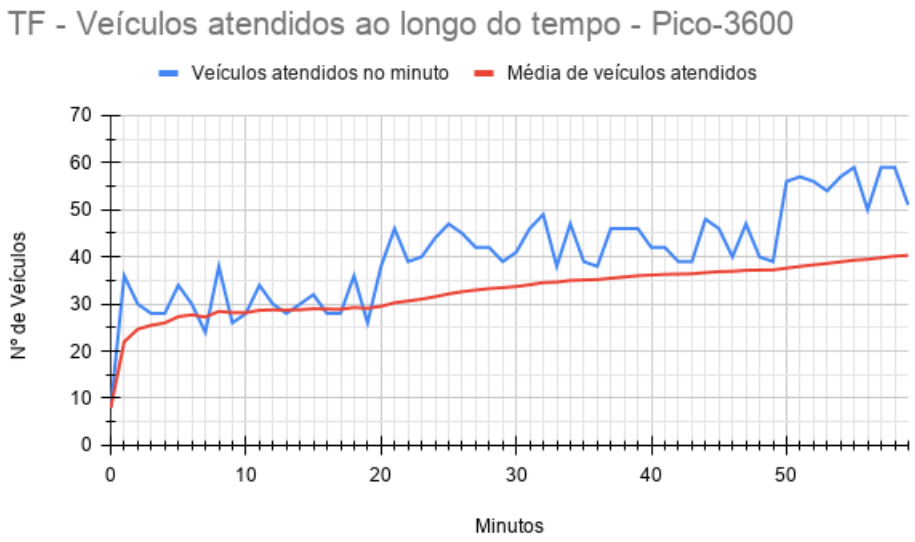

Figura 13 – Resultados de fluxo para TF em Pico-3600

Figura 14 – Resultados de tempo de espera para TF em Pico-3600

3.3 Hierarchical Reinforcement Learning

3.3.1 Características gerais

Este estudo propõe o uso de *Hierarchical Reinforcement Learning* como meio de dividir o problema de gerenciamento de um semáforo em tarefas menores e especializadas, afim de tirar o máximo de desempenho no controle do fluxo. Foram estabelecidos, dois sub-objetivos: o de maximizar a vazão de veículos e o de otimizar os tempos de espera, cada um formando uma sub-política. A determinação de qual sub-objetivo perseguir é feita pela política principal. Todas as três políticas, definidas a seguir, são gerenciadas pelo AI que obedece as definições de HRL:

- Sub-política Q-Size (π_q): Busca atender o máximo de veículos possível, através de ações primitivas;
- Sub-política W-Size (π_w): Busca diminuir o tempo de espera no cruzamento, através de ações primitivas;
- Política H-Agent (μ): Seleciona uma das duas sub-políticas Q-Size e W-Size. Busca iniciá-las quando seu desempenho é ótimo, maximizando seu benefício.

Para todos os efeitos, as sub-políticas são um processo de *Q-learning* completo, em que o agente poderia assumir qualquer uma delas como política principal e interagir com o ambiente de acordo. Sendo assim, é possível não só comparar os resultados obtidos pelo *Hierarchical Reinforcement Learning* com o Semáforo de Tempos Fixos, mas também com agentes que utilizem apenas a sub-política Q-Size ou W-Size de forma exclusiva. Ainda, o processo de treino, detalhado mais adiante, de ambas as sub-políticas, necessário para o treino do HRL, ocorre justamente com o agente seguindo-as exclusivamente, fazendo com que os resultados observados sejam iguais àqueles que se obteria caso o agente fosse um QL comum. Logo, com tais valores em mãos, resta apenas compará-los.

Com as políticas definidas, é necessário estabelecer três conjuntos de dados: o de características que definem os estados, S, o de ações primitivas, A, que interagem durante $t = 1$ instantes de tempo com o ambiente e o de *options*, O, que interagem com ambiente por $t = n$ instantes de tempo.

Os estados são definidos a partir de uma observação do ambiente, isto é, o processo retorna um conjunto de dados que são interpretados de forma a trazer significado para o AI, sendo que todos os cenários definidos para treino e teste retornam três conjuntos de informações: o número de veículos em cada via (sendo que o máximo é de 25), o momento que cada veículo entrou na fila, e se a fase de cada semáforo está em verde ou vermelho (tempo entreverdes são ignorados). O tempo de verde foi configurado com

o mínimo permitido, de 10 segundos (DENATRAN, 1984), e o de entreverdes com 5 segundos, totalizando um tempo de fase de 15 segundos. Sendo assim, três classificações foram elaboradas:

- E_q: Classificação pelo número de veículos nas filas. As vias de um mesmo sentido, NS com SN e OL com LO, são vistas como uma única fila. Tem-se:
  - LOW: Caso a quantidade de veículos na fila seja menor ou igual a vazão máxima (≤ 20);
  - MID: Caso a quantidade de veículos na fila seja menor ou igual a duas vezes a vazão máxima (≤ 40);
  - HIGH: Caso a quantidade de veículos na fila seja maior que duas vezes a vazão máxima (> 40).
- E_w: Classificação pelo tempo médio de espera das filas:
  - LOW: Caso o tempo médio de espera seja menor ou igual ao tempo mínimo de verde (≤ 10);
  - MID: Caso o tempo médio de espera seja menor ou igual ao tempo mínimo de verde + tempo de uma fase (≤ 25);
  - HIGH: Caso o tempo médio de espera seja maior que tempo mínimo de verde + tempo de uma fase (> 25).
- E_{tf}: Estado das fases do semáforo:
  - g: caso a fase esteja no estado verde;
  - r: caso a fase esteja no estado vermelho.

Os valores de vazão de E_q estão relacionados a vazão máxima num tempo de verde mínimo. Para obtê-los, aplicou-se um Semáforo de Tempos Fixos, com tempo de verde de 10 segundos e entreverde de cinco segundos, ao cenário Pico-3600, que gera fluxo acima da capacidade das vias em momentos de surto, e o valor de vazão máximo em uma fase obtido durante essa simulação foi de 20. As clases MID e HIGH foram determinadas de acordo com a definição de ciclo. Como um ciclo é a sequência de acionamento de todas a fases e, no cenário estudado, são duas, logo, uma fila que possua 40 veículos gasta até um ciclo, já que a vazão é 20 veículos por fase para ser esvaziada e é classificada como MID. Com mais de 40 veículos, mais de um ciclo é necessário, levando a classificação para HIGH.

Analogamente, o mesmo foi feito para a classe E_w. Uma via recebe classificação MID caso o tempo de espera de seus veículos seja, no máximo, igual a 25 segundos,

indicando que a via oposta foi selecionada duas vezes seguidas para receber a cor verde. Este cenário pode ser observado na Figura 15, entre S0 e S2, onde a via vertical (V) é selecionada duas vezes seguidas ($a_0 = v$ e $a_1 = v$), levando a via horizontal (H) a um tempo de espera de 25 segundos (10 segundos de verde + 10 segundos de verde + 5 segundos de transição). Qualquer valor acima desse intervalo indica que a via está a mais de duas fases, logo, mais do que um ciclo, em espera, levando-a a ser classificada como HIGH.

As políticas usam combinações das classificações definidas acima para descrever seu estado e estes serão detalhados nos respectivos sub-tópicos.

Já as ações primitivas foram limitadas a duas:

- **h**: passar (ou manter) a via horizontal para o estado verde pelo tempo mínimo (10 segundos);
- **v**: passar (ou manter) a via vertical para o estado verde pelo tempo mínimo (10 segundos).

É importante frisar que as ações ocupam, no cruzamento semaforizado, um mínimo de 10 segundos, caso a ação estenda um verde já ativo, ou 15 segundos (5 segundos de entreverde + 10 segundos de verde), caso selecione uma fase que está em vermelho. Ainda, as ações e, por consequência, as definições de estado, sempre se dão ao fim do tempo de verde mínimo da fase que foi selecionada na ação anterior. Apesar da variação de tempo entre as ações, o AI sempre as enxerga como um único passo, já que foi feita uma única interação com o ambiente. A Figura 15 ilustra como seriam três interações com o ambiente. No início das iterações, o agente aguarda até o fim do verde que estiver ativo para definir o primeiro estado, S_0, e tomar a primeira ação, a_0. Como a primeira ação pediu uma inversão da fase ativa, o semáforo aciona primeiro o tempo entreverdes para só então colocar a via Vertical em verde, fazendo com que a_0 dure 15 segundos. Ao final do tempo de verde, ocorre a segunda definição de estado e tomada de ação (S_1, a_1) e esta, por sua vez, mantém a via ativa em verde, logo, dura apenas 10 segundos, já que não tem que ativar o entreverdes.

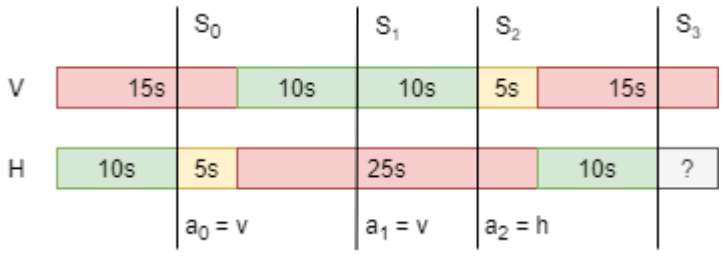

Figura 15 – Diagrama de estados e ações ao longo do tempo

As *options* dão início a um dos dois sub-objetivos que se buscar atingir, sendo que:

- σ_q: Da início a execução de ações primitivas seguindo a sub-política Q-Size;
- σ_w: Da início a execução de ações primitivas seguindo a sub-política W-Size.

O diagrama apresentado na Figura 16 resume a relação entre políticas, *options* e ações, onde a hierarquia do Agente Inteligente proposto para o cruzamento em questão segue a seguinte ordem: A macro-política H-Agent seleciona qual sub-objetivo perseguir ao selecionar uma das duas sub-políticas, Q-Size ou W-Size, através da *options* e estas executam ações primitivas que influenciam diretamente no semáforo do cruzamento, cada uma trabalhando para atingir seu sub-objetivo.

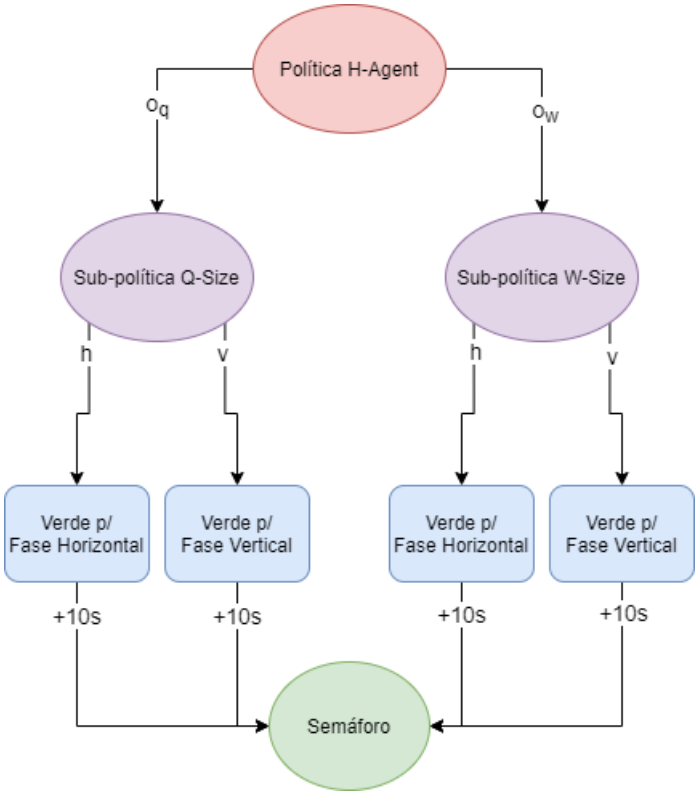

Figura 16 – Relação entre políticas e sub-políticas do agente HRL

3.3.1.1 Execuções de treino e teste

O processo de treino do agente foi divido em três, um para política principal e dois para suas sub-políticas, e se deu através de 24 repetições, que simulam a duração de um dia completo de operação, do cenário Fixo-3600. Este foi selecionado por oferecer diversos níveis de saturação em todas as vias, possibilitando ao AI conhecer o máximo possível de variações disponíveis no cenário em questão. A política de seleção de ação/*option* ótima utilizada foi a ϵ-greedy, que alterna entre momentos de exploração e de aproveitamento de acordo com o parâmetro ϵ, este representando a probabilidade de o AI explorar ao invés de selecionar a melhor ação/*option* conhecida e que dá a ele aleatoriedade em seu comportamento. Ainda, fez-se uso de uma lógica de decaimento, de modo que, no início

do treino, a taxa de exploração é mais alta, configurada em 45% ($\epsilon = 0.45$). A partir daí, a cada três execuções, ϵ decai em 5%, terminando o treino com uma taxa de exploração de 10%.

Os cenários de teste utilizados foram os mesmos aplicados ao Semáforo de Tempos Fixos: Fixo-1800, Pico-2700 e Pico-3600. O primeiro é o cenário ideal para TF e os dois últimos simulam períodos de surto de demanda, que vão além de sua capacidade de vazão, logo, comparar os resultados obtidos nos três cenários permite analisar o desempenho do AI perante aquilo que é alcançado com os semáforos clássicos. Os testes foram feitos em uma única execução do cenário, com uma taxa de exploração fixa configurada em 2% ($\epsilon = 0.02$).

Dessa maneira, sempre que for mencionado treino e teste, o texto refere-se a este ciclo de 24 execuções durante o treino e uma durante o teste.

3.3.1.2 Configuração dos Hiperparâmetros

A taxa de aprendizagem, α, e de desconto, γ foram configurados individualmente para cada uma das três políticas do AI, sendo que os valores que apresentaram melhor desempenho, para cada uma delas, foram obtidos através de uma busca exaustiva entre todas as combinações possíveis de dois conjuntos, um para α, de 0.1 até 1, com intervalo de 0.1, e outro para γ, de 0.1 até 0.9, também com intervalo de 0.1. A busca se deu em duas etapas:

1. Busca Geral: Foram obtidas médias de desempenho entre três execuções do ciclo de treino (Fixo-3600) e teste (Pico-3600) e, através desses resultados, foi feita uma filtragem específica para cada política:
   - Q-Size: foram selecionados os cinco pares de hiperparâmetros que apresentaram a maior vazão média de veículos;
   - W-Size: foram selecionados os cincos pares que apresentaram o menor tempo de espera médio.
   - H-Agent: foram selecionados nove pares de hiperparâmetros, sendo:
     - três que apresentaram a maior vazão média de veículos;
     - três que apresentaram o menor tempo de espera médio;
     - três que apresentaram a melhor relação $\frac{vazão\ média}{espera\ média}$.
2. Busca refinada: Os pares selecionados na fase de Busca Geral foram novamente submetidos ao ciclo de treino e teste, no entanto, dessa vez as médias foram tiradas entre 10 execuções. O melhor par dessa fase foi selecionado como o ideal para a política em questão.

Os dados de desempenho coletados em cada uma das fases foram:

- max_q: Maior número de veículos atendidos dentre as execuções;
- min_q: Menor número de veículos atendidos dentre as execuções;
- avg_q: Média de veículos atendidos;
- std_q: Desvio padrão da média de veículos atendidos ;
- max_w: Maior tempo de espera dos veículos dentre as execuções;
- avg_w: Menor tempo de espera dos veículos dentre as execuções;
- min_w: Média do tempo de espera dos veículos;
- std_w: Desvio padrão da média do tempo de espera;
- rel_h: Relação $\frac{vazão\,média}{espera\,média}$ (Apenas para H-Agent).

Os resultados obtidos para cada uma das políticas são discutidos em suas respectivas seções. Como destacado anteriormente, as sub-políticas são, na verdade, políticas otimizadas através do algoritmo clássico de QL. Sendo assim, nos próximos tópicos, ambas serão tratadas dessa maneira e apenas quando a Política H-Agent for explorada é que será devidamente introduzida na hierarquia.

3.3.2 Sub-política Q-Size

3.3.2.1 Definições

A Sub-política Q-Size tem por objetivo maximizar o número de veículos atendidos. Para tanto, foi utilizada a classe E_q, que categoriza as filas de veículos de acordo com o seu tamanho, com o pressuposto de que se o agente der preferência para a fila que tiver mais veículos, isso o leva a maximizar o fluxo. Ainda, o agente precisa conhecer qual o estado atual do semáforo, determinado pela classe E_{tf}, para saber qual é a via com preferência de passagem no momento e, então, ter condição de selecionar a ação correta. Dessa forma, os estados para a Sub-política Q-Size são descritos por:

- E_q: Nove estados possíveis, sendo:
  - Via Horizontal: LOW, MID ou HIGH
  - Via Vertical: LOW, MID ou HIGH
- E_{tf}: Dois estados possíveis, já que os estados entre os semáforos são excludentes, sendo:

- Via Horizontal: G ou R
- Via Vertical: G ou R

A combinação de ambas as classes totalizam, então, um conjunto $9 \cdot 2 = 18$ estados.

O único conjunto de ações disponível durante a sub-política é o de ações primordiais, A, dessa forma, a tabela $Q(s, a)$ possui 18 linhas e duas colunas, totalizando um universo de 36 pares (s, a).

Além dos estados que descrevem o cruzamento, também precisa ser definida uma função de recompensa, que gere um *feedback* que indique se a ação do agente foi, ou não, acertada de acordo com o seu objetivo. Para o caso da política Q-Size, utilizou-se:

$$r = \begin{cases} 1 & \textit{se } n^o \textit{ de veículos em } t{+}1 < \textit{que em } t \\ 0 & \textit{caso contrário} \end{cases} \tag{3.1}$$

A Figura 17 demonstra como o agente, sob a sub-política Q-Size, interage com o ambiente. A figura 17*a* apresenta um possível cenário de um cruzamento semaforizado: a via vertical (V) possui 30 veículos e está com a cor vermelha ativa, enquanto a via horizontal (H) possui oito veículos e está no final da cor verde ativa. Outros oito veículos, divididos igualmente entre as vias, entrarão no cruzamento durante a próxima interação. A Figura 17*b* apresenta como o agente enxerga tal cenário: a via V é classificada como MID por possuir 30 veículos, enquanto a H, com oito, é classificada como LOW. Já o semáforo recebe o estado [R, G], com V em vermelho e H em verde, completando então o estado (MID, LOW, [R, G]). O agente possui, então, duas alternativas, caso opte pela ação de acionar a cor verde para V, 20 veículos (vazão máxima) deixarão o cruzamento, enquanto apenas oito entram, diminuindo o número de veículos no cruzamento e retornando uma recompensa $r_0 = 1$. Caso opte pela ação de liberar a via H, apenas oito veículos deixam o cruzamento, enquanto outros oito entram, mantendo o número antes e depois da ação iguais, levando a uma recompensa $r_0 = 0$, logo, a ação V é a melhor neste cenário.

3.3.2.2 Hiperparâmetros

Como descrito na seção 3.3.1.2, a definição dos hiperparâmetros α e γ envolveu uma rodada de testes com 90 pares, aplicados três vezes ao ciclo de treino e teste, nos cenários Fixo-3600 e Pico-3600 respectivamente. Destes, os cinco melhores foram selecionados para serem reaplicados, agora dez vezes, ao ciclo de treino e teste. O principal dado analisado foi a média de veículos atendidos, avg_q e o melhor par da segunda rodada foi selecionado para a Sub-política Q-Size.

Os resultados para os cinco melhores pares podem ser vistos nas Tabelas 4 e 5, onde os melhores resultados de cada coluna foram destacados em verde.

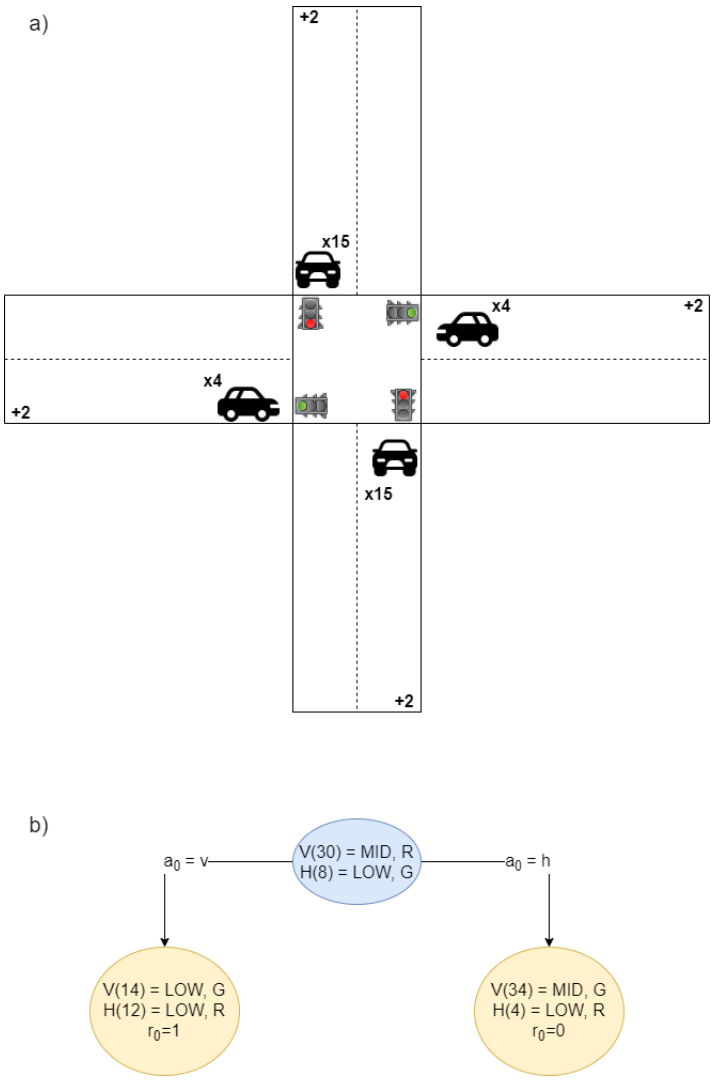

Figura 17 – Tradução do ambiente para estados do Q-Size

Tabela 4 – Resultados da primeira rodada de testes de hiperparâmetros de Q-Size.

α	γ	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w
0,1	0,1	2965	2957	2959,66	3,77	30,22	22,11	26,12	3,31
0,1	0,5	2970	2953	2959,66	7,4	28,04	21,72	24,36	2,73
0,1	0,4	2964	2954	2958	4,32	27,86	21,21	25,03	2,8
0,1	0,6	2958	2949	2952,66	3,85	27,45	20,64	23,75	2,81
0,2	0,4	2954	2948	2951,33	2,49	27,85	21,61	25,59	2,82

Tabela 5 – Resultados da segunda rodada de testes de hiperparâmetros de Q-Size.

α	γ	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w
0,1	0,4	2962	2946	2956,6	4,27	28,39	22,18	25,10	2,54
0,1	0,5	2972	2854	2941	36,87	28,47	21,37	25,74	2,27
0,1	0,6	2960	2860	2929,3	37,75	28,01	22,02	26,62	1,71
0,2	0,4	2965	2745	2922,3	68,45	28,46	21,66	24,36	2,37
0,1	0,1	2963	2841	2911,8	46,82	30,65	22,38	28,35	2,24

Foi selecionado então, $\alpha = 0,1$ e $\gamma = 0,4$, por ter apresentado não só maior média de vazão, mas também a maior estabilidade (menor desvio-padrão).

3.3.2.3 Resultados

Q-Size foi submetido aos mesmos cenários que o TF: Fixo-1800, Pico-2700 e Pico-3600. No entanto, os resultados demonstrados aqui são a média de 10 ciclos de teste/treino, já que, pela natureza do agente em assumir comportamento aleatório, de acordo com seu parâmetro de explorar versus aproveitar, ϵ, os valores podem divergir entre diferentes execuções, logo, a média busca amortizar essas diferenças. Os valores observados podem ser vistos na Tabela 6, com os melhores resultados destacados em verde.

Tabela 6 – Resultados para a aplicação do agente Q-Size e TF aos cenários de teste

Cenário	V_s	V_m	W
Q-Size			
Fixo-1800	1779	29,66	25,09
Pico-2700	2433	40,56	25,32
Pico-3600	2929	48,83	27,79
TF			
Fixo-1800	1784	29,73	16,63
Pico-2700	2349	39,15	29,34
Pico-3600	2420	40,33	31,53

O desempenho para o cenário Fixo-1800 apresenta valores parecidos com os observados para TF com exceção do tempo médio de espera, W, que é 8,46 segundos maior, tornando o Q-Size 50% mais lento que o TF para este caso em específico. Tal diferença é motivada pelo fato de que o agente Q-Size não busca otimizar os tempos de espera, mas sim aumentar a vazão, priorizando a fila com mais veículos, o que leva uma via a ser selecionada seguidas vezes em detrimento da outra, aumentando o tempo de espera dos veículos nela. Ainda assim, Q-Size não obtém valores melhores que TF.

As Figuras 18 e 19 apresentam o rendimento minuto a minuto do agente. Destaca-se como o formato de onda triangular não é mais tão bem definido, isso se dá porque Q-Size tem liberdade de atender uma mesma via seguidas vezes, de acordo com a demanda.

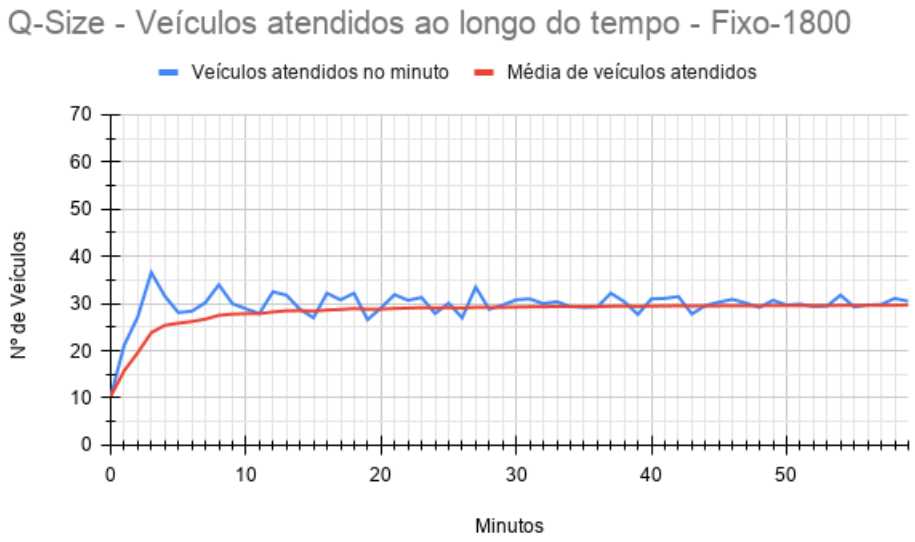

Figura 18 – Resultados de fluxo para Q-Size em Fixo-1800

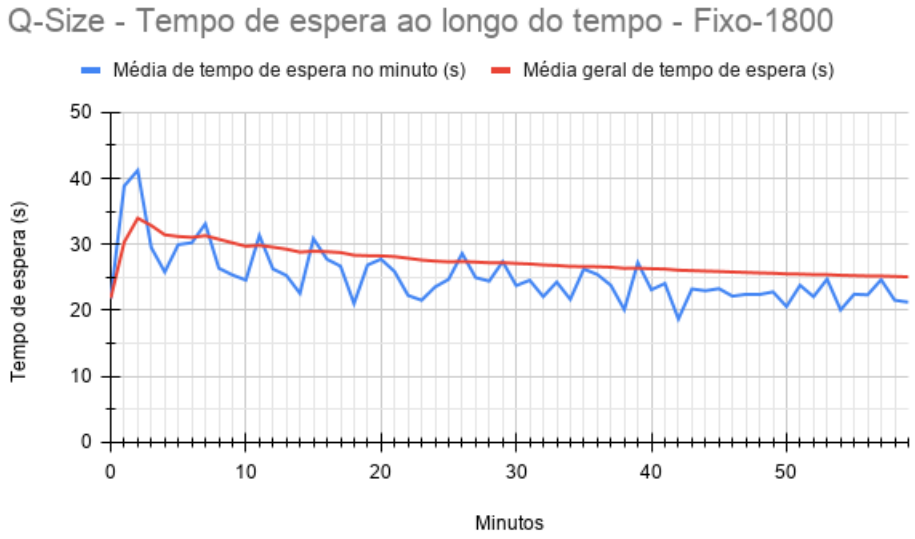

Figura 19 – Resultados de tempo de espera para Q-Size em Fixo-1800

No cenário Pico-2700, a capacidade que Q-Size tem de selecionar via por demanda começa a mostrar resultados melhores. Para o mesmo fluxo de entrada, há um aumento de 3,57% na vazão, apesar da vazão/minuto ser virtualmente a mesma. O destaque maior se dá pelo tempo de espera, 13,7% menor. Apesar do aumento tímido na vazão, ao se observar a Figura 20, é possível notar que o agente reage rapidamente a nova demanda de pico, que ocorre entre os minutos 20 e 30, com adaptação rápida na vazão, finalizando o escoamento aos 40 minutos. Isso se dá porque, de acordo com as configurações da via, o fluxo máximo em uma única pista é de 1875 v/h (DENATRAN, 1984), ou 31,25 v/m, no entanto, a segunda fase do pico envolve um fluxo de 45 v/m em uma única via, acima do que é de fato suportado, logo, apesar da otimização do controle do semáforo, a via em

si também está em um estado de constrição, refletindo então na vazão do cruzamento. Embora isso também pudesse servir de justificativa para o TF, foi observado, na Figura 11, que ele não é capaz de atender a demanda mesmo após o seu término, prolongando o efeito de constrição.

Figura 20 – Resultados de fluxo para Q-Size em Pico-2700

Para o tempo médio de espera, a diferença é mais acentuada. O TF teve um impacto grande no desempenho já no início do primeiro surto, como observado na Figura 12, diferente do que acontece para Q-Size, que consegue manter os valores próximos a média desde o início da simulação. Mais uma vez, isso é resultado da capacidade do agente de escolher livremente qual via atender de acordo com a demanda: caso uma via não possua qualquer veículo, ela não será selecionada em detrimento de uma que tem demanda maior.

O cenário Pico-3600, traz as diferenças mais contundentes. Q-Size apresentou uma vazão 21% maior do que o TF, com um tempo de espera 11,8% menor. Considerando que, para o cenário Pico-2700, o agente foi capaz de atender toda a demanda do primeiro pico durante sua duração mais os 10 minutos seguintes, é de se esperar que esse espaço disponível possa ser usado para atender veículos de uma demanda maior. É justamente isso que se pode observar na Figura 22. Aqui, a demanda atinge 135 v/m no seu momento mais extremo. No entanto, o agente adapta-se rapidamente ao surto, apenas dois minutos após seu início, atendendo 60 v/m. Ainda, a média de fluxo apresenta uma subida ainda maior do que a observada no Pico-2700, indo para 40 v/m (máximo observado para o TF), aos 30 minutos e chegando a um máximo de 48,83 v/m, 21,07% maior do que o visto no TF. No entanto, não há capacidade de dar vazão a toda demanda gerada, mesmo com os números altos, justificado pela constrição física da via, que é capaz de suprir apenas 31,25 v/m, contra os 60 v/m do momento mais crítico.

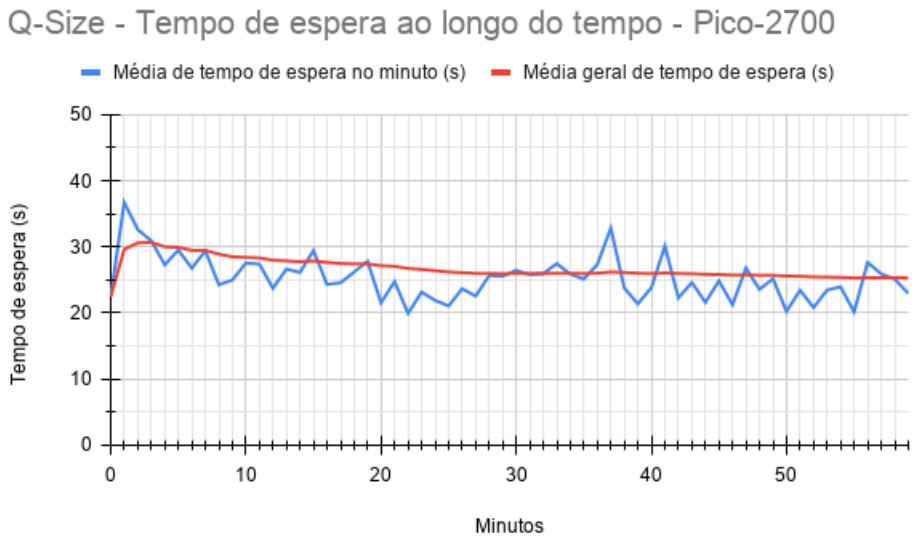

Figura 21 – Resultados de tempo de espera para Q-Size em Pico-2700

Figura 22 – Resultados de fluxo para Q-Size em Pico-3600.

O gráfico de tempo médio no Pico-3600, ilustrado pela 23, apresenta um formato parecido com o do Pico-2700, com a diferença de que o primeiro apresenta uma leve tendência de queda, enquanto o segundo demonstra caminhar para estabilidade. Isso é um indício de que o agente Q-Size está próximo ao seu ponto máximo de rendimento, no entanto, dado que o fluxo de pico está acima da capacidade da via, valores maiores de fluxo de entrada seriam impraticáveis em qualquer modelo de abordagem, elevando apenas o tempo de espera sem gerar grande impacto na vazão.

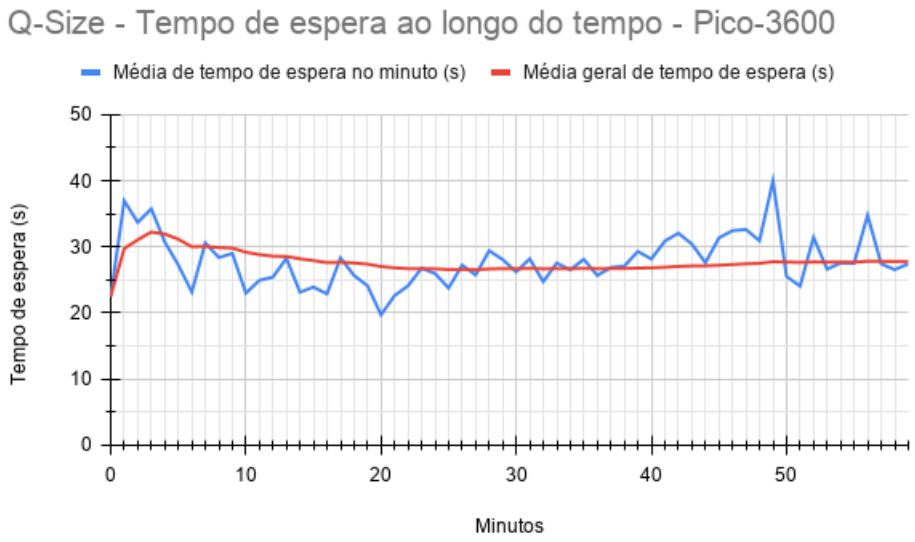

Figura 23 – Resultados de tempo de espera para Q-Size em Pico-3600.

3.3.3 Política W-Size

3.3.3.1 Definições

A Sub-política W-Size tem por objetivo minimizar o tempo de espera dos veículos no cruzamento. Isso faz com que W-Size use um conjunto de estados diferente de Q-Size. Aqui, E_w é usada para categorizar as filas de veículos de acordo com seu tempo de espera médio, e isso implica que o agente não terá informação alguma de quantos veículos há em cada fila. O pressuposto é que, independente da quantidade de veículos, o agente deve selecionar a via em que seus elementos estão há mais tempo esperando para serem liberados, diminuindo o tempo de espera geral do cruzamento. Ainda, da mesma forma como foi feito para Q-Size, o agente aqui também precisa saber qual o estado do semáforo, E_{tf}, para tomar as ações primitivas de acordo. Sendo assim, o conjunto de estados para W-Size é:

- E_w: Nove estados possíveis, sendo:
  - Via Horizontal: LOW, MID ou HIGH;
  - Via Vertical: LOW, MID ou HIGH.
- E_{tf}: Dois estados possíveis, sendo:
  - Via Horizontal: G ou R;
  - Via Vertical: G ou R.

Como W-Size também se limita as ações primordiais, a tabela $Q(s, a)$ totaliza, mais uma vez, um universo de 36 pares *(s, a)*, com 18 estados e duas ações.

A função de recompensa também possui formato análogo ao configurado para Q-Size:

$$r = \begin{cases} 1 & \textit{se o tempo de espera em } t{+}1 < \textit{que em } t \\ 0 & \textit{caso contrário} \end{cases} \tag{3.2}$$

A Figura 24 ilustra como o agente, sob a Sub-política W-Size, interage com o ambiente. A Figura 24*a* representa um possível cenário de um cruzamento: a via vertical (V), de cor vermelha ativa, possui um tempo de espera médio de 10 segundos ((15 + 5)/2), enquanto a via horizontal (H), no final de sua cor verde ativa, possui um tempo de espera médio de 30 segundos ((50 + 10)/2). A figura Figura 24*b* apresenta como o agente interpreta o cenário: a via V recebe a classificação LOW, por possui um tempo médio de espera de 10 segundos, igual ao tempo mínimo de verde. Já a via H é classificada como HIGH, porque 30 segundos está acima do tempo de 25 segundos. O semáforo, por sua vez, recebe o estado [R, G], de acordo com a cor ativa de cada via, levando então ao estado (LOW, HIGH, [R G]). Caso o agente decida atender V e supondo que todos os veículos foram atendidos, o tempo de espera desta via zera, diminuindo o tempo médio do cruzamento em 10 segundos, no entanto, o tempo na via horizontal e, portanto, do cruzamento, acresce de uma fase (15 segundos), levando o cruzamento a um tempo de espera maior, de 45 segundos, logo, uma recompensa $r_0 = 0$. Caso a ação tomada seja liberar a via H e, mais uma vez, todos os veículos foram atendidos, o tempo médio do cruzamento decresce de 30 segundos, e, mesmo com a via vertical tendo um acréscimo de 15 segundos, o tempo em $t+1$ fica 15 segundos menor que em t, gerando uma recompensa de $r_0 = 1$, logo, a melhor ação neste caso.

3.3.3.2 Hiperparâmetros

O mesmo processo empregado na seção 3.3.2.2 é repetido aqui, no entanto, ao invés de usar avg_q como o dado de análise, é usando avg_w. Os resultados para os cinco melhores pares podem ser vistos nas Tabelas 7 e 8 e melhor resultado individual de cada coluna está destacado em verde.

Foram selecionados $\alpha = 0,1$ e $\gamma = 0,1$, com a menor média de tempo de espera, bem como menor desvio-padrão.

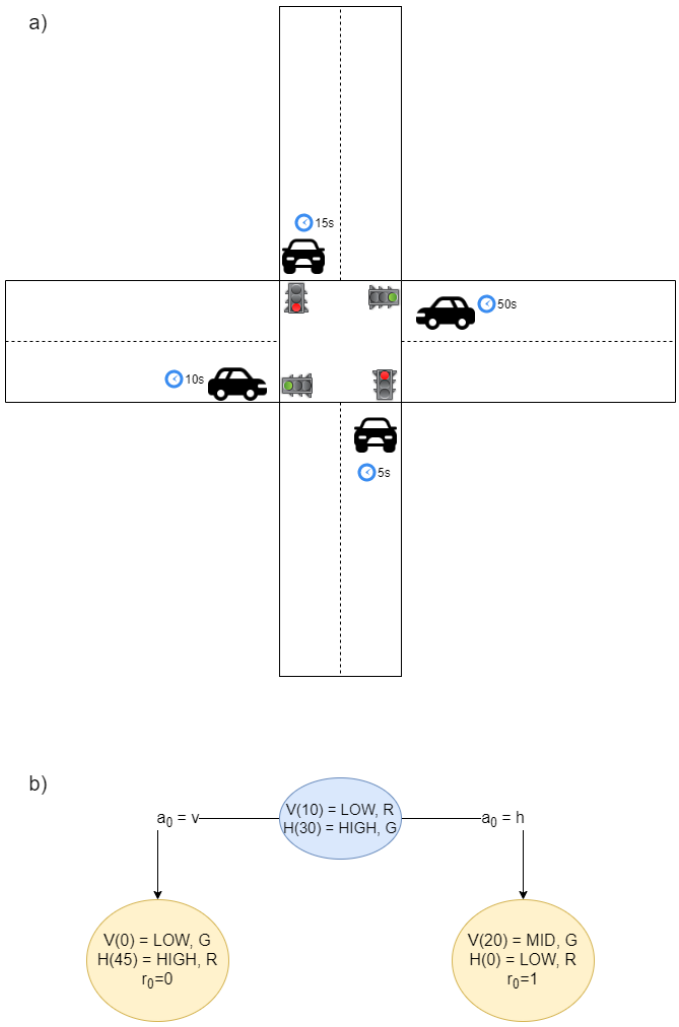

Figura 24 – Tradução do ambiente para estados do W-Size

Tabela 7 – Resultados da primeira rodada de testes de hiperparâmetros de W-Size.

α	γ	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w
0,1	0,1	2859	2844	2849,33	6,84	21,13	20,73	20,95	0,16
0,1	0,4	2865	2843	2853,66	8,99	21,91	20,70	21,15	0,54
0,2	0,1	2859	2801	2835,33	24,85	23,67	20,32	21,75	1,41
0,2	0,4	2845	2833	2838	5,09	22,31	21,52	21,86	0,33
0,3	0,5	2850	2821	2838,66	12,65	22,10	21,85	21,98	0,10

Tabela 8 – Resultados da segunda rodada de testes de hiperparâmetros de W-Size.

α	γ	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w
0.1	0.1	2865	2788	2837,6	22,15	23,64	20,94	21,86	0,78
0.1	0.4	2842	2644	2805,9	56,87	25,16	20,75	22,39	1,32
0.2	0.4	2863	2686	2806,5	54,60	24,90	20,86	22,61	1,48
0.2	0.1	2856	2655	2777,1	80,19	25,71	21,14	23,15	1,62
0.3	0.5	2853	2638	2784,9	71,43	25,93	21,01	23,17	1,66

3.3.3.3 Resultados

W-Size foi submetido aos mesmos cenários que TF e Q-Size: Fixo-1800, Pico-2700 e Pico-3600 e, assim como Q-Size, também teve seus resultados tirados da média de 10 ciclos de teste e treino, estes reunidos na Tabela 9, com os melhores resultados destacados em verde:

Tabela 9 – Resultados para a aplicação do agente W-Size, Q-Size e TF aos cenários de teste

Cenário	V_s	V_m	W
W-Size			
Fixo-1800	1782	29,23	17,14
Pico-2700	2396	39,93	19,05
Pico-3600	2826	47,03	23,21
Q-Size			
Fixo-1800	1779	29,66	25,09
Pico-2700	2433	40,56	25,32
Pico-3600	2929	48,83	27,79
TF			
Fixo-1800	1784	29,73	16,63
Pico-2700	2349	39,15	29,34
Pico-3600	2420	40,33	31,53

Mais uma vez, o desempenho para o cenário Fixo-1800 apresenta valores parecidos, no entanto, W-Size se mostra capaz de atingir valores de tempo de espera 31,68% melhores que Q-Size. As Figuras 25 e 26 exibem o desempenho de W-Size para este caso. Como o cruzamento possui um fluxo bem abaixo de seu ponto de saturação e igualmente distribuído entre as vias, o desempenho na vazão torna-se trivial, independente da abordagem. No entanto, o tempo de espera pode sofrer variações mesmo num cenário de igualdade de fluxo, a depender da ordem de seleção das vias a receberem prioridade e, como W-Size é capaz de avaliar essa dimensão, apresenta então resultados melhores que Q-Size e próximos ao TF.

Para Pico-2700, W-Size já demonstra rendimentos melhores e nos mesmos moldes de Q-Size, isto é, a diferença na vazão é marginal, com um aumento de apenas 2% em relação a TF, e uma perda 1,5% quando comparado a Q-Size. No entanto, o tempo de espera é 35% menor que TF e 24,76% menor que Q-Size, comprovando não só a eficácia do agente, mas também como seus resultados são diferentes em relação a Q-Size, já que aqui o agente sacrifica, por enquanto de forma tímida, sua vazão para diminuir o tempo de espera no cruzamento, demonstrando ser capaz de cumprir com o objetivo.

Novamente, o agente se mostra capaz de atender toda a demanda do primeiro surto, como pode ser visto da Figura 27. Isto se dá porque, embora o modelo de tomada de decisão seja diferente, W-Size interage com o ambiente da mesma maneira que Q-Size,

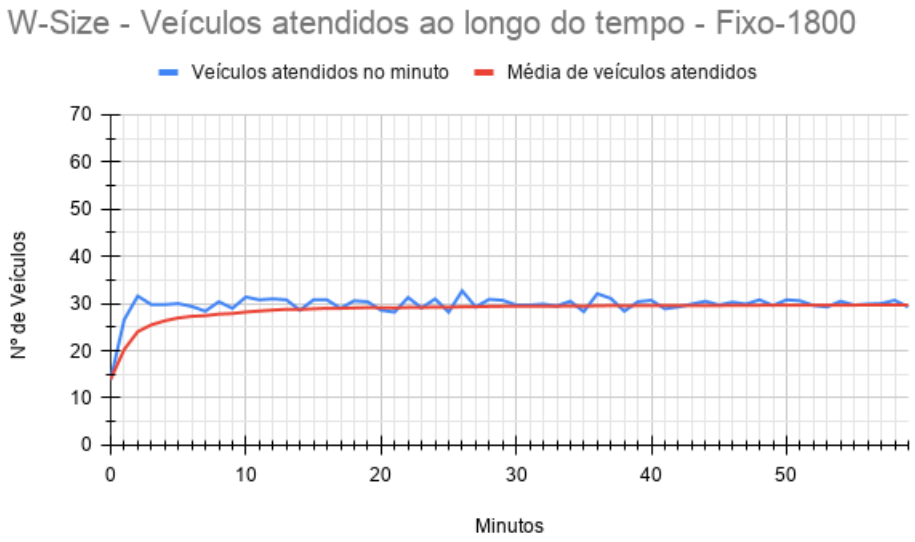

Figura 25 – Resultados de fluxo para W-Size em Fixo-1800.

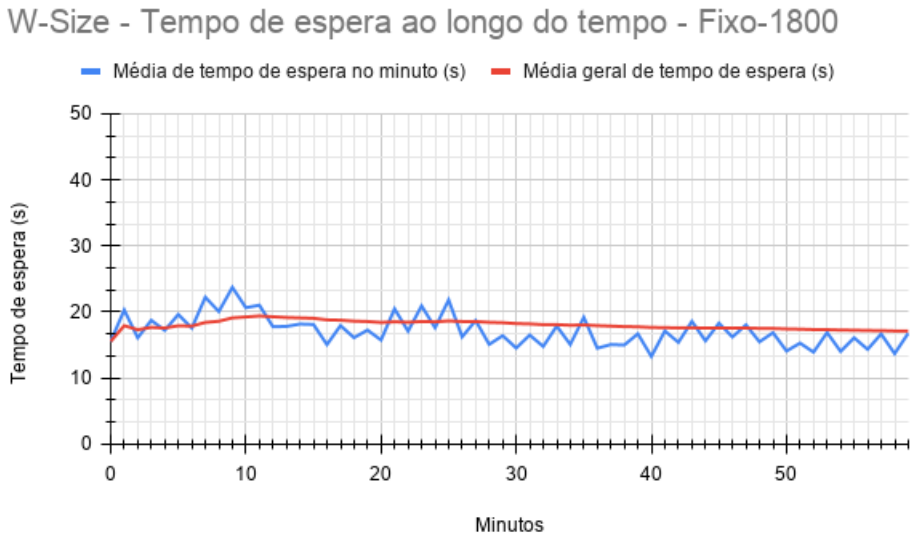

Figura 26 – Resultados de temo de espera para W-Size em Fixo-1800

sendo capaz de selecionar uma mesma via seguidas vezes. Como uma demanda maior em uma via desencadeia também um tempo de espera maior, já que, quanto mais veículos, menor a chance de vazão completa em um verde, logo, veículos podem esperar por mais tempo, há uma correlação entre vazão e queda do tempo de espera, ocasionando então em um fluxo quase tão bom quanto Q-Size, para este cenário. Também é importante notar que, embora a diferença entre as duas políticas seja pequena, Q-Size é capaz de manter uma vazão maior nos momentos de pico, como pode ser observado dos 20 aos 32 minutos e depois dos 50 aos 60 minutos. Também há uma diferença nos picos máximo de vazão, onde W-Size não é capaz de atingir a barreira de 60 v/m, chegando ao máximo de 59 aos 38 minutos, diferente de Q-Size, que possui picos frequentes igual ou acima desse valor.

Isso se reflete no momento em que o primeiro pico da vazão se encerra, aos 37 minutos, 1 minuto antes do que o W-Size.

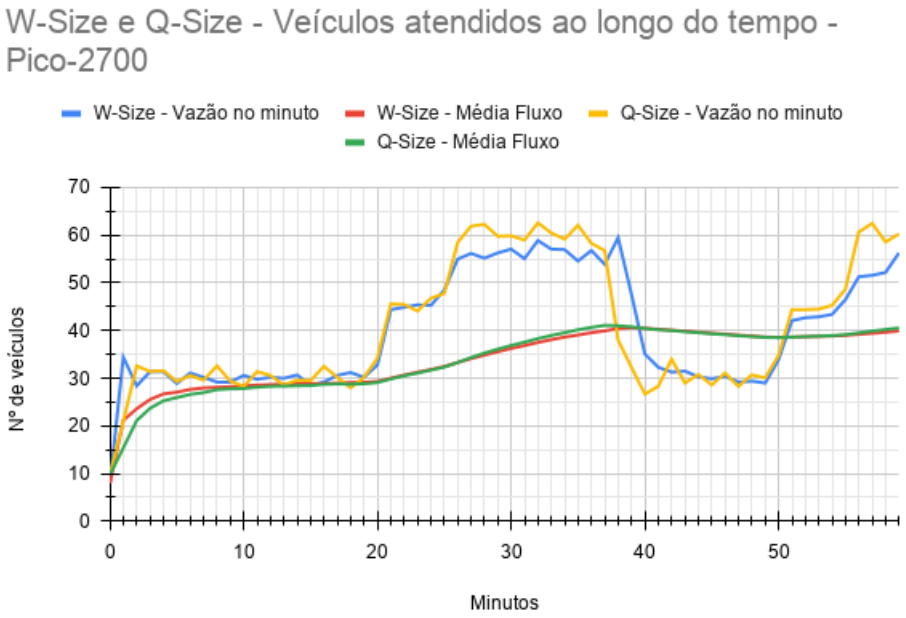

Figura 27 – Resultados de fluxo para W-Size em Pico-2700

Já a evolução do tempo de espera, que pode ser observada na Figura 28, difere não só de TF, mas também de Q-Size (Figura 21). Agora, os picos de tempo de espera coincidem com os surtos de demanda. Isto ocorre porque, quando o agente escolhe uma via em detrimento de outra, observa-se um aumento no tempo de espera natural, já que uma via, invariavelmente, ficará fechada. Sendo assim, em momentos em que a demanda entre as vias difere de forma notável, como ocorre nos surtos, o tempo de espera acaba também por subir. Ainda assim, os valores observados são menores que os de Q-Size, sendo o valor mais alto igual a 27,64, contra 36,72 da outra sub-política.

O cenário Pico-3600 traz, novamente, diferenças mais acentuadas. No entanto, dessa vez elas não se resumem a comparação apenas com TF, os agentes *Q-learning* também exibem resultados diferentes entre si. Neste caso, W-Size apresenta um fluxo 16,77% maior que TF, porém 3,5% menor que Q-Size. Já para o tempo de espera, W-Size é 26,38% melhor do que TF e 16,48% melhor que Q-Size. Tais resultados condizem com a proposta de cada Sub-política: W-Size busca diminuir o tempo de espera enquanto Q-Size busca aumentar a vazão.

Na Figura 29, pode-se observar a diferença de rendimento entre as sub-políticas. W-Size, após o primeiro surto, tem sua vazão orbitando um valor de 55 v/m, como pode ser observado do minuto 21 ao 50, enquanto Q-Size consegue mantê-la próxima a 60 no mesmo período. Já no minuto 23, Q-Size atinge sua maior vazão por minuto, de 62 v/m,

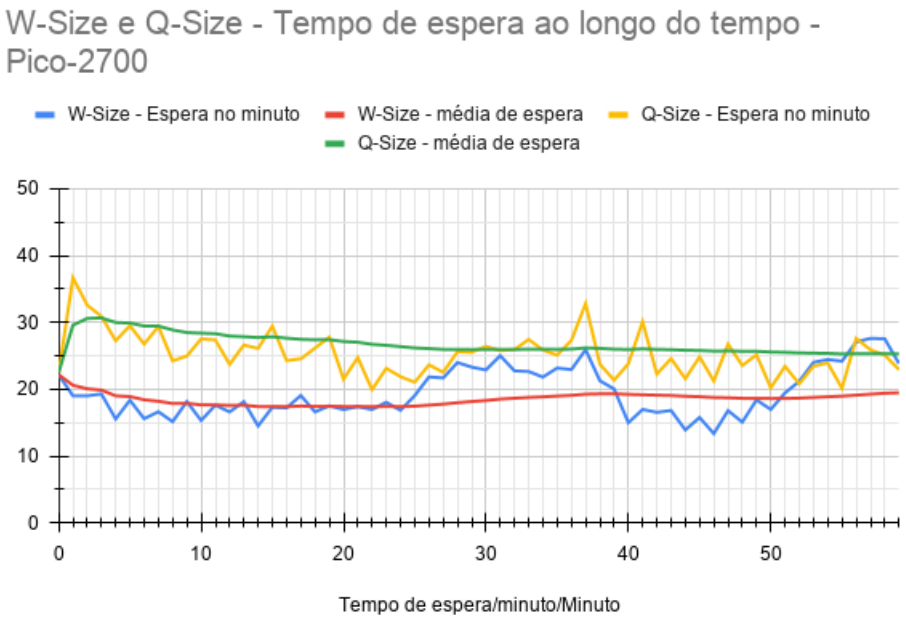

Figura 28 – Resultados de tempo de espera para W-Size em Pico-2700

diferente de W-Size, que, embora atinja um valor maior, de 64 v/m, só o alcança aos 53 minutos de simulação.

Embora ambas políticas atinjam valores médios próximos ao final da simulação (47,03 contra 48,83), os picos e a consistência em momentos chave é que garantem a diferença de desempenho final. Também é válido lembrar que no cenário em questão, existem momentos em que o fluxo de entrada está acima da capacidade da via, como, por exemplo, na segunda metade do primeiro surto, com 60 v/m por via, no entanto, ainda assim, Q-Size consegue atingir valores maiores que W-Size.

Para o tempo de espera, o gráfico de W-Size, apresentado na Figura 30, exibe um comportamento bastante diferente do que foi observado anteriormente. É possível ver como os tempos de espera iniciam baixos e começam a subir e assim se mantém após o primeiro surto, diferente do que foi visto na Figura 28, onde as variações acompanhavam o início e fim do surto. Isso se dá porque o número de veículos na via extrapola a capacidade de vazão de um verde mínimo, o que acarreta em alta no tempo de espera e, como o fluxo aqui é maior, não há momento de alívio no fluxo e, portanto, no tempo de parada. Porém, como o agente é guiado para diminuir tais valores, ele é capaz de assegurar que o valor não suba de forma acentuada, como ocorre com o TF e pode ser visto na Figura 14, impedindo que perdas de desempenho ocorram com maior frequência.

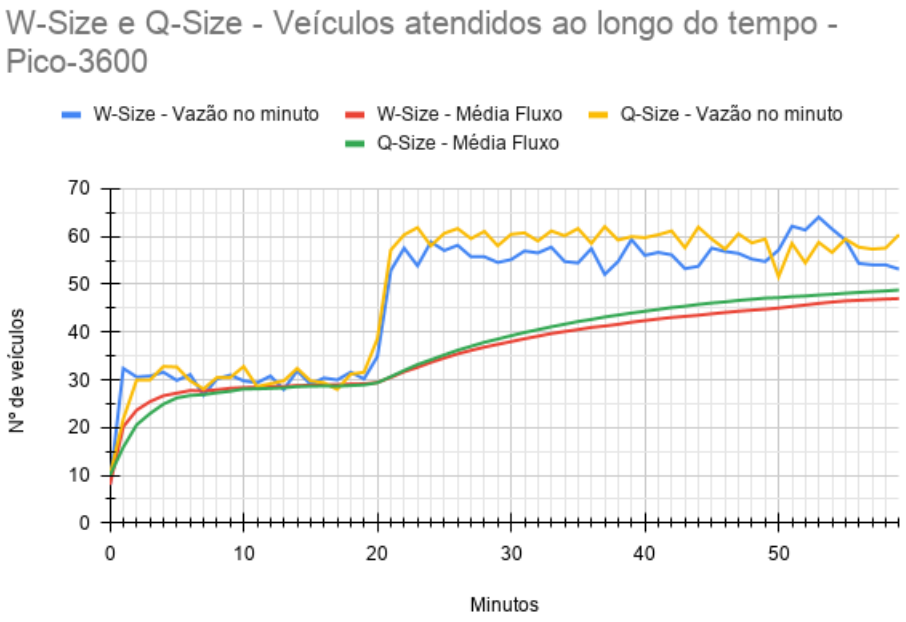

Figura 29 – Resultados de fluxo para W-Size em Pico-3600

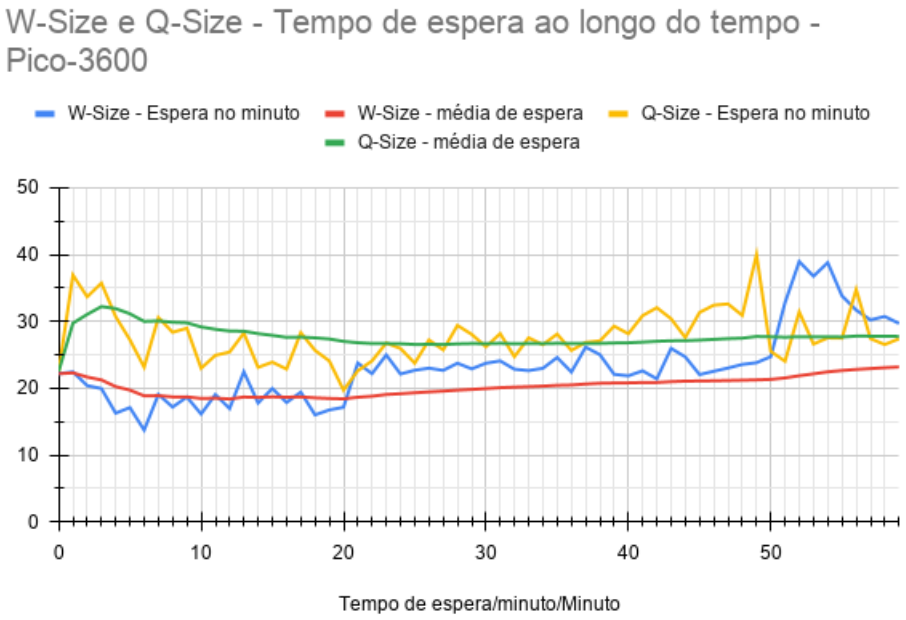

Figura 30 – Resultados de tempo de espera para W-Size em Pico-3600

3.3.4 Política H-Agent

3.3.4.1 Definições

O agente *Hierarchical Reinforcement Learning* regido pela política principal, H-Agent, busca otimizar tanto a vazão de veículos quanto seus tempos de espera e, para isso, faz uso das duas sub-políticas discutidas anteriormente, cada uma especializada em um

desses dois aspectos, para alcançar seu objetivo. O que a política principal faz, portanto, é selecionar uma das políticas de hierarquia mais baixa no momento que ela, a sub-política, terá um rendimento ótimo quanto ao seu objetivo. Ao se descrever os estados dessas sub-políticas, destaca-se que o agente, quando lançando mão de uma delas, precisa ler o ambiente de uma maneira que as informações obtidas sejam traduzidas da forma que lhe traga uma descrição condizente com seu objetivo e isso também vale para a política principal. Logo, sua leitura do ambiente deve lhe trazer informações que ajudem na escolha de qual sub-política selecionar, já que é através dela que seu objetivo será alcançado.

Sendo assim, H-Agent usa como estado a combinação dos estados úteis para cada sub-política: E_q, da sub-política Q-Size, e E_w, da W-Size, ou seja, a política principal deve enxergar o ambiente da mesma maneira que as sub-políticas, para ser capaz de identificar qual delas terá o maior desempenho em cada estado. Ainda, como H-Agent não interage diretamente com o ambiente, o estado que descreve o semáforo, E_{tf}, é irrelevante e, portanto, desconsiderado. Dessa forma, H-Agent descreve o cruzamento através dê:

- E_q: Nove estados possíveis, sendo:
  - Via Horizontal: LOW, MID ou HIGH;
  - Via Vertical: LOW, MID ou HIGH.
- E_w: Nove estados possíveis, sendo:
  - Via Horizontal: LOW, MID ou HIGH;
  - Via Vertical: LOW, MID ou HIGH.

Totalizando, então, 81 estados.

Análogo ao que acontece nas sub-poíticas, H-Agent também tem um único conjunto de ações disponíveis, O, composto pelas duas *Options*, fazendo com que sua tabela $Q(s, o)$ possua duas colunas, totalizando 162 estados. A recompensa, por sua vez, foi definida como a relação entre o número de veículos atendidos, V_f, e o tempo de espera médio, V_w, indicando, dessa maneira, as *options* que retornam a melhor relação de vazão e tempo de espera. No entanto, ambos os valores são normalizados, através de Min-max (AGGARWAL, 2015), para o intervalo $(0, 1)$, onde, para vazão, V'_f, o limite inferior é 0 e o superior, que representa o tamanho máximo de uma fila, 50, enquanto que para o tempo de espera, V'_w, o limite inferior é de 10, que é o tempo mínimo de verde, enquanto o superior é 30, tempo de um ciclo de acordo com os tempos de verde e entreverde configurados para o agente HRL.

$$r = \frac{V'_f}{V'_w + 0,0001} \tag{3.3}$$

A constante $0,0001$ é apenas para garantir que não ocorra uma divisão por zero.

A Figura 31 ilustra como o agente interpreta e interage com o ambiente. A figura Figura 31*a* reúne os cenários apresentados para o Q-Size e o W-Size em um só: a via vertical (V), de cor vermelha ativa, possui um tempo de espera médio de 10 segundos e uma fila com 30 veículos no total, enquanto a via horizontal (H), no final de sua cor verde ativa, possui um tempo de espera médio de 30 segundos, com 8 veículos. No próximo instante t, todas as vias receberão dois novos veículos, no entanto, consideraremos que eles entrarão na fila no último segundo antes do fim da ação, fazendo com que tenham um tempo de espera igual a 0. A figura Figura 31*b* apresenta como o agente interpreta o cenário: a via V recebe a classificação MID, em relação ao volume de carros e LOW, por possuir um tempo médio de espera de 10 segundos. Já a via H é classificada como LOW, por possuir apenas 8 veículos e HIGH, por possuir 30 segundos de tempo médio de espera. Logo, a descrição final do estado fica ([MID, LOW], [LOW, HIGH]). Considerando que as *options* ocupam apenas um instante t, afim de facilitar o exemplo, e de acordo com o mostrado em *b)*, caso o agente decida optar pela *option* que inicia Q-Size, esta, por sua vez, optaria pela ação primordial v, como vimos na Figura 17, gerando então uma recompensa, para a macro-política, de $r_0 = 0,41$. Caso o agente opte pela *option* W-Size, esta tomaria a ação primordial h, retornando uma recompensa de $r_0 = 1,52$, logo, o melhor curso de ação.

3.3.4.2 Hiperparâmetros

O processo de obtenção dos hiperparâmetros se deu de maneira diferente do descrito nas seções 3.3.2.2 e 3.3.3.2. Para o H-agent, no lugar de classificar os resultados através de um único parâmetro, foram observados os três melhores de três parâmetros diferentes, o que levou a um grupo de até nove valores: três que apresentaram a melhor relação entre vazão e tempo de espera médio, rel_h, três com melhor vazão média, avg_q e três com melhor tempo médio de espera, avg_w. Caso algum par fizesse parte de mais de um desses conjuntos, sua adição valeria para ambas as classificações. Também é possível notar que, como H-Agent seleciona entre duas sub-políticas, ambas também devem passar por um processo de treino, sendo assim, para poder se analisar apenas o desempenho final apresentado por H-Agent, as duas sub-políticas foram treinadas antes das rodadas de teste, com os valores de hiperparâmetros determinados anteriormente. Também é importante notar que, para essa fase, β foi fixado em 5 para ambas as sub-políticas.

Os resultados da primeira rodada podem ser observados na Tabela 10, com os melhores valores por coluna destacados em verde. Após sua análise, cinco pares foram selecionados para a segunda rodada e seus resultados podem ser vistos na Tabela 11. Finalmente, O par escolhido foi $\alpha = 0,2$ e $\gamma = 0,9$ que teve o melhor desempenho em dois dos três resultados observados: rel_h e avg_q.

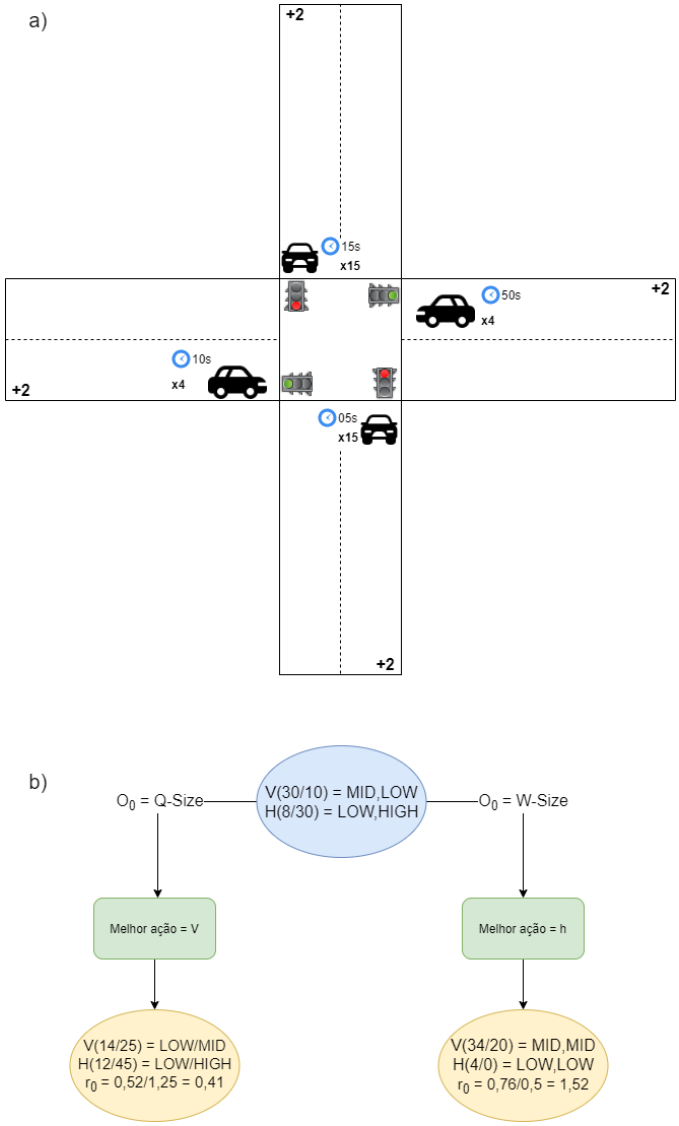

Figura 31 – Tradução do ambiente para estados de H-Agent.

3.3.4.3 Parâmetro Beta

No nível mais alto da hierarquia, ainda há um terceiro parâmetro para ser configurado, o critério de parada das sub-políticas, β. Como foi explicado na seção 2.3.1, β pode ser configurado para interromper a sub-política de três maneiras: um número máximo de ações primitivas, ou atingir um objetivo (estado) específico ou quando tornar-se mais vantajoso assumir uma sub-política diferente da corrente. Considerando que $\beta = 1$ indica que é para a sub-política ser interrompida, para o agente HRL trabalhado aqui, tem-se:

$$\beta_O = \begin{cases} 1 & se\ o\ o\ número\ de\ passos > k \\ 1 & se\ Q(s_{t+1}, o') > Q(s_{t+1}, o) \\ 0 & caso\ contrário \end{cases} \tag{3.4}$$

O número de passos tomados, k, é o único valor configurável, que pode ou não ser

Tabela 10 – Resultados de três categorias da primeira rodada de testes de hiperparâmetros de H-Agent.

α	γ	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w	rel_h
rel_h										
0,3	0,8	2939	2910	2922	12,35	18,39	17,17	17,83	0,50	163,80
0,4	0,6	2941	2907	2928,33	15,17	18,45	17,20	17,91	0,52	163,42
0,5	0,5	2934	2917	2925	6,97	18,41	17,79	18,03	0,27	162,16
avg_q										
0,2	0,9	2952	2937	2946,66	6,84	21,76	18,64	20,28	1,27	145,28
0,1	0,9	2942	2914	2930	11,77	19,18	17,68	18,43	0,61	158,97
0,4	0,6	2941	2907	2928,33	15,17	18,45	17,20	17,91	0,52	163,42
avg_w										
0,3	0,8	2939	2910	2922	12,35	18,39	17,17	17,83	0,50	163,80
0,4	0,6	2941	2907	2928,33	15,17	18,45	17,20	17,91	0,52	163,42
0,5	0,5	2934	2917	2925	6,97	18,41	17,79	18,03	0,27	162,16

Tabela 11 – Resultados da segunda rodada de testes de hiperparâmetros de H-Agent, ordenados por rel_h.

α	γ	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w	rel_h
0,2	0,9	2970	2893	2922,1	20,39	21,84	17,70	19,50	1,33	149,7931331
0,3	0,8	2953	2871	2914,4	23,65	21,60	18,14	19,48	1,11	149,5911076
0,1	0,9	2968	2886	2918,2	22,99	21,48	17,97	19,61	1,27	148,7978066
0,5	0,5	2919	2894	2905	7,29	21,79	17,73	19,52	1,25	148,7880519
0,4	0,6	2929	2893	2911,7	11,77	20,96	17,51	20,05	0,97	145,1973731

específico para cada sub-política e pode ser considerado como um terceiro hiperparâmetro e, dessa forma, também deve possuir um valor que maximize os resultados obtidos. Dessa maneira, optou-se por aplicar o mesmo conjunto de testes de definição de α e γ para β, este com um valor específico para cada sub-política, β_q e β_w. O conjunto de valores considerado foi o intervalo entre 2 e 5, com incremento de um, para cada sub-política, gerando então um universo de 16 pares possíveis, estes submetidos a primeira rodada de testes, agora com os hiperparâmetros $\alpha = 0,2$ e $\gamma = 0,9$ fixos, que gerou os resultados observados na Tabela 12, com os melhores valores por coluna destacado em verde.

Ficando, então, cinco pares para a segunda rodada, onde seus resultados podem ser vistos na Tabela 13. Logo, os valores de β das sub-políticas Q-Size e W-Size ficaram em 5 e 2, respectivamente.

3.3.4.4 Resultados

H-Agent foi submetido aos mesmos cenários que TF e as sub-políticas: Fixo-1800, Pico-2700 e Pico-3600, com seus resultados, também sendo tirados da média de 10 ciclos de teste e treino, reunidos na Tabela 14, com os melhores resultados destacados em verde e os piores em vermelho. Bem como na escolha dos hiperparâmetros, as sub-políticas foram

Tabela 12 – Resultados de três categorias da primeira rodada de testes de β para H-Agent.

β_q	β_w	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w	rel_h
rel_h										
5	2	2940	2892	2921	20,83	19,24	18,55	18,98	0,30	153,85
4	5	2952	2872	2912	32,65	21,09	18,32	19,28	1,27	150,98
3	2	2963	2900	2928,66	26,02	20,48	18,79	19,77	0,71	148,13
avg_q										
2	4	2959	2921	2934	17,68	22,90	19,64	21,22	1,33	138,21
4	4	2955	2901	2931,33	22,54	22,03	19,64	20,62	1,02	142,11
3	2	2963	2900	2928,66	26,02	20,48	18,79	19,77	0,71	148,13
avg_w										
5	2	2940	2892	2921	20,83	19,24	18,55	18,98	0,30	153,85
4	5	2952	2872	2912	32,65	21,09	18,32	19,28	1,27	150,98
3	2	2963	2900	2928,66	26,02	20,48	18,79	19,77	0,71	148,13

Tabela 13 – Resultados da segunda rodada de testes de β para H-Agent, ordenados por rel_h.

β_q	β_w	max_q	min_q	avg_q	std_q	max_w	min_w	avg_w	std_w	rel_h
5	2	2964	2908	2936,33	14,58	20,57	17,47	19,09	1,04	153,80
4	4	2960	2914	2939	17,98	21,14	18,04	19,94	0,94	147,33
4	5	2935	2833	2889,3	30,13	22,07	17,66	20,16	1,57	143,31
3	2	2955	2789	2900,8	44,71	23,19	18,72	20,86	1,53	139,02
2	4	2923	2852	2890,9	21,20	22,29	18,94	20,90	0,86	138,25

treinadas separadamente de H-Agent, logo, os resultados demonstrados aqui são fruto apenas de testes repetidos na política macro do H-Agent.

Novamente, os valores de fluxo total e vazão por minuto observados para Fixo-1800 e exibidos nas Figuras 32 e 33 se mostram próximos entre as abordagens, destacando-se apenas o tempo médio de espera, onde H-Agent se coloca num desempenho entre Q-Size e W-Size, sendo 23,9% mais eficiente que o primeiro e 11,3% menos eficiente que o segundo. Aqui, o gráfico de tempo de espera é mais estável que o observado em Q-Size (Figura 19) e semelhante ao do W-Agent (Figura 26), o que reflete seu tempo médio final, mais próximo do segundo.

A partir do cenário Pico-2700, é possível ver a diferença de desempenho que H-Agent traz para o agente, com seus resultados sendo um misto entre as duas sub-políticas anteriores, onde a vazão é virtualmente igual Q-Size, (2428 contra 2433), com o tempo de espera se aproximando mais de W-Size, com uma diferença de 10,28%. A análise das Figuras 34 e 35 corrobora com o que foi observado na Tabela 14, onde o desempenho para vazão assemelha-se a Q-Size, com subidas mais acentuadas em situação de demanda maior e atendendo todo o pico da primeira demanda aos 38 minutos, 1 minuto antes de W-Size, enquanto o tempo de espera aproxima-se de W-Size, com variações de acordo

Tabela 14 – Resultados para a aplicação do agente W-Size, Q-Size e TF aos cenários de teste

Cenário	V_s	V_m	W
H-Agent			
Fixo-1800	1781	29,68	19,08
Pico-2700	2428	40,47	21,01
Pico-3600	2923	48,73	22,8
W-Size			
Fixo-1800	1782	29,23	17,14
Pico-2700	2396	39,93	19,05
Pico-3600	2826	47,03	23,21
Q-Size			
Fixo-1800	1779	29,66	25,09
Pico-2700	2433	40,56	25,32
Pico-3600	2929	48,83	27,79
TF			
Fixo-1800	1784	29,73	16,63
Pico-2700	2349	39,15	29,34
Pico-3600	2420	40,33	31,53

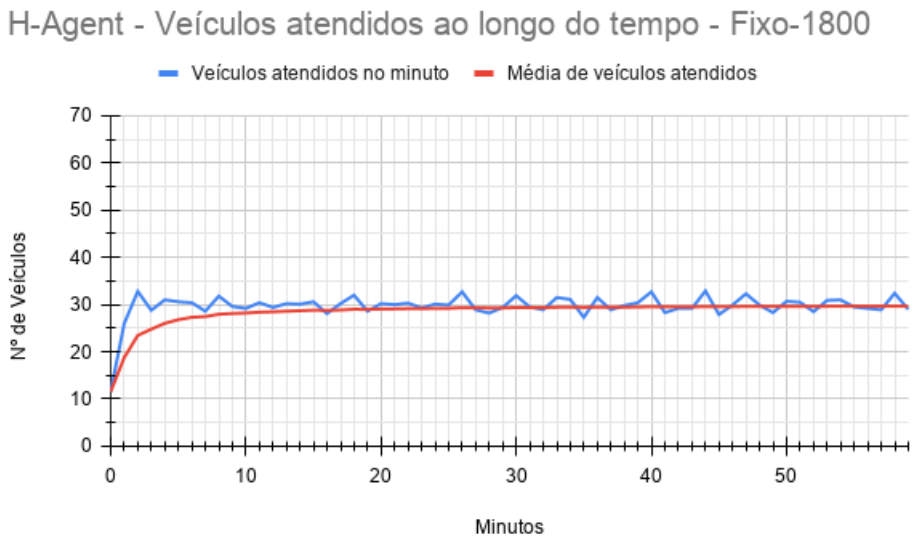

Figura 32 – Resultados de fluxo para H-Agent em Fixo-1800

com a demanda, porém sempre mais próximo à média, indício de uma estabilidade maior.

Ao se comparar os resultados para Pico-3600, a capacidade de H-Agent de sintetizar, em um só agente, os desempenhos de Q-Size e W-Size fica mais evidente. Mais uma vez, seu resultado para fluxo total, quando comparado a Q-Size, é semelhante, 2923 contra 2929, mas também capaz de atingir valores maiores por mais tempo, como, por exemplo, entre os minutos 21 e 30 ou a partir dos 50 minutos, onde H-Agent apresenta a vazão maior e mais estável das três abordagens. Ainda, diferente do que aconteceu para

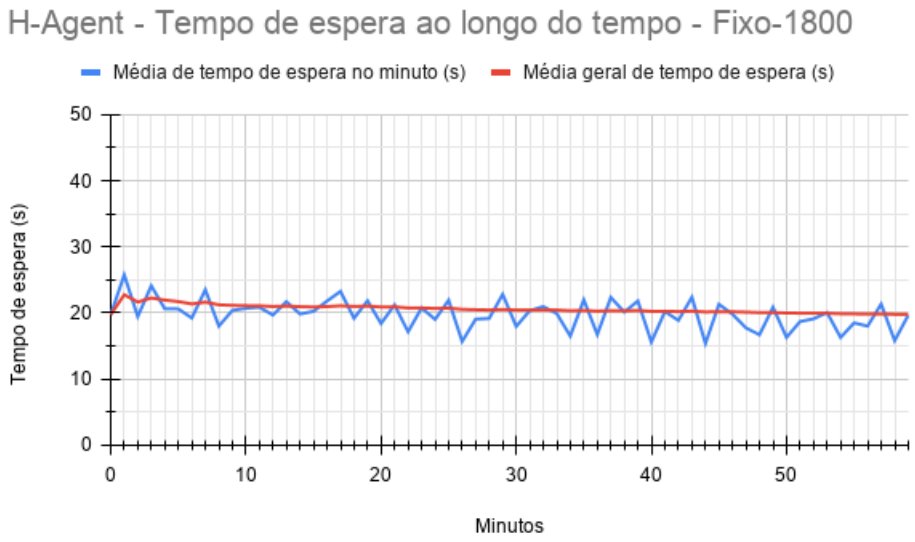

Figura 33 – Resultados de tempo de espera para H-Agent em Fixo-1800

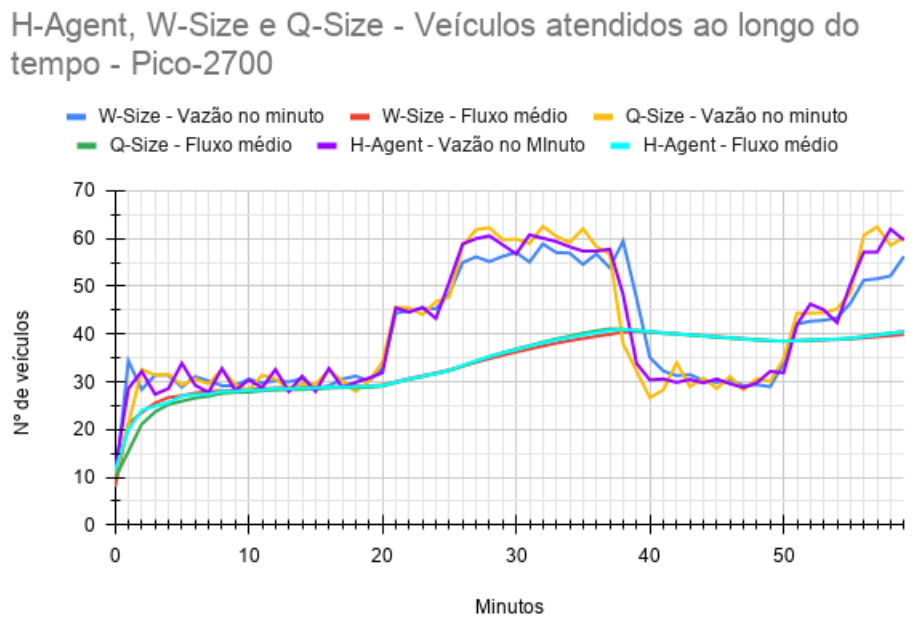

Figura 34 – Resultados de fluxo para H-Agent em Pico-2700

o Pico-2700, agora H-Agent não só apresenta um tempo de espera próximo a W-Size, seu resultado é também 1,76% menor e com picos de perda de desempenho menores e menos frequentes, como visto no minuto 52. Apesar de ser possível considerar que o resultado está dentro da variação de desempenho de ambos, o importante é notar que H-Agent conseguiu obter os melhores resultados de cada sub-política, sem sacrificar o que ambas renunciaram, atingindo o mesmo resultado de vazão que Q-Size sem perder desempenho no tempo de espera médio e vice-versa, obtendo o menor tempo de espera sem minar a vazão final. Novamente, a análise dos gráficos da Figuras 36 e 37 corrobora com o re-

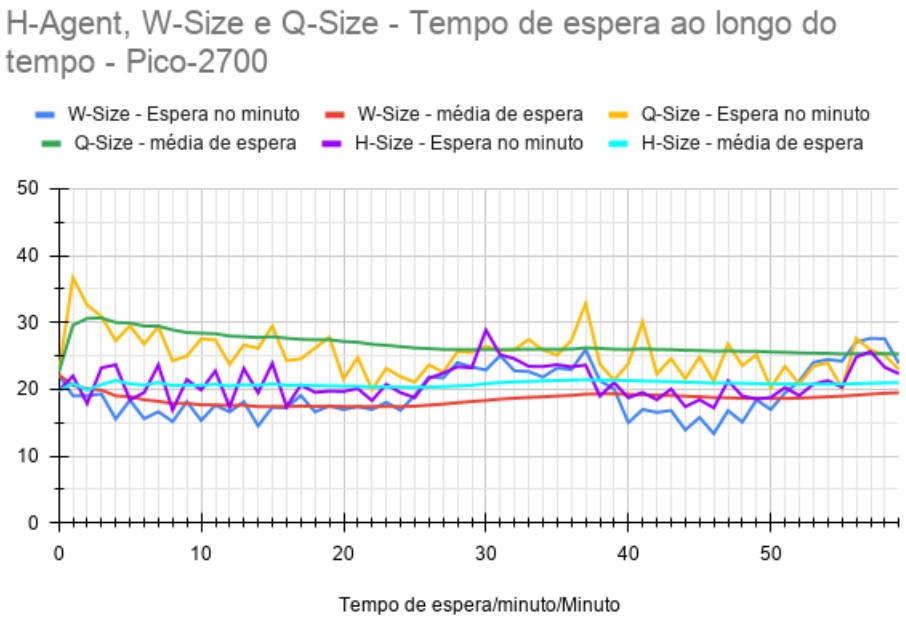

Figura 35 – Resultados de tempo de espera para H-Agent em Pico-2700

sultado, mostrando como H-Agent aproxima-se das curvas de suas sub-políticas em suas respectivas especialidades.

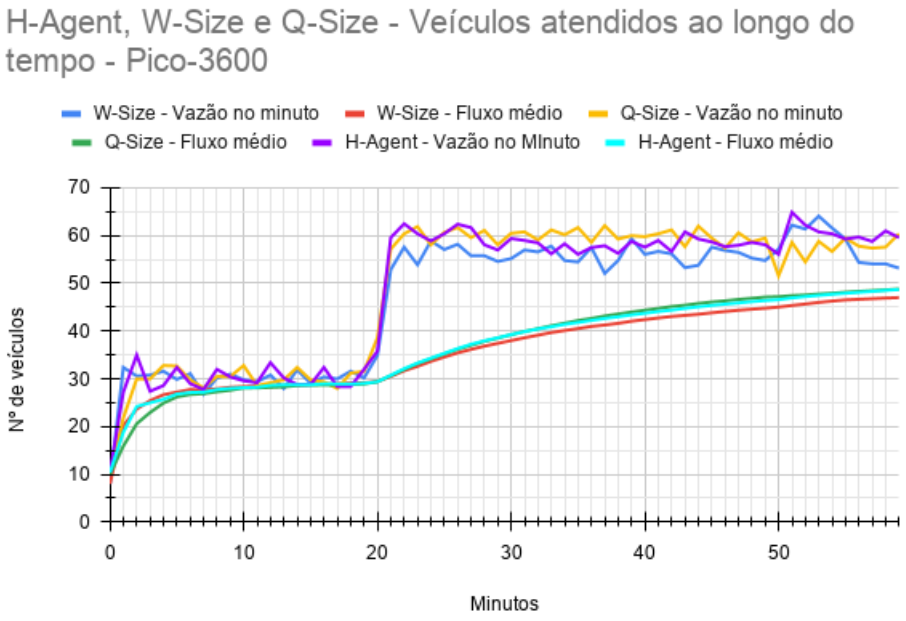

Figura 36 – Resultados de fluxo para H-Agent em Pico-3600

Os valores apresentados por H-Agent demonstram que, a partir de HRL, é possível combinar, no caso do presente estudo, duas sub-políticas e gerar valores superiores aos seus desempenhos isolados, mesmo em seu campo de especialidade. Isto se deve ao fato de o agente *Hierarchical Reinforcement Learning*, guiado pelo seu modelo de recompensa,

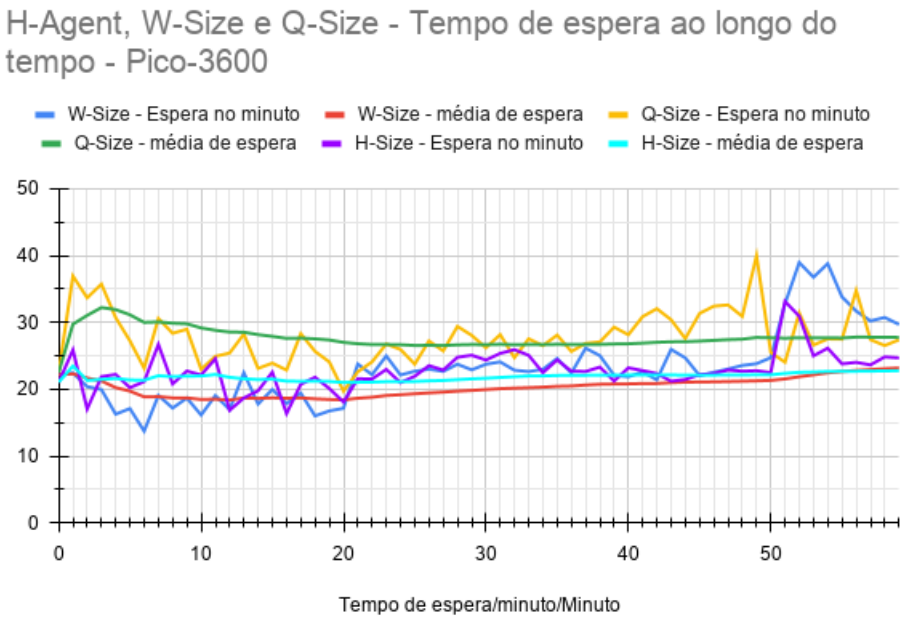

Figura 37 – Resultados de tempo de espera para H-Agent em Pico-3600

busca iniciar suas políticas de menor hierarquia em momentos (estados) em que o máximo de desempenho pode ser extraído delas. Dessa maneira, a perda que cada uma gera, por ter apenas uma maneira de ler e interagir com o ambiente, é não só mitigada mas também suplantada pelas capacidades da outra, criando, por fim, um agente mais estável e capaz.

3.4 Semáforo de Tempo Fixos e H-Agent

A comparação final de desempenho se dá, então, entre o Semáforo de Tempos Fixos (TF) e o agente *Hierarchical Reinforcement Learning*, H-Agent, objeto final da proposta apresentada neste trabalho. Embora parte dos resultados apresentados aqui sejam, de fato, um retrospecto do que foi apresentado nas respectivas seções de cada abordagem, nas seções 3.2 e 3.3.4, seus resultados ainda não foram confrontados através de gráficos.

Os resultados de Fixo-1800 serão suprimidos porque, embora haja variações, principalmente no tempo de espera, como pode ser observado na Tabela 14, as curvas apresentam resultados similares, logo, não trazem comparações pertinentes à discussão.

Começando por Pico-2700, é possível ver na Figura 38 como o agente *Hierarchical Reinforcement Learning*, mesmo com tempos de verde menores (10 segundos contra 15 segundos), consegue adaptar-se a demanda crescente da via. No entanto, apesar da diferença de 50% entre os tempos de via aberta, H-Agent é capaz de selecionar uma mesma via vezes seguidas, o que, de fato, altera o tempo de verde da via em questão, em incrementos de 10 segundos. É justamente essa característica de não estar atado a intercalar o atendimento das vias, independente do fluxo, como ocorre com TF, que faz com que o

sistema torne-se mais capaz.

No entanto, é válido questionar que, se o agente pode escolher seguidas vezes uma única via, logo a via oposta verá seu tempo de espera subir, puxando consigo o tempo de espera total do cruzamento e, de fato, é isto que se observa quando se usa apenas Q-Size como política: este é capaz de elevar a vazão, mas aumenta o tempo de espera. Porém, como o H-Agent tem como opção assumir o comportamento de W-Size, este que tem uma abordagem inversa a de Q-Size, ele consegue suprimir a deficiência do primeiro. Logo, ao se observar a Figura 39 se pode ver que, mesmo com um fluxo maior que TF, H-Agent mantém o tempo de espera estável e mais baixo, com valores próximos a 21 segundos, salvo alguns picos, como o observado em 30 minutos, durante toda a simulação.

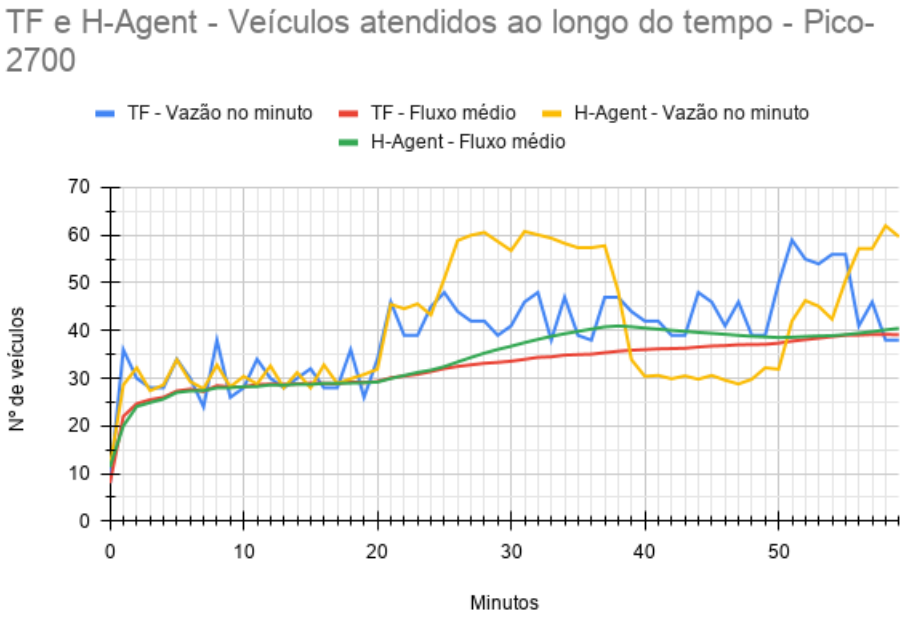

Figura 38 – Resultados de fluxo para H-Agent e TF em Pico-2700

Uma segunda comparação pertinente a se fazer é verificar quanto tempo, após a duração de 60 minutos de simulação, cada uma das duas abordagens gasta para esvaziar o cruzamento, se observando também como isso se reflete nos tempos de média e de espera. Para obter tais resultados, uma segunda e única execução dos cenários foi feita em TF e H-Agent. É possível notar que os primeiros 60 minutos de TF gera os mesmos resultados observados anteriormente, já que o simulador garante geração de fluxo determinística e TF possui comportamento fixo. No entanto, para H-Agent, isso não se repete. Como o agente possui comportamento estocástico nos momentos de exploração, provindos da dinâmica explorar versus aproveitar, seus resultados divergem dos observados anteriormente, porém, isso não prejudica a proposta de comparação.

Através das Figuras 40 e 41, é possível verificar como não só TF demora ainda mais 13 minutos para esvaziar todo o fluxo gerado, $\frac{1}{5}$ do tempo proposto pelo cenário,

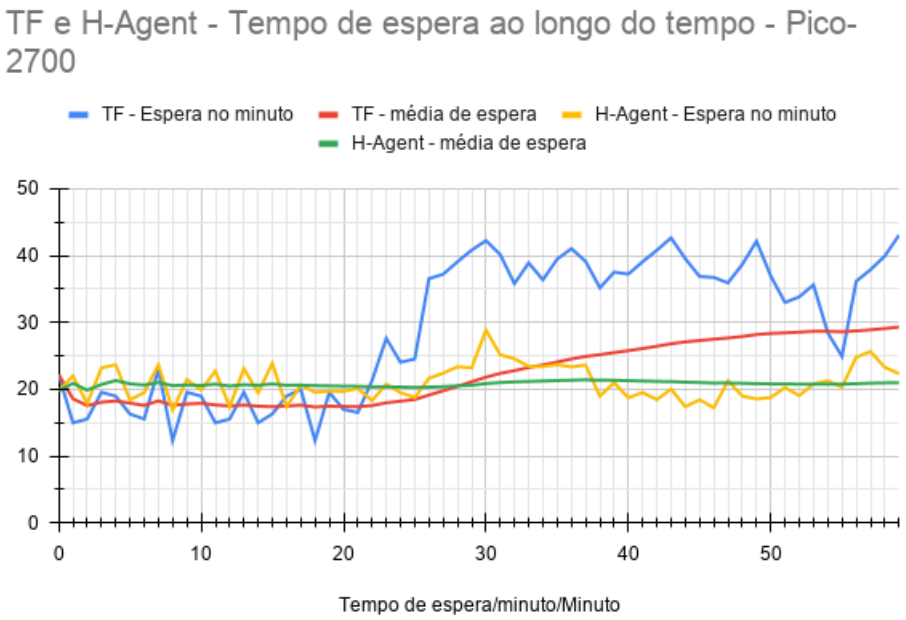

Figura 39 – Resultados de tempo de espera para H-Agent e TF em Pico-2700

como também vê seu tempo de espera continuar degradando-se além da marca de 60 minutos, com um pico de mais de um minuto e meio no final da simulação, reflexo de sua inabilidade de esvaziar as vias com maior demanda. O H-Agent, por outro lado, esvazia o cruzamento aos 65 minutos, oito minutos antes de TF e, mais importante, o faz sem aumentar o tempo de espera, que diminui nos minutos finais.

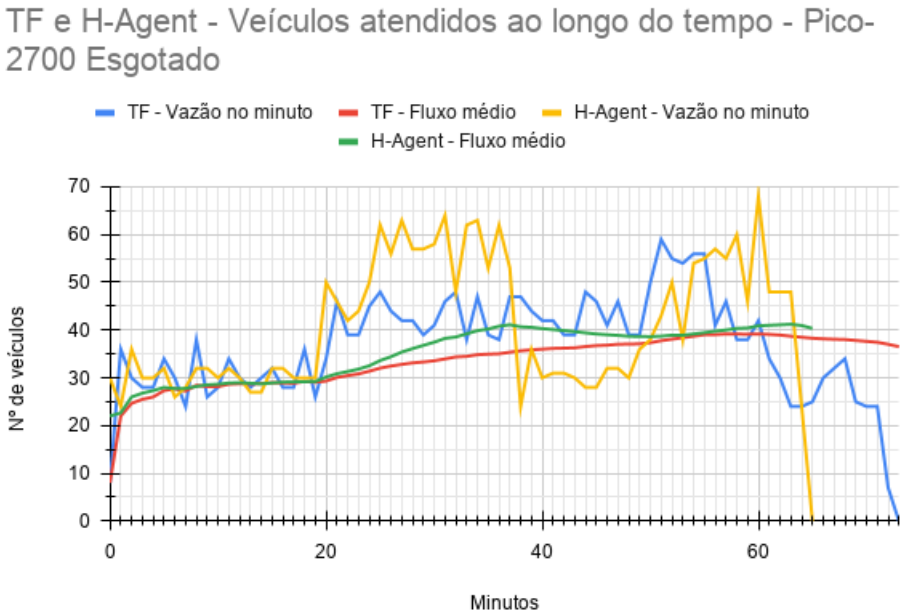

Figura 40 – Resultados de fluxo para H-Agent e TF em Pico-2700 até esvaziar o cruzamento

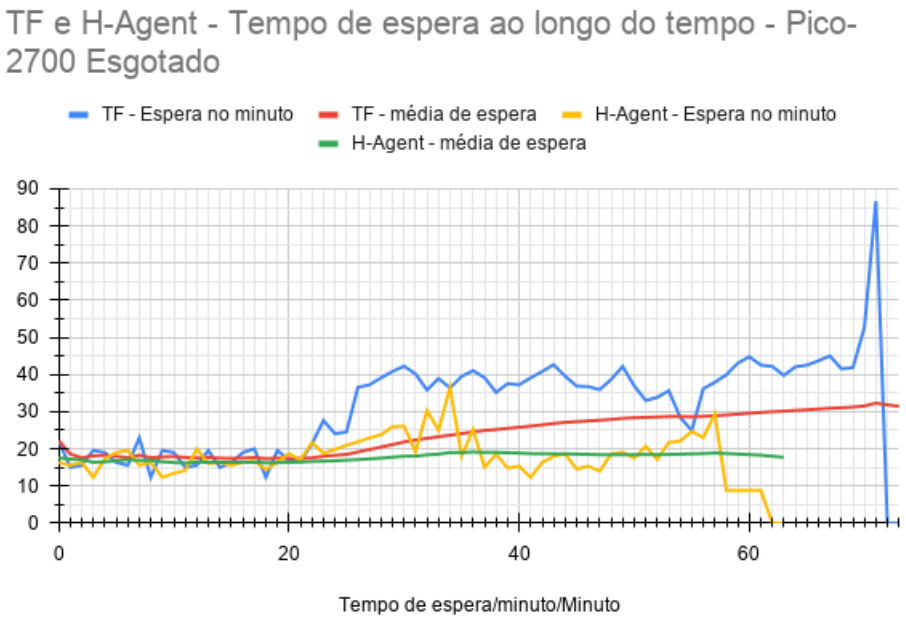

Figura 41 – Resultados de tempo de espera para H-Agent e TF em Pico-2700 até esvaziar o cruzamento

O mesmo comportamento visto em Pico-2700 repete-se em Pico-3600, ilustrado nas Figuras 42 e 43. A principal diferença são os valores em si, o fluxo maior impede H-Agent de atender toda a demanda dentro do tempo de simulação, porém, a média de vazão apresenta tendência de subida, indicando que nos minutos pós barreira de 60, o cruzamento deve ser esvaziado. Ainda, H-Agent consegue manter a estabilidade no tempo de espera, enquanto TF degrada-se a partir do primeiro surto.

Já quanto ao tempo pós cenário gasto por cada agente, ilustrado nas Figuras 44 e 45, obviamente há um aumento, onde TF precisa de 29 minutos adicionais, metade do tempo de simulação proposto, para conseguir esvaziar a demanda, enquanto H-Agent precisa de mais 16 minutos, não muito além dos 13 minutos observados no cenário anterior para TF. A queda rápida no tempo de espera em H-Agent, pós 60 minutos, indica que o fluxo consumido nos 15 minutos extras é atendido de forma bem distribuída, sem clara preferência por uma via, com exceção do pico logo antes do final. Esse mesmo pico ocorre com TF, embora com uma defasagem de 15 minutos.

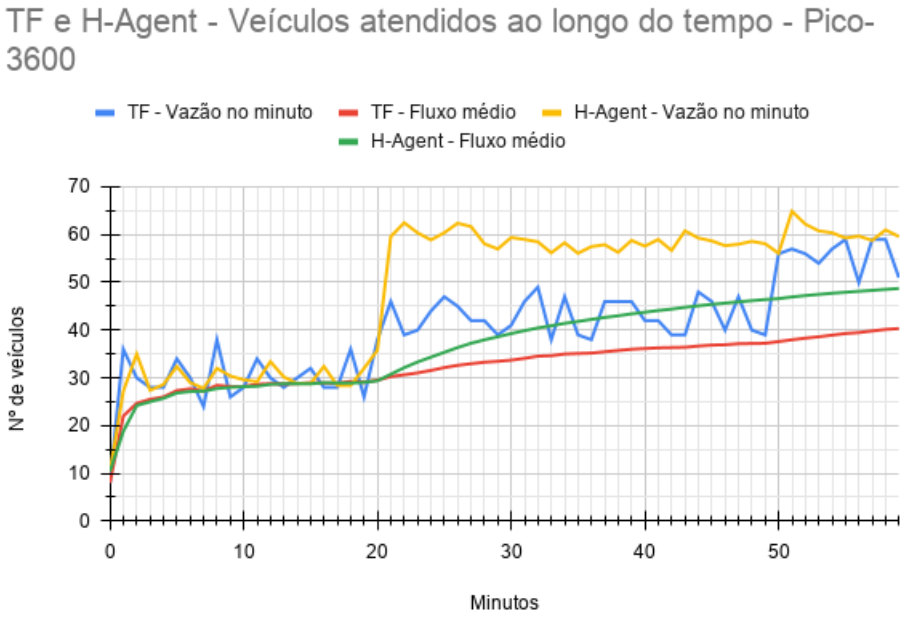

Figura 42 – Resultados de fluxo para H-Agent e TF em Pico-3600

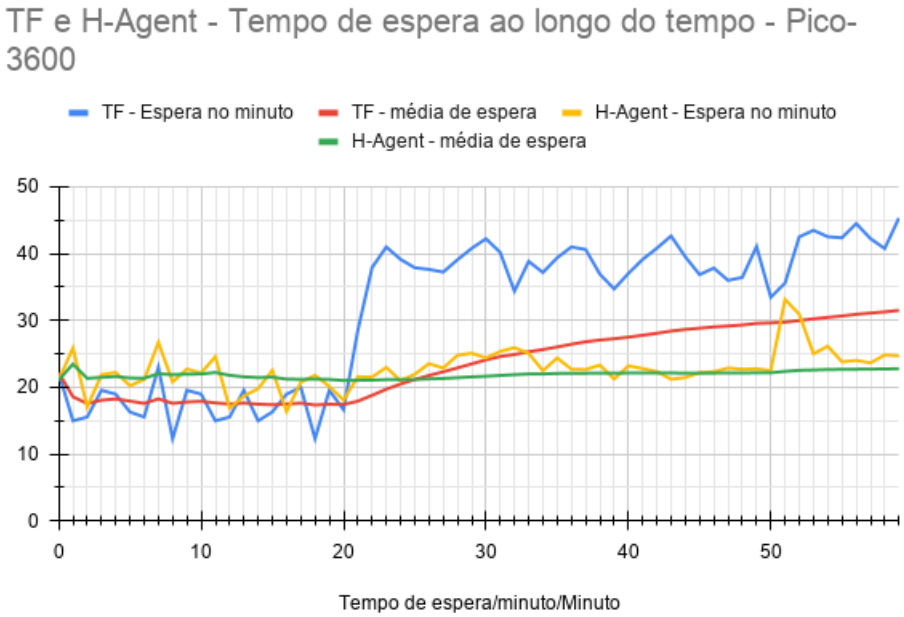

Figura 43 – Resultados de tempo de espera para H-Agent e TF em Pico-3600

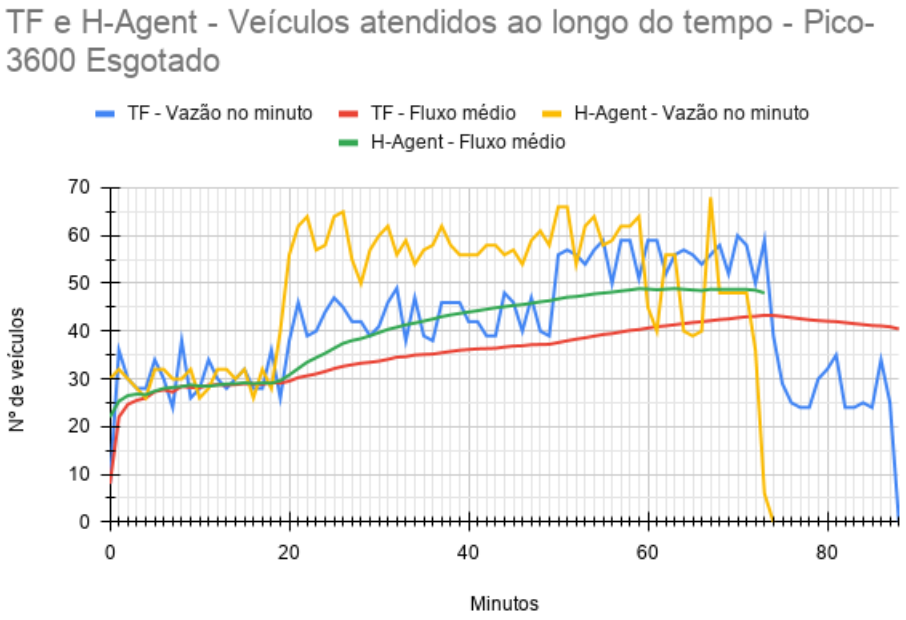

Figura 44 – Resultados de fluxo para H-Agent e TF em Pico-3600 até esvaziar o cruzamento

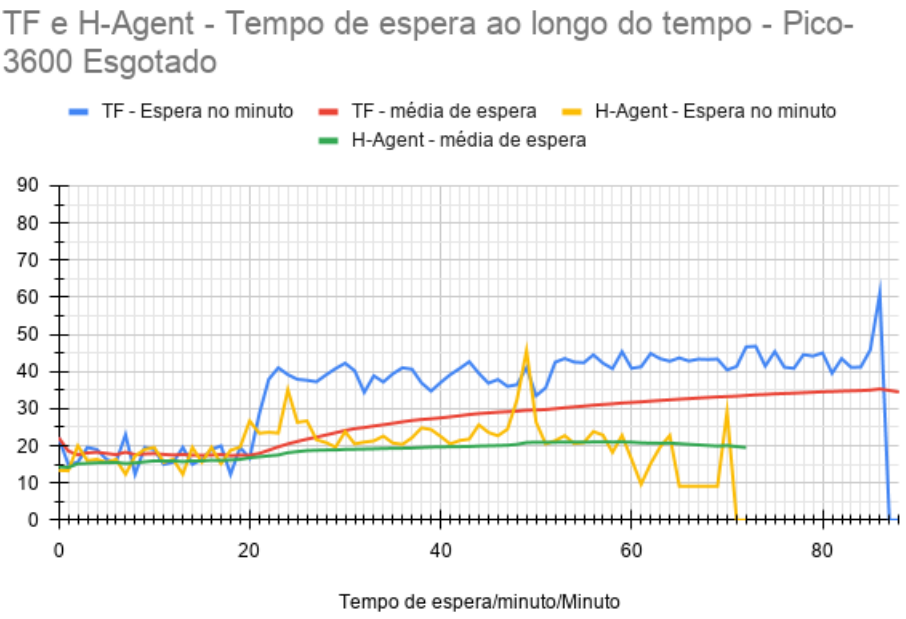

Figura 45 – Resultados de tempo de espera para H-Agent e TF em Pico-3600 até esvaziar o cruzamento

4 Conclusão

Um cruzamento de vias sinalizadas, essencialmente, busca não só garantir passagem segura para os veículos, mas também garantir que todos sejam atendidos de forma organizada e bem distribuída (DENATRAN, 1984). São justamente essas duas características que o Método de Webster, abordagem mais comum para o cálculo de tempos em um semáforo, se propõe a solucionar, determinando os tempos de abertura de cada uma das vias, de acordo com a demanda de veículos. No entanto, o fluxo de veículos em uma via possui, naturalmente, um perfil estocástico, isto é, a característica do fluxo é largamente afetada por eventos aleatórios, intrínsecos ao próprio ambiente e processo, e que podem mudar radicalmente o comportamento das filas de veículos em um cruzamento, mesmo por longos períodos de tempo. Esse traço vai de encontro a maneira que o Método de Webster realiza seus cálculos que, embora considere valores imutáveis, como a largura da via e a velocidade máxima permitida, se vê também embasado em valores que podem se alterar a qualquer momento, como o fluxo de veículos máximo que pode ocorrer no cruzamento em questão. Dessa maneira, uma vez efetuado o cálculo, ele estará permanentemente dimensionado para o fluxo considerado no momento de suas concepção e jamais se alterará. Sendo assim, ataca-se com uma abordagem determinística um problema que é intrinsecamente variável.

Uma nova abordagem, então, é exigida, capaz de adaptar-se melhor ao problema proposto e, como apresentado por Bielli et al. (1991), Zhao, Dai e Zhang (2012) e Yau et al. (2017), uma dessas abordagens é a Inteligência Artificial (IA), apta a tomar decisões a partir da dados do ambiente, de forma dinâmica. Com IA é possível, então, atacar um problema naturalmente dinâmico com uma ferramenta capaz de adaptar-se as variações. Como pôde ser observado ao longo do estudo, *Reinforcement Learning* possui um conjunto de capacidades que se encaixam ao problema observado, sendo que destas destacam-se seu modelo de aprendizado *online*, que garante aprendizado à medida que os dados chegam, com o agente já inserido no ambiente, e seu perfil *model-free*, que indica que o agente não precisa de um modelo completo e conhecido do processo para poder inferir seu comportamento e tomar decisões.

O simples fato de utilizar uma abordagem com capacidade de aprendizado e adaptação num processo dinâmico, já possibilita a obtenção de resultados melhores quando comparados a um modelo determinístico, mas sem a garantia de que o modelo escolhido seja realmente ideal para o problema em questão. Porém, como se pôde observar ao comparar não só *Hierarchical Reinforcement Learning* e o Método de Webster, mas também suas sub-políticas entre si, cada uma delas uma política *Q-Learning* completa, foi visto que cada abordagem maximiza os resultados em direção a um tipo de desempenho, que

diferem entre si em sua efetividade, mas todas apresentam melhorias significativas, entre 20% e 30%, em relação ao Método de Webster.

Sendo assim, o uso de *Reinforcement Learning* e seu *framework* baseado em Processos Decisórios de Markov, mostra-se acertado para o gerenciamento de tráfego veicular, como demonstrado por Yau et al. (2017) e reafirmado pelas sub-políticas desenvolvidas aqui. Ao se elevar o RL básico para a abordagem de dividir para conquistar do *Hierarchical Reinforcement Learning*, se observa que é possível atacar mais de um perfil do problema em questão sem sacrificar o desempenho de outros indicadores, mesmo aqueles que possam apresentar comportamento conflituoso entre si, dando um equilíbrio maior ao resultado final. Mais do que isso, foi possível alcançar tal rendimento com um modelo relativamente simples, onde, ao se somar as três políticas trabalhadas, totalizando 117 estados, traduzidos em 234 pares $Q(s, a)$, que ocupam, por sua vez, 234 células de uma matriz. Somando a simplicidade do modelo com o fato de que os hiperparâmetros podem ser otimizados de forma mais profunda e acentuada, se passa a ter em mãos um modelo não só capaz de suprir o primeiro desafio proposto aqui, mas também com espaço de expansão para abordar problemas de magnitudes maiores.

4.1 Considerações Finais e Trabalhos Futuros

Embora a ferramenta SUMO ofereça uma gama considerável de variáveis a se configurar, o trabalho se limitou apenas aos seus valores padrão. Há a possibilidade de configurar o simulador para mimetizar completamente um cruzamento real, caso hajam dados suficientes, colocando o agente a prova de situações mais complexas. O mesmo se aplica na comparação do estudo feito aqui a trabalhos de terceiros, caso estes disponibilizem as variáveis chave de simulação para que uma comparação justa possa ser realizada.

Não se pode, também, desconsiderar que o processo analisado aqui foi resumido em um único cruzamento que, apesar de suprir a necessidade básica de gerenciamento de trânsito e representar a gama majoritária de casos reais, tem implicações a níveis macro, ao se envolver no planejamento todo o complexo de vias da cidade. Sendo assim, um próximo passo também poderia ser o de evoluir o modelo para uma abordagem multi-agente, onde cada cruzamento do sistema, regido por um agente local, troca informações e considera os estados de seu vizinho para tomar a decisão final, abrindo também a oportunidade para a adição de novos níveis hierárquicos.

Apesar de ter-se comparado *Hierarchical Reinforcement Learning* (HRL) com um semáforo de tempos fixos (TF), não se pode ignorar que, em momentos em que o cruzamento gerenciado possui fluxo padrão, acordado com os valores utilizados no Método de Webster, como o cenário Fixo-1800, o semáforo tradicional se mostra capaz de atender com excelência a demanda observada, com alto desempenho e baixa variação. Sendo assim,

feitas as devidas adaptações, HRL poderia utilizar TF como uma de suas sub-políticas, o que potencialmente diminuiria suas flutuações, devido a exploração, em momentos de baixa demanda. Não apenas isso, a partir dessa ideia, é possível expandir HRL para que este possua sub-políticas específicas e de comportamento fixo, como TF, mas que supram objetivos determinísticos, como por exemplo, uma sub-política que dê preferência para uma via que contenha um veículo oficial, como ambulância ou bombeiros, ou algum outro que possua prioridade de acordo com a engenharia de tráfego da cidade, como ônibus.

Também se deve considerar que o modelo final traz o questionamento de como a mesma abordagem se comportaria caso se desse a ela maiores capacidades de interpretação e tomada de decisão perante o processo. Uma evolução natural seria substituir o modelo tabular e buscar apoio nas Redes Neurais Artificiais, como proposto por Kulkarni et al. (2016). Sem a limitação do espaço de memória, seria possível abandonar, por exemplo, a tradução das filas e tempos de espera em classes e passar a usar seus valores absolutos, além de adicionar novas *features*, como velocidade média dos veículos e data (dia da semana e mês) e hora das observações. Mais do que isso, também seria possível tornar as ações mais assertivas que, ao invés de apenas indicar qual via deve ser aberta por um tempo fixo de tempo, também determinaria o valor desse tempo.

A partir do momento que se aumenta a complexidade do agente HRL e se dá a ele poder computacional, é possível também considerar abordagens mais arrojadas como, por exemplo, fazer com que as *options* iniciem algoritmos de *Reinforcement Learning* complexos e de alto desempenho, que serão iniciadas no momento mais oportuno e de máximo retorno. Em suma, HRL cria oportunidades variadas e únicas de abordar um problema já conhecido e estudado, mas que ainda não foi devidamente solucionado.

Referências

AGGARWAL, C. C. *Data Mining*. [S.l.]: Springer-Verlag GmbH, 2015. ISBN 9783319141428. 74

AGGARWAL, C. C. *Neural Networks and Deep Learning*. [S.l.]: Springer-Verlag GmbH, 2018. ISBN 9783319944630. 31

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*, Elsevier BV, v. 85, p. 732–752, dec 2017. 42, 43

BARTO, A. G.; MAHADEVAN, S. Recent advances on hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, Springer Science and Business Media LLC, v. 13, n. 4, p. 341–379, 2003. 32

BELLMAN, R. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, Proceedings of the National Academy of Sciences, v. 38, n. 8, p. 716–719, aug 1952. 25

BELLMAN, R. *Dynamic programming*. Princeton, N.J: Univ. Pr, 1957. ISBN 9780691079516. 30

BELLMAN, R. A markovian decision process. *Indiana Univ. Math. J.*, v. 6, p. 679–684, 1957. ISSN 0022-2518. 20, 24

BEN-DAVID, S.; KUSHILEVITZ, E.; MANSOUR, Y. Machine learning. *Springer Science and Business Media LLC*, Springer Science and Business Media LLC, v. 29, n. 1, p. 45–63, 1997. 30

BENGIO, Y.; COURVILLE, A.; VINCENT, P. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Institute of Electrical and Electronics Engineers (IEEE), v. 35, n. 8, p. 1798–1828, aug 2013. 31

BIELLI, M. et al. Artificial intelligence techniques for urban traffic control. *Transportation Research Part A: General*, Elsevier BV, v. 25, n. 5, p. 319–325, sep 1991. 17, 43, 88

BOLDRINI, J. L. *Algebra Linear*. [S.l.]: HARBRA, 1986. 19

BRADTKE, S.; DUFF, M. Reinforcement learning methods for continuous-time markov decision problems. 12 1994. 34

CASTRO, G. B.; HIRAKAWA, A. R.; MARTINI, J. S. Adaptive traffic signal control based on bio-neural network. *Procedia Computer Science*, Elsevier BV, v. 109, p. 1182–1187, 2017. 44

DAYAN, P.; HINTON, G. E. Feudal reinforcement learning. In: HANSON, S. J.; COWAN, J. D.; GILES, C. L. (Ed.). *Advances in Neural Information Processing Systems 5*. [S.l.]: Morgan-Kaufmann, 1993. p. 271–278. 32

DEARDEN, R.; FRIEDMAN, N.; RUSSELL, S. Bayesian q-learning. In: *Proceedings of the Fifteenth National/Tenth Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence*. USA: American Association for Artificial Intelligence, 1998. (AAAI '98/IAAI '98), p. 761–768. ISBN 0262510987. 43

DENATRAN. *Manual de Semáforos*. [S.l.]: Departamento Nacional de Trânsito, 1984. 16, 38, 39, 46, 48, 55, 63, 88

DENATRAN. *Frota de Veículos*. 2020. Disponível em: <http://www.denatran.gov.br/estatistica/237-frota-veiculos>. 16

DIETTERICH, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. 1999. 32

DIFEBBRARO, A.; GIGLIO, D.; SACCO, N. Urban traffic control structure based on hybrid petri nets. *IEEE Transactions on Intelligent Transportation Systems*, Institute of Electrical and Electronics Engineers (IEEE), v. 5, n. 4, p. 224–237, dec 2004. 16

EKEOCHA, R. J. O.; IHEBOM, V. I. The use of queuing theory in the management of traffic intensity. *International Journal of Sciences*, Alkhaer Publications, v. 4, n. 03, p. 56–63, 2018. 16

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*, Institute of Electrical and Electronics Engineers (IEEE), v. 14, n. 3, p. 1140–1150, sep 2013. 42

ERTEL, W. *Introduction to Artificial Intelligence*. [S.l.]: Springer-Verlag GmbH, 2018. ISBN 3319584863. 17

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, Massachusetts, USA: MIT Press, 2016. 16, 31

HAYKIN, S. O. *Neural Networks and Learning Machines (3rd Edition)*. Upper Saddle River, New Jersey, USA: Pearson, 2008. ISBN 0131471392. 31

HENGST, B.; SAMMUT, C.; WEBB, G. I. Hierarchical reinforcement learning. In: _____. *Encyclopedia of Machine Learning*. Boston, MA: Springer US, 2010. p. 495–502. ISBN 978-0-387-30164-8. 17, 32

JIN, J.; MA, X. Adaptive group-based signal control by reinforcement learning. *Transportation Research Procedia*, Elsevier BV, v. 10, p. 207–216, 2015. 44

KUBAT, M. *An Introduction to Machine Learning*. [S.l.]: Springer-Verlag GmbH, 2017. ISBN 3319639129. 17

KULKARNI, T. D. et al. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. 2016. 90

LAPAN, M. *Deep Reinforcement Learning Hands-On*. [S.l.]: Packt Publishing, 2018. ISBN 1788834240. 25, 30, 31, 41

LARSON, R. *Elementary statistics : picturing the world*. Boston: Pearson, 2015. ISBN 9781292058610. 46

LEVIN, D. A.; PERES, Y.; WILMER, E. L. *Markov Chains and Mixing Times*. [S.l.]: American Mathematical Society, 2008. ISBN 0821847392. 19, 20

LI YISHENG LV, F.-Y. W. L. Traffic signal timing via deep reinforcement learning. *IEEE/CAA Journal of Automatica Sinica*, Institute of Electrical and Electronics Engineers (IEEE), v. 3, n. 3, p. 247–254, jul 2016. 41

LOPEZ, P. A. et al. Microscopic traffic simulation using sumo. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. [S.l.]: IEEE, 2018. p. 2575–2582. 44

MARKOV, A. Extension of the Limit Theorems of Probability Theory to a Sum of Variables Connected in a Chain. In: HOWARD, R. (Ed.). *Dynamic Probabilistic Systems (Volume I: Markov Models)*. New York City: John Wiley & Sons, Inc., 1907. cap. reprinted in Appendix B, p. 552–577. 19

MNIH, V. et al. Playing atari with deep reinforcement learning. 2013. 31

NVIDIA. *Nvidia Jetson Nano*. 2019. Disponível em: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>. 43

OTTERLO, M. van; WIERING, M. Reinforcement learning and markov decision processes. In: *Adaptation, Learning, and Optimization*. [S.l.]: Springer Berlin Heidelberg, 2012. p. 3–42. 29

RAVICHANDIRAN, S. *Hands-On Reinforcement Learning with Python*. Birmingham, United Kingdom: Packt Publishing, 2018. ISBN 1788836529. 31

RIBAS-FERNANDES, J. J. et al. A neural signature of hierarchical reinforcement learning. *Neuron*, Elsevier BV, v. 71, n. 2, p. 370–379, jul 2011. 7, 33

RIEDEL, T.; BRUNNER, U. Traffic control using graph theory. *Control Engineering Practice*, Elsevier BV, v. 2, n. 3, p. 397–404, jun 1994. 16

RUMMERY, G.; NIRANJAN, M. On-line q-learning using connectionist systems. *Technical Report CUED/F-INFENG/TR 166*, 11 1994. 29

SILVER, D. et al. Mastering the game of go without human knowledge. *Nature*, Springer Science and Business Media LLC, v. 550, n. 7676, p. 354–359, oct 2017. 31

SKANSI, S. *Introduction to Deep Learning*. [S.l.]: Springer-Verlag GmbH, 2018. ISBN 9783319730042. 31

SUMO. *Simulation of Urban Mobility*. 2020. Disponível em: <https://www.eclipse.org/sumo/>. 44

SUMO. *Traffic Control Interface*. 2020. 44

SUTTON, R. S. Learning to predict by the methods of temporal differences. *Machine Learning*, Springer Science and Business Media LLC, v. 3, n. 1, p. 9–44, aug 1988. 29

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning*. [S.l.]: MIT Press Ltd, 1998. ISBN 0262193981. 7, 17, 21, 24, 25, 26, 29

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. [S.l.]: A Bradford Book, 2018. ISBN 0262039249. 7, 17, 21, 22, 23, 24, 27, 28

SUTTON, R. S.; PRECUP, D.; SINGH, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, Elsevier BV, v. 112, n. 1-2, p. 181–211, aug 1999. 7, 32, 33, 34, 35, 36, 37

TAN, M. K. et al. Hierarchical multi-agent system in traffic network signalization with improved genetic algorithm. In: *2018 IEEE International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*. Kota Kinabalu, Malaysia, Malaysia: IEEE, 2018. p. 1–6. 41

TENSORFLOW. *TensorFlow*. 2019. Disponível em: <https://www.tensorflow.org/>. 43

TIAN, D. et al. Swarm intelligence inspired adaptive traffic control for traffic networks. In: *International Conference on Industrial Networks and Intelligent Systems INISCOM*. Da Nang, Vietnam: Springer International Publishing, 2018. p. 3–13. 41

VANDAELE, N.; WOENSEL, T. V.; VERBRUGGEN, A. A queueing based traffic flow model. *Transportation Research Part D: Transport and Environment*, Elsevier BV, v. 5, n. 2, p. 121–135, mar 2000. 16

WATKINS, C. J. C. H.; DAYAN, P. Q-learning. *Machine Learning*, Springer Science and Business Media LLC, v. 8, n. 3-4, p. 279–292, may 1992. 27

WEBSTER, F. V.; COBBE, B. M. *Traffic Signals*. [S.l.]: Her Majesty's Stationary Office, 1966. 38

YAU, K.-L. A. et al. A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Computing Surveys*, Association for Computing Machinery (ACM), v. 50, n. 3, p. 1–38, jun 2017. 17, 42, 43, 88, 89

ZENG, J.; HU, J.; ZHANG, Y. Adaptive traffic signal control with deep recurrent q-learning. In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. Changshu. China: IEEE, 2018. p. 1215–1220. 41, 44

ZHAO, D.; DAI, Y.; ZHANG, Z. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, Institute of Electrical and Electronics Engineers (IEEE), v. 42, n. 4, p. 485–494, jul 2012. 17, 43, 88