UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE ESTATÍSTICA E MATEMÁTICA APLICADA

PROGRAMA DE PÓS-GRADUAÇÃO EM MODELAGEM E MÉTODOS QUANTITATIVOS

MESTRADO ACADÊMICO EM MODELAGEM E MÉTODOS QUANTITATIVOS

FRANCISCO EDYVALBERTY ALENQUER CORDEIRO

APRENDIZADO POR REFORÇO PROFUNDO APLICADO AO PROBLEMA DE ALOCAÇÃO DE VEÍCULOS DINÂMICO E ESTOCÁSTICO

FORTALEZA

2023

FRANCISCO EDYVALBERTY ALENQUER CORDEIRO

APRENDIZADO POR REFORÇO PROFUNDO APLICADO AO PROBLEMA DE ALOCAÇÃO DE VEÍCULOS DINÂMICO E ESTOCÁSTICO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Métodos Quantitativos do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Inteligência Computacional e Otimização.

Orientador: Prof. Dr. Anselmo Ramalho Pitombeira Neto.

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C819a Cordeiro, Francisco Edyvalberty Alenquer.
Aprendizado por reforço profundo aplicado ao problema de alocação de veículos dinâmico e estocástico / Francisco Edyvalberty Alenquer Cordeiro. – 2023.
106 f. : il. color.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Modelagem e Métodos Quantitativos, Fortaleza, 2023.
Orientação: Prof. Dr. Anselmo Ramalho Pitombeira Neto.

1. Otimização matemática. 2. Simulação de eventos discretos. 3. Aprendizado por reforço. 4. Redes neurais (Computação). 5. Q-learning. I. Título.

CDD 510

FRANCISCO EDYVALBERTY ALENQUER CORDEIRO

APRENDIZADO POR REFORÇO PROFUNDO APLICADO AO PROBLEMA DE ALOCAÇÃO DE VEÍCULOS DINÂMICO E ESTOCÁSTICO

Dissertação apresentada ao Programa de Pós-Graduação em Modelagem e Métodos Quantitativos do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de mestre em Modelagem e Métodos Quantitativos. Área de Concentração: Inteligência Computacional e Otimização.

Aprovada em: 13/12/2023

BANCA EXAMINADORA

Prof. Dr. Anselmo Ramalho Pitombeira Neto (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Ricardo Coelho Silva
Universidade Federal do Ceará (UFC)

Prof. Dr. César Lincoln Cavalcante Mattos
Universidade Federal do Ceará (UFC)

Dedico este trabalho às mulheres da minha vida, minha mãe e minha namorada, que são o motivo de eu continuar seguindo em frente.

AGRADECIMENTOS

À Sonja, minha mãe, por todo o seu apoio incondicional e por ser meu maior exemplo de perseverança.

À Renata, minha namorada, por estar comigo em todos os momentos e fazer com que os meus piores dias passem despercebidos. Você me traz paz em meio a tantas tempestades.

Ao Prof. Dr. Anselmo Ramalho Pitombeira Neto pelos valiosos ensinamentos, conversas e pela excelente orientação. Todo o seu apoio tornou esta jornada muito mais tranquila.

Ao Prof. Dr. César Lincoln Cavalcante Mattos pela excelência em seu papel como professor e por ser uma inspiração na busca pelo conhecimento.

Ao Prof. Dr. Juvêncio Santos Nobre por ter me mostrado que a caminhada pode ser desafiadora, mas que com esforço e dedicação sempre podemos dar um passo adiante.

Ao Prof. Dr. Ricardo Coelho Silva pelas excelentes contribuições neste trabalho. Admiro muito o seu comprometimento em ajudar os alunos a irem mais longe.

RESUMO

O problema de alocação de veículos dinâmico e estocástico consiste em decidir quais veículos atribuir a solicitações que surgem de maneira aleatória no tempo e no espaço. Este desafio abrange diversas situações práticas, como o transporte de cargas por caminhões, sistemas de atendimento de emergência e serviços de transporte por aplicativo. Neste estudo, o problema foi modelado como um processo de decisão semimarkoviano, permitindo tratar o tempo como uma variável contínua. Nessa abordagem, os momentos de decisão coincidem com eventos discretos, cujas durações são aleatórias. A aplicação dessa estratégia baseada em eventos resulta em uma significativa redução do espaço de decisões, diminuindo a complexidade dos problemas de alocação envolvidos. Além disso, mostra-se mais adequada para situações práticas quando comparada com os modelos de tempo discreto frequentemente utilizados na literatura. Para validar a abordagem proposta, foi desenvolvido um simulador de eventos discretos e realizado o treinamento de dois agentes tomadores de decisão utilizando o algoritmo de aprendizado por reforço chamado Double Deep Q-Learning. Os experimentos numéricos foram conduzidos em cenários realistas de Nova York, e os resultados da abordagem proposta foram comparados com heurísticas comumente empregadas na prática, evidenciando melhorias substanciais, incluindo a redução de até 50% nos tempos médios de espera em comparação com as demais políticas testadas.

Palavras-chave: otimização matemática; simulação de eventos discretos; aprendizado por reforço; redes neurais (computação); Q-learning.

ABSTRACT

The dynamic and stochastic vehicle allocation problem involves deciding which vehicles to assign to requests that arise randomly in time and space. This challenge includes various practical scenarios, such as the transportation of goods by trucks, emergency response systems, and app-based transportation services. In this study, the problem was modeled as a semi-Markov decision process, allowing the treatment of time as a continuous variable. In this approach, decision moments coincide with discrete events with random durations. The use of this event-based strategy results in a significant reduction in decision space, thereby reducing the complexity of the allocation problems involved. Furthermore, it proves to be more suitable for practical situations when compared to discrete-time models often used in the literature. To validate the proposed approach, a discrete event simulator was developed, and two decision-making agents were trained using the reinforcement learning algorithm called Double Deep Q-Learning. Numerical experiments were conducted in realistic scenarios in New York, and the results of the proposed approach were compared with commonly employed heuristics, demonstrating substantial improvements, including up to a 50% reduction in average waiting times compared to other tested policies.

Keywords: mathematical optimization; discrete event simulation; reinforcement learning; neural networks (computer science); Q-learning.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

$\mathcal{S}$	Espaço de estados possíveis
$\mathcal{A}$	Conjunto de ações viáveis
$\mathcal{A}(s)$	Conjunto de ações viáveis no estado s
$\mathcal{R}$	Conjunto das recompensas
t	Instante de tempo
S_t	Estado no instante t
A_t	Ação realizada no instante t
R_t	Recompensa recebida no instante t
s	Estado atual
s'	Estado posterior
a	Ação atual
a'	Ação posterior
r	Recompensa atual
$p(s' \mid s, a)$	Probabilidade de ir para o estado s' dado que a decisão a foi tomada no estado s
G_t	Soma acumulada das recompensas obtidas a partir do tempo t
π	Representa uma política, que é uma função $\mathcal{S} \to \mathcal{A}$
$\mathbb{E}$	Valor esperado
v	Função de valor de estado
v^*	Função de valor de estado ótima
$v_\pi(s)$	Função de valor seguindo uma política π estando no estado s
q	Função de valor de ação
q^*	Função de valor de ação ótima
$q_\pi(s, a)$	Função de valor de ação seguindo uma política π e tomando a ação a no estado atual s
γ	Fator de desconto
$\mathcal{L}$	Função de perda

SUMÁRIO

1 INTRODUÇÃO

De acordo com Cheng e Qu (2009), serviços de mobilidade urbana possuem uma grande importância em áreas metropolitanas. Nos dias atuais tais serviços podem ser solicitados de maneira tradicional, em que há a necessidade de contactar atendentes de centrais telefônicas para os informar acerca dos detalhes das viagens; de maneira direta, em que se entra em contato diretamente com o próprio motorista; ou por meio de aplicativos para dispositivos móveis. Gao *et al.* (2016) afirmam que os serviços que operam de maneira tradicional em geral confiam demasiadamente na experiência de seus motoristas, o que resulta em baixa eficiência no serviço e longos períodos de espera para os clientes, pois em geral os motoristas não possuem informações precisas do contexto como um todo.

Com a rápida evolução das tecnologias móveis, tem sido percebida uma adesão em massa pela utilização de *smartphones* no dia a dia de grande parte da população urbana de todo o mundo. Segundo Qin *et al.* (2020), essa população, que em 2019 representou 82% da população dos Estados Unidos e 59% da população da China, é a mais propensa a buscar alternativas de transporte ao invés da compra de um veículo próprio.

Com isso, está sendo possível ocorrer uma grande transformação na mobilidade urbana mundial, trazendo consigo um alto potencial de impactos positivos relacionados à poluição, ao consumo de energia e ao controle de tráfego, conforme afirmam Alonso-Mora *et al.* (2017). Gao *et al.* (2016) citam que além das oportunidades relativas à sustentabilidade, percebeu-se também uma grande oportunidade de obtenção de lucros com a prestação de serviços de mobilidade urbana por meio de aplicativos de dispositivos móveis, tais como Uber, DiDi e Lyft, os quais se tornaram rapidamente predominantes graças aos recentes avanços dessas tecnologias, facilitando o acesso aos serviços de táxi.

Para que esses serviços sejam eficientes, atendendo à demanda em intervalos de tempo reduzidos, é necessário que o sistema consiga conduzir o estado de toda a plataforma para situações em que, na medida do possível, haja pelo menos um veículo para atender cada novo chamado recebido em um tempo tolerável pelo usuário. Qin *et al.* (2020) citam três estratégias utilizadas para a otimização dessas operações: o despacho de veículos (normalmente utilizando a estratégia de *matching*), o reposicionamento de veículos e a precificação de corridas.

Esses mecanismos visam resolver problemas distintos, mas com objetivos semelhantes. No despacho de veículos o problema está em definir o melhor veículo para cada chamada em espera. Nesse caso normalmente busca-se minimizar o tempo de espera do cliente ou maximizar

o lucro dos motoristas. No reposicionamento de veículos o problema está em fazer com que os veículos disponíveis se posicionem em localidades com maior probabilidade de ocorrência de chamados, reduzindo assim o tempo de espera dos clientes. Já na precificação de corridas o problema se limita a definir preços dinâmicos para as corridas para controlar o volume de chamados em determinadas localidades, balanceando assim a relação oferta e a demanda durante intervalos de tempo em que há o eventual descontrole desses quesitos, o que evita cancelamentos de chamados em massa. Neste trabalho o problema será visto sob a ótica do despacho de veículos.

A satisfação do cliente ao contratar serviços de mobilidade urbana não é um atributo inerente aos serviços prestados por aplicativos móveis. Na realidade, para que esses serviços sejam prestados de maneira eficiente e atinjam um alto nível de satisfação dos clientes, é necessário que a tomada de decisão por parte do prestador do serviço seja planejada da forma mais vantajosa possível para o sistema como um todo, considerando inclusive os impactos futuros dessas decisões, o que torna a busca por soluções ótimas uma tarefa não trivial.

Grande parte dos trabalhos anteriores que buscam resolver o problema pelo ponto de vista do despacho de veículos assumem que as épocas de decisão ocorrem em intervalos de tempo fixos, sendo o período entre esses intervalos utilizado para a coleta de informações por parte do sistema. Nas épocas de decisão o objetivo é realizar a alocação dos veículos aos chamados de forma ótima, o que normalmente envolve resolver um problema de alocação muitos-para-muitos. Essa abordagem é denominada de *matching* e possui algumas limitações:

1. Em toda época de decisão, o espaço de decisões viáveis corresponde a todas as alocações possíveis de veículos a chamados. Quando consideradas as restrições do problema, como por exemplo a tolerância máxima de espera do chamados, esse problema pode ser visto como um problema de alocação generalizado, que é um problema de otimização combinatória da classe NP-Difícil. Os trabalhos que utilizam essa abordagem utilizam uma relaxação do problema, desconsiderando a tolerância dos chamados, e o resolvem como um problema de alocação linear, o que simplifica bastante o problema real.
2. Como essa abordagem trata todas as alocações simultaneamente e possíveis rejeições de chamados só ocorrem após a tomada de decisão, é assumido que todas as alocações propostas pelo sistema serão aceitas.
3. Como as épocas de decisão ocorrem em períodos fixos, há a necessidade de definição

do tamanho do intervalo. Quando o intervalo é muito curto há um aumento no custo computacional, pois haverá necessidade de resolver mais problemas de alocação, já quando os intervalos são longos o tempo de espera dos chamados tende a aumentar, uma vez que os sistema ficará ocioso entre esses períodos.

O objetivo geral deste trabalho é propor uma modelagem para o problema de alocação de veículos dinâmico e estocástico, fazendo uso de aprendizado por reforço profundo para a definição da política de tomada de decisão do sistema e buscando minimizar o tempo de espera dos clientes e superar as limitações das estratégias baseadas em *matching*.

Visando alcançar o objetivo geral, os seguintes objetivos específicos foram definidos:

1. Formular o problema como um processo de decisão semimarkoviano;
2. Desenvolver um ambiente adequado para a realização dos experimentos computacionais;
3. Implementar os agentes envolvidos na solução do problema;
4. Realizar os experimentos utilizando dados reais;
5. Comparar os resultados obtidos com os resultados de heurísticas clássicas.

As seguintes contribuições serão realizadas neste trabalho.

1. Proposta de uma modelagem para o problema de alocação de veículos dinâmico e estocástico utilizando aprendizado por reforço profundo considerando uma abordagem baseada em eventos, a qual resulta em uma redução considerável da complexidade do espaço de decisões.
2. Desenvolvimento de um ambiente de simulação que utiliza dados reais para amostrar a demanda de chamados da plataforma e simular a dinâmica do ambiente.
3. Implementação de dois agentes decisores que utilizam aprendizado por reforço profundo para a definição de suas políticas.
4. Introdução de atributos estocásticos que são frequentemente ignorados em outros trabalhos, como a probabilidade de rejeição das alocações por parte do veículo e a tolerância máxima de espera dos chamados.
5. Comparação da abordagem proposta com políticas frequentemente utilizadas na prática.

No Capítulo 2 serão abordados todos os conceitos envolvidos na modelagem realizada neste trabalho. No Capítulo 3 serão apresentadas algumas abordagens que outros autores utilizaram para resolver o problema de alocação de veículos dinâmico e estocástico. No Capítulo 4 será apresentada a forma como o problema foi formulado, bem como serão apresentados detalhes de implementação acerca do ambiente computacional. No Capítulo 5 serão apresentadas

as definições feitas para a configuração dos experimentos, bem como os resultados obtidos na execução dos testes. Por fim, no Capitulo 6 serão apresentadas as conclusões acerca do trabalho e possíveis trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo será apresentado todo o apanhado teórico envolvido na proposta apresentada neste trabalho.

2.1 Problema de alocação clássico

Um problema de alocação em sua forma mais básica pode ser exemplificado como um problema que consiste em atribuir tarefas a pessoas de forma a minimizar o tempo total de realização de todas as tarefas. Essa ideia pode ser estendida para situações em que se deseja atribuir tarefas a máquinas, trabalhadores a máquinas, funcionários a cargos, professores a disciplinas, táxis a passageiros e muitas outras possibilidades.

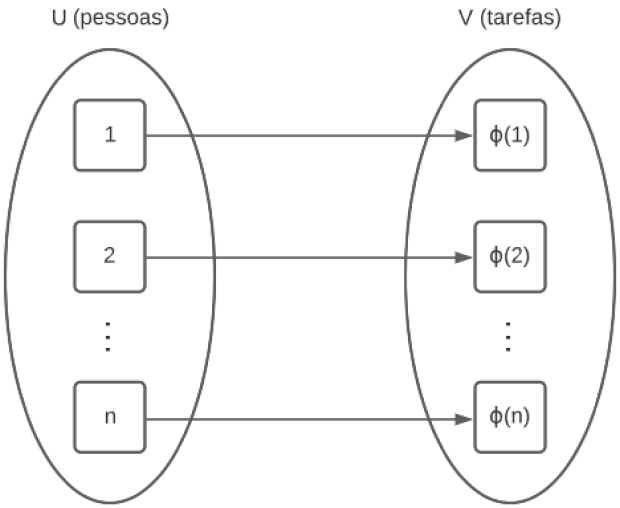
É natural perceber que problemas dessa natureza surgem em diversos campos do conhecimento, uma vez que é comum a necessidade de otimização de sistemas com vistas a superar o problema da escassez de recursos, o qual está abundantemente presente em problemas da vida real.

Existem diversas formas de descrever este problema matematicamente. De acordo com Burkard *et al.* (2009), uma dessas formas é enxergar o problema como um mapeamento bijetivo ϕ entre dois conjuntos finitos U e V com n elementos cada um. Dado que os conjuntos U e V são conhecidos, pode-se representar uma alocação por uma permutação dos elementos, ou seja, considerando o exemplo citado anteriormente, cada elemento do conjunto das pessoas atende exatamente uma tarefa e toda tarefa é atendida por exatamente uma pessoa. Sendo assim, cada elemento $u \in U$ possui exatamente um correspondente $\phi(u) \in V$, conforme Figura 1.

Pentico (2007) cita que o termo "Problema de Alocação", do inglês "*Assignment Problem*", aparentemente surgiu no ano de 1952, quando Votaw e Orden (1952) publicaram o trabalho entitulado "*The personnel assignment problem*". Contudo, geralmente se reconhece o ano de 1955 como o ano em que se iniciou o desenvolvimento de métodos de solução práticos para esta classe de problemas, quando Kuhn (1955) publicou um artigo apresentando o método húngaro para solucionar o problema de alocação determinístico e estático. De acordo com Burkard *et al.* (2009), este foi o primeiro método a resolver um problema de alocação em tempo polinomial, o que demonstra sua grande contribuição para a área quando foi proposto.

Uma possível modelagem matemática de um problema de alocação clássico em que se deseja atribuir n tarefas a n recursos de forma a minimizar o custo total está apresentado a

Figura 1 – Mapeamento bijetivo de pessoas e tarefas



Fonte: Elaborada pelo autor.

seguir:

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{2.1}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, \ldots, n \tag{2.2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, \ldots, n \tag{2.3}$$

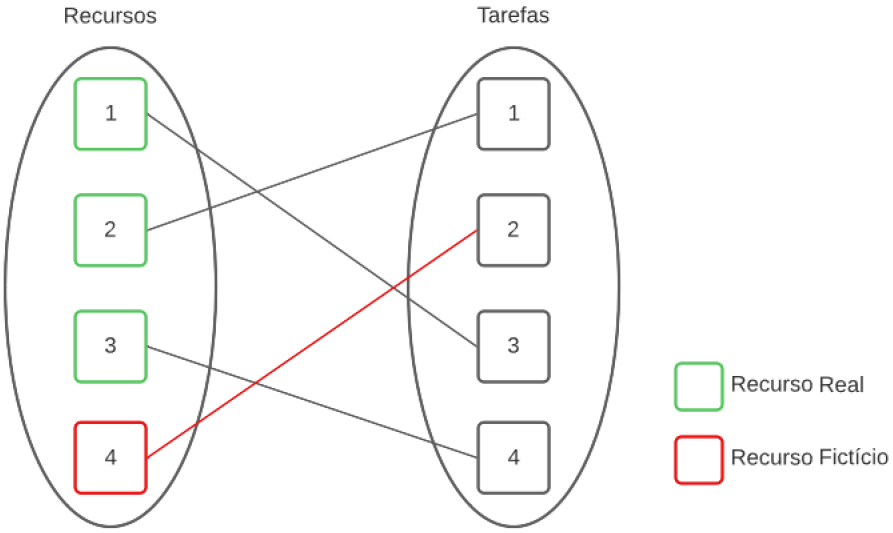
em que x_{ij} são as variáveis do modelo e assumem os valores:

$$x_{ij} = \begin{cases} 1, & \text{se a tarefa } j \text{ é atribuida ao recurso } i \\ 0, & \text{caso contrário} \end{cases}$$

e os valores c_{ij} representam o custo de alocar uma tarefa j para um recurso i. As restrições 2.2 garantem que só haverá um recurso i alocado a cada tarefa j e as restrições 2.3 garantem que só haverá uma tarefa j atribuída a cada recurso i.

Quando um problema de alocação é desbalanceado, em geral contendo mais tarefas que recursos, uma prática comum é adicionar recursos fictícios, que simbolizam o não atendimento das tarefas que forem atribuídas a eles. Nesses casos, totalizam-se n recursos reais, m recursos fictícios e $n + m$ tarefas, conforme Figura 2. Deste modo, haverá balanceamento entre tarefas e recursos e a solução do problema permanecerá similar à solução apresentada anteriormente.

Figura 2 – Recursos fictícios para balanceamento do problema de alocação



Fonte: Elaborada pelo autor.

Apesar de ser uma forma de possibilitar a aplicação de técnicas como o método húngaro para solucionar um problema de alocação desbalanceado, essa pode não ser uma solução passível de aplicação em problemas reais, pois algumas tarefas não seriam atribuídas a nenhum recurso real, permanecendo assim em estado pendente. Para solucionar esse problema, alguns autores propuseram modificações no método húngaro para que todas as tarefas excedentes sejam atendidas. Rabbani *et al.* (2019), por exemplo, formulam o problema alterando as restrições 2.3 da seguinte forma:

$$\sum_{j=1}^{n} x_{ij} \geq 1, \qquad i = 1, \ldots, n. \tag{2.4}$$

Em termos práticos, a consequência dessa modificação é que os recursos podem ser alocados a mais de uma tarefa.

O problema de alocação generalizada (PAG), que é frequentemente associado com o problema da mochila, tem como objetivo identificar uma alocação ótima de n itens em m mochilas, considerando que cada mochila possui uma capacidade máxima. O problema de alocação apresentado anteriormente trata-se de um caso especial do problema de alocação generalizada quando os pesos dos itens e as capacidades das mochilas são iguais a 1. Vale ressaltar que o exemplo foi apresentado com base no problema da mochila, mas existem diversas aplicações práticas do PAG, conforme apresentam Cattrysse e Wassenhove (1992). Assim sendo, os termos itens e mochilas são intercambiáveis com outros termos, a depender do contexto em que o PAG está sendo aplicado.

Fisher *et al.* (1986) afirmam que o PAG é um problema NP-difícil, uma vez que o problema de partições é NP-completo e é redutível ao PAG. Assim sendo, como a solução de grandes instâncias se torna intratável, já foram propostas diversas abordagens baseadas em heurísticas e metaheurísticas para a obtenção de soluções aproximadas em tempo viável. A formulação matemática do PAG a seguir descreve a mesma formulação apresentada por Öncan (2007), portanto, apenas para fins de exemplificação, utilizaremos a terminologia adequada para o problema da mochila. Contudo, conforme citado anteriormente, o problema pode ser estendido para outros contextos.

$$\min \quad \sum_{i=1}^{n}\sum_{j=1}^{m} c_{ij}x_{ij} \tag{2.5}$$

$$\text{s.a.} \quad \sum_{j=1}^{n} r_{ij}x_{ij} \leq b_i, \qquad i = 1, \ldots, m, \tag{2.6}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad j = 1, \ldots, n, \tag{2.7}$$

em que x_{ij} são as variáveis do modelo e assumem os valores:

$$x_{ij} = \begin{cases} 1, & \text{se o item } j \text{ é atribuído à mochila } i \\ 0, & \text{caso contrário} \end{cases}$$

e os valores c_{ij} representam o custo de alocar um item j à mochila i, os valores b_i representam a capacidade da mochila i e os valores r_{ij} representam o peso do item j se este for atribuído à mochila i.

As restrições 2.6 garantem que as capacidades das mochilas não serão excedidas e as restrições 2.7 garantem que cada item será atribuído a apenas uma mochila.

2.2 Problema de roteamento de veículos

O problema de roteamento de veículos (PRV) pode ser entendido como um caso particular do problema de alocação generalizado, em que os recursos são os veículos e as tarefas são as localizações que esses veículos precisam alcançar. Além disso, cada veículo (recurso) tem um custo associado relativo a cada localização (tarefa), sendo que tais custos não são necessariamente iguais.

Este problema foi introduzido por volta de 1959, quando Dantzig e Ramser (1959) apresentaram uma formulação matemática e uma abordagem para a solução de um problema real

que visa determinar rotas ótimas para uma frota de caminhões de entrega de gasolina fazendo a coleta nas distribuidoras e entregando em um número elevado de postos de gasolina. Dantzig e Ramser (1959) citam que esse problema pode ser considerado uma generalização do conhecido problema do caixeiro viajante.

Desde então, diversas heurísticas foram propostas para o problema e, dadas as particularidades de diversos problemas reais, surgiram com o passar dos anos algumas variantes. Toth e Vigo (2002a) apresentam algumas das principais variantes do problema: o PRV capacitado, o PRV com restrição de distância. o PRV com janelas de tempo e o PRV com coleta e entrega.

No PRV capacitado todas as localizações que devem ser visitadas são referentes a entregas e esta demanda é determinística, conhecida a priori. Outra importante característica, segundo Xiao *et al.* (2012), é a restrição de que a carga transportada em cada rota não exceda a capacidade de carregamento do veículo designado. No PRV com restrição de distância, como a própria nomenclatura sugere, há a inclusão de restrição de distância máxima a ser percorrida por cada veículo.

Em relação ao PRV com janela de tempo, tem-se uma extensão do PRV capacitado, em que, segundo Cordeau *et al.* (2002), o serviço atribuído a cada cliente i deve iniciar em um tempo a_i e o veículo deve permanecer no local até o tempo b_i, sendo $b_i > a_i$. Para problemas dessa natureza podem ser definidos dois tipos de janela de tempo: *soft* e *hard*. No caso da janela de tempo *soft* permite-se a violação da janela de tempo, contudo existe um custo associado a cada violação. Já no caso da janela de tempo *hard* não é permitido que se chegue ao cliente após o horário definido para início do serviço.

De acordo com Goetschalckx e Jacobs-Blecha (1989), o PRV com *backhauls* envolve pontos de coleta e entrega. Nessa configuração do problema, os pontos de entrega são aqueles que receberão uma quantidade de mercadorias do centro de distribuição e os pontos de coleta são aqueles em que os veículos farão a coleta, ou seja, os próprios centros de distribuição. Toth e Vigo (2002b) exemplificam uma situação prática desse problema apresentando o caso dos supermercados, em que os veículos precisam fazer a coleta de mercadorias em centros de distribuição para realizarem a entrega dessas mesmas mercadorias nos supermercados. Goetschalckx e Jacobs-Blecha (1989) citam ainda que um pressuposto crucial para o PRV com *backhauls* é que todas as entregas de cada rota devem ser realizadas antes de se realizar qualquer coleta de mercadorias, isso porque não é viável o rearranjo das mercadorias entre pontos de entrega.

Conforme Desauniers *et al.* (2002), o PRV com coleta e entrega é um problema em

que se tem uma frota heterogênea de veículos baseadas em múltiplos terminais. Essa frota deve satisfazer um conjunto de solicitações de transporte e cada solicitação é definida por um ponto de coleta, um ponto de entrega correspondente e a demanda a ser transportada entre os dois pontos. É possível notar que o PRV capacitado pode ser considerado um caso particular do PRV com coleta e entrega, em que os destinatários de todas as solicitações são um central de distribuição comum.

O problema de roteamento de veículos faz parte da classe de problemas classificados como NP-difícil, como evidenciado por Kumar e Panneerselvam (2012). Dessa forma, encontrar soluções ótimas e precisas em tempo hábil para aplicações práticas pode se tornar inviável, a depender do tamanho da instância do problema. Diante dessa limitação, diversas técnicas aproximadas têm sido desenvolvidas ao longo dos anos para solucionar esse problema. É importante destacar que os problemas apresentados até agora são considerados estáticos e determinísticos, o que não reflete a realidade de alguns problemas de transporte. Quando se trata de versões desses problemas que envolvem características dinâmicas e estocásticas, como é o caso do problema de alocação de veículos dinâmico e estocástico que será abordado adiante, a dificuldade em resolvê-los naturalmente aumenta. É interessante notar que o problema de alocação de veículos dinâmico e estocástico está intimamente relacionado ao problema de roteamento de veículos, mas envolve um nível maior de complexidade, uma vez que a alocação de veículos é feita de forma sequencial e leva em consideração a incerteza temporal e espacial.

2.3 Problema de alocação de veículos dinâmico e estocástico

De acordo com Mendonça *et al.* (2021), o problema de alocação de veículos pode ser definido da seguinte forma: dada uma frota de veículos e um conjunto de localizações, deseja-se alocar os veículos disponíveis para visitar as localizações de um conjunto de chamados ao longo do tempo de modo a minimizar o custo médio associado às decisões de alocação.

Atualmente, algumas organizações amplamente reconhecidas no mundo enfrentam o problema de alocação de veículos dinâmico e estocástico, como aplicativos de entregas de refeições, a exemplo do iFood e Rappi, que requerem que seus entregadores coletem e entreguem alimentos; sistemas de atendimento médico de emergência (SAMU), que precisam alocar ambulâncias para ocorrências de emergência; e aplicativos de mobilidade urbana, como Uber, Lyft e DiDi, que realizam a alocação de veículos a partir de chamadas recebidas através de aplicativos de viagens.

Quando a demanda a ser transportada no problema de roteamento de veículos com coleta e entrega são pessoas, como no caso dos aplicativos de mobilidade urbana apresentados anteriormente, temos a caracterização de um problema *dial-a-ride*. Este problema está intimamente relacionado ao problema de roteamento de veículos com coleta e entrega e com janela de tempo, uma vez que as pessoas normalmente possuem restrições de tempo, conforme Molenbruch *et al.* (2017). De acordo com Cordeau e Laporte (2007), o que faz o problema *dial-a-ride* diferente da maioria dos problemas de roteamento de veículos é a perspectiva humana, pois nesse caso há a necessidade de ponderar a experiência do usuário frente à minimização dos custos operacionais.

O problema de alocação de veículos dinâmico e estocástico (PAVDE) abordado neste trabalho possui grande similaridade com o problema *dial-a-ride*; contudo, difere em alguns aspectos. Ao contrário do problema *dial-a-ride*, no PAVDE não há a existência de garagens ou depósitos centrais. Além disso, como trata-se de um problema que considera a dinâmica e a estocasticidade do ambiente em que está inserido, a priori não se possui informações acerca das origens e dos destinos das viagens, e nem mesmo do momento em que as solicitações surgem, o que adiciona considerável complexidade ao problema.

2.4 Problema de decisão sequencial

A quase todo instante os indivíduos estão tomando decisões com vistas a atingir algum objetivo específico, mesmo que algumas vezes de forma involuntária. Uma cadeia de decisões pode levá-lo a diversas situações, ou estados, a depender do contexto em que está inserido. Além disso a ordem em que as decisões são encadeadas influencia na busca pelo alcance do objetivo e existe um inter-relacionamento entre elas, isto é, uma mesma decisão tomada em momentos diferentes pode gerar diferentes resultados, positivos ou negativos, e levar o tomador de decisão a situações distintas, mesmo considerando um contexto determinístico.

Hadley (1964) e Hausman (1969) definiram um problema de decisão sequencial como um cenário no qual a resolução envolve uma série de decisões, com duas ou mais tomadas em momentos diferentes ao longo do tempo. Nesse contexto, cada decisão é suscetível a influências tanto de escolhas anteriores como de variáveis estocásticas.

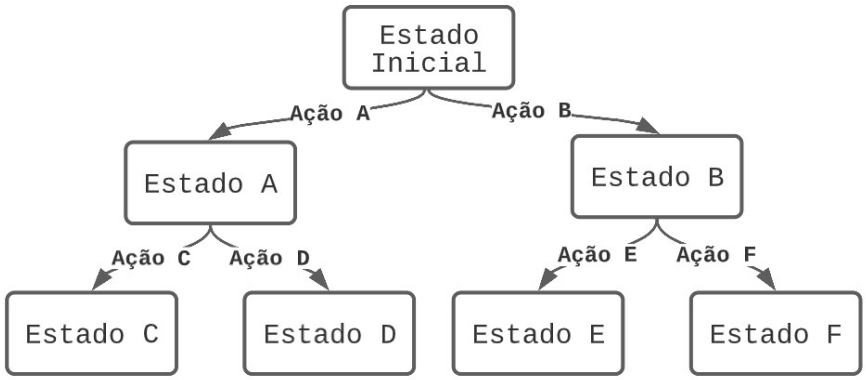
Hausman (1969) afirma que a diferença crucial entre problemas de decisão sequenciais e não sequenciais é que em problemas sequenciais as decisões futuras se baseiam parcialmente em informações desconhecidas no presente, mas que serão observadas em algum momento futuro, ou seja, à medida que as decisões são tomadas novas informações tendem a surgir no ambiente.

Não é difícil perceber que as consequências das decisões que são tomadas durante um dado espaço de tempo podem ser tanto imediatas, quando ocorrem imediatamente após a realização da ação que foi decidida, quanto podem ocorrer em momentos futuros, caracterizando o que chamaremos adiante de consequência atrasada ou recompensa atrasada. Além disso, também é possível perceber que, quando consideramos um determinado contexto, as ações não devem ser consideradas isoladamente, uma vez que uma decisão tomada em um certo tempo t impacta em um tempo posterior $t + 1$, a decisão tomada no tempo $t + 1$ impacta no tempo posterior $t + 2$ e assim por diante. Situações dessa natureza, em que as decisões são tomadas sequencialmente de tal forma que as decisões tomadas no presente impactam de alguma forma as decisões que serão tomadas no futuro, são características de problemas de decisão sequencial.

De acordo com Puterman (2005), as decisões são usualmente tomadas levando em consideração o atual estado do tomador de decisão (também chamado de agente ou controlador), ou seja, cada estado possui um determinado conjunto de decisões viáveis. Sendo assim, as ações tomadas no tempo presente impactam não só o estado futuro do agente decisor, mas também o espaço de ações futuramente disponível para a tomada de decisão.

Uma das formas mais simples de se representar um problema de decisão sequencial é por meio de árvores de decisão, conforme Figura 3.

Figura 3 – Árvore de decisões



Fonte: Elaborada pelo autor.

Na Figura 3 é possível perceber algumas das características dos problemas de decisão sequencial apresentados anteriormente. Em primeiro lugar nota-se que é possível atingir qualquer um dos nós da árvore a partir das ações disponíveis para a tomada de decisão. Em segundo lugar é perceptível que para alcançar os estados pertencentes às folhas da árvore (nós que não possuem filhos) é necessário percorrer uma determinada cadeia de decisões, não sendo possível

realizar saltos entre os estados que não são adjacentes. Por exemplo, para atingir o estado E é necessário que se tenha tomado inicialmente a ação B. Assim, caso o agente inicie o processo decidindo pela ação A, não há possibilidade, nessa configuração, de atingir nem o estado E e nem o estado F, pois o fluxo ocorre apenas em uma direção. Outra característica a se observar é que, assim como o espaço de estados possíveis muda com a realização das ações, o espaço das ações possíveis também pode mudar. No caso apresentado, inicialmente o agente possui duas ações possíveis: A e B. Quando se toma a ação A o agente evolui para o estado A e as ações possíveis passam a ser C e D, já quando se toma a ação B o agente evolui para o estado B e as ações possíveis passam a ser E e F. Vale ressaltar que as árvores de decisão utilizadas para esse tipo de problema não precisam ser necessariamente binárias como no exemplo apresentado, tendo sido essa configuração escolhida apenas para fins explicativos.

Powell (2022) afirma que quase qualquer problema dinâmico com estados e ações discretas podem ser modelados por meio de árvores de decisão. Contudo, o problema dessa abordagem é que as árvores tendem a crescer de forma explosiva, mesmo em problemas relativamente pequenos, tornando assim a solução do problema inviável computacionalmente. Alagoz *et al.* (2010) mencionam ainda que as árvores de decisão possuem sérias limitações em modelar situações complexas, sobretudo quando os resultados ou eventos ocorrem ao longo do tempo.

Como muito frequentemente o objetivo de um problema de tomada de decisão sequencial é a otimização da recompensa acumulada no decorrer de todo o processo, uma outra forma de tratar o problema é como um problema de otimização, conforme apresentam Lew e Mauch (2007). Nesse caso considera-se um problema de otimização da seguinte forma:

$$\max_{x \in \Omega} f(x) \tag{2.8}$$

em que x corresponde à decisão a ser tomada, Ω corresponde ao conjunto que contém todas as decisões viáveis em um dado momento e f corresponde à função objetivo.

Como neste exemplo o problema de otimização está sendo considerado como uma maximização, pode-se presumir que f se trata de uma função de recompensa, caso contrário seria uma função de penalização.

O valor ótimo do problema pode ser denotado por $f^* = f(x^*)$, em que x^* é o valor de $d \in \Omega$ que faz com que f assuma seu valor ótimo, ou seja, x^* é a solução ótima do problema. Desta forma define-se x^* da seguinte maneira:

$$x^* = \arg\max_{x \in \Omega} f(x) \tag{2.9}$$

No caso de problemas de decisão sequencial, em que naturalmente se tem mais de uma decisão a ser tomada, pode-se então expandir a formulação apresentada para o caso em que x é um vetor de decisões. Assim o argumento da função f passa a ser um conjunto de decisões $\{x_1, x_2, ..., x_n\}$.

Como todo problema de otimização, uma possibilidade de otimização, e talvez a mais ingênua, é testar todos os valores possíveis para a variável de decisão, quando se tem apenas uma variável, ou todas as combinações possíveis das variáveis de decisão viáveis, quando se tem mais de uma variável. Segundo Mahoor *et al.* (2017), essa abordagem, que é denominada método de força bruta, é muito útil para resolver problemas que possuem instâncias pequenas, além de ser um método de fácil implementação. Além disso, como essa abordagem consiste em enumerar sistematicamente todas as possíveis soluções e avaliar cada uma delas, quando o problema é determinístico há garantia de convergência para a solução ótima.

Devido ao método de força bruta ser ineficiente e se tornar rapidamente impraticável quando o espaço de decisões cresce, Lew e Mauch (2007) propõem então formular o problema considerando as decisões de forma sequencial, diferentemente de como foi apresentado anteriormente. Dessa forma assume-se que a tupla $(x_1, x_2, ..., x_n)$ é uma sequência de n decisões de tal forma que f^* seja obtida por meio de otimizações encadeadas, conforme apresentado a seguir.

$$f^* = \max_{(x_1,x_2,...,x_n)\in\Omega} \{f(x_1, x_2, ..., x_n)\} \tag{2.10}$$

$$= \max_{x_1\in\omega_1} \{\max_{x_2\in\omega_2} \{\dots \{\max_{x_n\in\omega_n} \{f(x_1, x_2, ..., x_n)\}\} \dots \}\} \tag{2.11}$$

em que $(x_1, x_2, ..., x_n)$ pertence ao espaço de decisões $\Omega = \omega_1 \times \omega_2 \times \cdots \times \omega_n$, com $x_i \in \omega_i$.

Como citado anteriormente, é bastante comum que em problemas de decisão sequencial as decisões sejam inter-relacionadas, isto é, o espaço de decisões ω_i é definido com base na sequência de decisões tomadas até o tempo i, ou seja, as decisões $(x_1, x_2, ..., x_{i-1})$.

Assumindo que $\omega_i(x_1, x_2, ..., x_{i-1})$ é o espaço de decisões no tempo i, sendo que a sequência de decisões $(x_1, x_2, ..., x_{i-1})$ foi tomada anteriormente e que $x_i \in \omega_i(x_1, x_2, ..., x_{i-1})$, pode-se reescrever a fórmula 2.11 explicitando esse inter-relacionamento entre as decisões da seguinte forma:

$$f^* = \max_{x_1\in\omega_1} \{\max_{x_2\in\omega_2(x_1)} \{\dots \{\max_{x_n\in\omega_n(x_1,...,x_{n-1})} \{f(x_1, x_2, ..., x_n)\}\} \dots \}\} \tag{2.12}$$

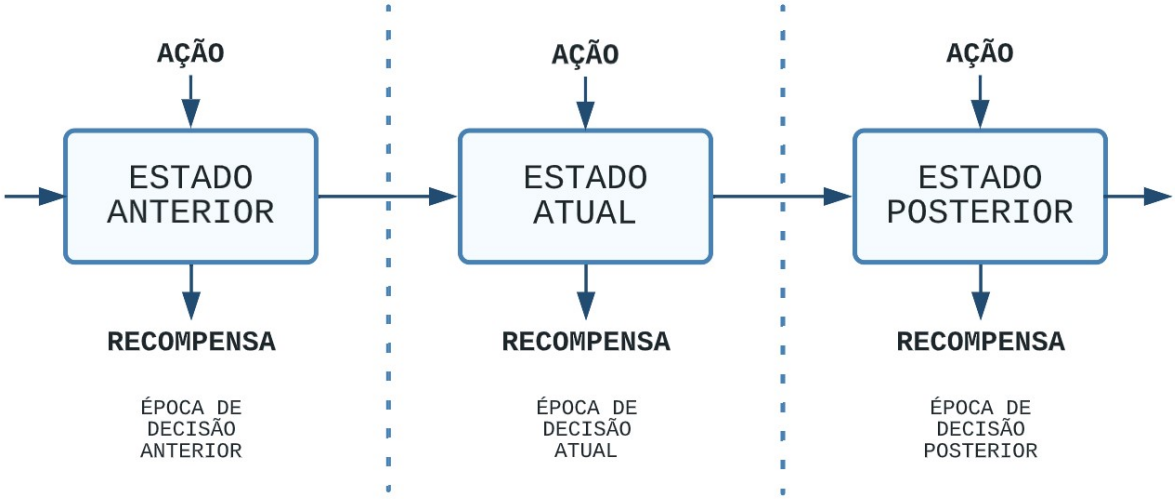
Lew e Mauch (2007) levam a formulação ainda mais adiante aprimorando-a de tal forma a chegar na dedução da equação funcional da programação dinâmica (*dynamic program-*

ming functional equation), contudo, como não é o objetivo principal deste trabalho, a equação 2.12 será considerada suficiente para fins explicativos.

As formulações apresentadas até o momento podem ser utilizadas para tentar resolver problemas estocásticos, mas de certa forma pressupõem que os problemas são determinísticos, uma vez que representar a dinâmica do problema por meio dessas formulações é uma tarefa complexa e frequentemente inviável computacionalmente. Assim, percebeu-se a necessidade de abordar outras representações do problema que possuam uma aplicação mais genérica e que seja adequada para representar problemas de decisão sequencial levando em consideração as nuances que integram essa classe de problemas.

Puterman (2005) apresenta uma representação simbólica de um modelo de decisão sequencial na forma da Figura 4.

Figura 4 – Representação simbólica de um modelo de decisão sequencial



Fonte: Puterman (2005), adaptado pelo autor.

Puterman (2005) afirma que existem alguns itens essenciais em um modelo de decisão sequencial. São eles:

1. Um conjunto de épocas de decisão;
2. Um conjunto de estados do sistema;
3. Um conjunto de ações disponíveis;
4. Um conjunto de recompensas ou custos imediatos dependentes de pares estado-ação;
5. Um conjunto de probabilidades de transições dependentes de pares estado-ação.

As épocas de decisão dizem respeito aos momentos em que o agente tomador de decisão receberá toda a informação provida pelo ambiente em que está inserido e deverá escolher

uma ação no conjunto de ações disponíveis dado aquele estado. Na representação simbólica apresentada na Figura 4 pode-se perceber um fluxo da esquerda para a direita em que cada época de decisão possui seu respectivo estado associado. Além disso, é possível notar também que em cada época de decisão se tem um fluxo vertical, de cima para baixo, simbolizando a tomada de decisão em um determinado estado e gerando como resultado uma recompensa, que na prática também pode ser uma penalidade dependendo do objetivo.

Um importante elemento dessa cadeia de estados é a função de probabilidade atribuída às transições de estado. Pela Figura 4 pode não ficar claro, mas quando o problema possui propriedades estocásticas, naturalmente o estado subsequente não ocorre de maneira determinística como parece estar representado. Ocorre que quando tomada uma decisão em um dado estado, normalmente se possui uma incerteza vinculada à transição para o próximo estado, ou seja, existe uma probabilidade associada a cada par estado-ação que indica a probabilidade de o agente fazer a transição para um dado estado futuro.

Existem algumas formas de se classificar um problema de decisão sequencial. Tais classificações podem ser utilizadas para determinar quais os métodos de solução mais adequados para o problema. Barreto *et al.* () classificam um problema de decisão sequencial por meio da análise de 5 (cinco) características desse problema. São elas: Markoviano ou não Markoviano, determinístico ou estocástico, pequeno ou grande, estacionário ou não estacionário e episódico ou contínuo.

Para identificar se um problema de decisão sequencial satisfaz à propriedade markoviana verifica-se se as transições e recompensas dependem unicamente do estado atual do agente e de sua ação selecionada, ou seja, informações históricas, como as ações já tomadas e os estados que foram assumidos durante as épocas de decisão anteriores, não são relevantes para determinar a recompensa que possa vir a ser recebida e nem as probabilidades associadas às transições do estado atual para o próximo estado. Assim, considerando que um problema de decisão sequencial é Markoviano, sabe-se que todas as informações relevantes para a tomada de decisão estão contidas no próprio estado atual, pois para a dinâmica do problema isso é o suficiente. Quando o problema não satifaz a propriedade markoviana, classifica-se como um problema de decisão sequencial não Markoviano. A propriedade markoviana pode ser expressa matematicamente da seguinte forma:

$$p(s_{t+1} \mid s_t) = p(s_{t+1} \mid s_1, \ldots, s_t), \tag{2.13}$$

em que s_t é o estado no tempo t e s_{t+1} é o estado no tempo subsequente $t + 1$. Quando se

deseja avaliar se um problema de decisão sequencial é determinístico ou estocástico, analisa-se, sobretudo, as probabilidades associadas às transições do sistema. Diz-se que um problema é determinístico quando toda ação a em um dado estado s_t sempre levará o ambiente para o mesmo estado s_{t+1}, não importando quantas vezes essa ação seja repetida nesse estado. Ou seja, em problemas determinísticos quando se escolhe uma ação em um determinado estado, só se possui um estado posterior em que se pode assumir. Em termos probabilísticos pode-se dizer, portanto, que a probabilidade de transição para esse estado é igual a 1 (um) e para todos os demais estados é igual a 0 (zero). Já no caso de problemas de decisão sequencial estocásticos tem-se distribuições de probabilidades associadas às transições do sistema, isto é, dada uma ação a em um estado s_t, não se tem garantia de qual será o próximo estado assumido pela sistema. Desta forma, percebe-se que há uma incerteza associada às transições.

Em relação ao tamanho do espaço de estados do problema, Barreto *et al.* () distinguem os problema de decisão sequencial considerando a capacidade de armazenamento do computador em o problema está sendo modelado. Para tanto, verifica-se se o conjunto de todos os pares ação-estado podem ser armazenados em uma tabela de pesquisa. Caso haja armazenamento suficiente para esse conjunto, diz-se que o espaço de estados é pequeno, caso contrário o espaço de estados é considerado grande. Barreto *et al.* () pontuam ainda o caso dos problemas que possuem espaço de estados contínuo, que o fato de existirem infinitos estados torna impossível o armazenamento total, tornando qualquer problema desta natureza um problema com espaço de estados grande.

Quanto à estacionariedade do problema, diz-se que um problema de decisão sequencial é estacionário quando sua dinâmica é fixa, ou seja, as distribuições de probabilidade associadas às transições de estados do sistema e às recompensas obtidas pelo agente não mudam ao longo do tempo. Já quando o problema é não estacionário, a dinâmica do sistema muda ao longo do tempo, isto é, a dinâmica de duas épocas de decisão distintas não é necessariamente a mesma. Assim, para identificar boas decisões e manter a performance, é necessário que o tomador de decisão esteja ciente que a mesma decisão tomada anteriormente pode não possuir mais o mesmo valor esperado quando tomada em outro momento. Problemas não estacionários são naturalmente mais difíceis de se obter bons resultados.

Para classificar se o problema de decisão sequencial é episódico ou contínuo, verifica-se se há um estado que simboliza o encerramento da tarefa, ou seja, um estado que após sua ocorrência o ambiente deverá ser reiniciado, pois não existem mais estados futuros. Esse estado

é comumente chamado de estado terminal. Caso ele exista, o problema é classificado como episódico, pois quando um estado terminal ocorre, termina-se um episódio e, se for o caso, inicia-se um outro a partir do ponto inicial. Um exemplo bastante comum de tarefa episódica é um jogo de xadrez, em que se tem a situação onde um dos jogadores vence a partida ou é decretado um empate. Essas situações são por definição um estado terminal do ambiente, uma vez que quando ocorrem encerra-se o jogo, evitando que existam estados futuros. No caso de problemas de decisão sequencial que se classificam como contínuos, não existem estados terminais, ou seja, o processo tem a natureza de ocorrer ao longo do tempo sem um critério claro para interrompê-lo. Barreto *et al.* () citam como exemplo uma tarefa de *trade* de ativos financeiros, em que o agente deve decidir se realiza uma compra ou uma venda. Nesse contexto, o agente pode seguir realizando as decisões infinitamente enquanto o mercado de ativos financeiros existir.

2.4.1 Processo de decisão Markoviano

Na seção 2.4 foram apresentadas duas formulações possíveis para o problema de decisão sequencial. Contudo, também foram citadas suas limitações e a necessidade de uma abordagem mais geral que seja adequada para ambientes estocásticos.

Atualmente, uma das abordagens mais utilizadas para problemas dessa natureza é a formalização do problema por meio de processos de decisão Markovianos (PDM). De acordo com Sutton e Barto (2018), os processos de decisão Markovianos representam um modelo clássico de tomada de decisões sequenciais em que as ações tomadas não apenas afetam as recompensas imediatas, mas também influenciam os estados subsequentes e, por conseguinte, as recompensas futuras. Assim, esses processos incluem recompensas que podem ocorrer em momentos posteriores e requerem uma consideração cuidadosa do equilíbrio entre as recompensas imediatas e as futuras.

Sutton e Barto (2018) afirmam ainda que processos de decisão Markovianos são uma forma matematicamente idealizada do problema de aprendizado por reforço onde conclusões teóricas precisas podem ser feitas. Dessa forma, dado que foi utilizado aprendizado por reforço neste trabalho, como será apresentado futuramente, entende-se que formular o problema desta forma pode ser adequado para os objetivos propostos.

Em um processo de decisão Markoviano existem duas figuras importantes que “ditam o ritmo” de como o processo evolui ao longo do tempo: o ambiente e o agente. O agente é a

entidade que está atuando de alguma forma dentro de um contexto e este contexto é o ambiente no qual o agente está introduzido. Durante a interação entre os dois surgem alguns conceitos importantes: o estado, a ação e a recompensa. O estado é uma representação de toda uma situação em um dado momento. Se considerarmos um jogo de xadrez, o estado pode ser considerado como a posição de todas as peças no tabuleiro; se considerarmos um jogo de labirinto, o estado pode ser representado pela posição do agente no labirinto. A ação é a execução de uma tomada de decisão feita pelo agente. Pode-se considerar como uma ação mover uma dada peça de xadrez para uma casa do tabuleiro ou, no caso de um jogo de labirinto, mover-se em alguma direção, como para a esquerda, por exemplo. A recompensa é o estímulo que o ambiente dá para o agente após a realização de alguma ação. Ainda considerando os exemplos apresentados, em um jogo de xadrez pode-se ter uma recompensa positiva, quando o agente vence o jogo, ou negativa, quando o agente perde o jogo. As recompensas também podem ser nulas, quando, dada a modelagem do problema, a ação do agente não deve ser nem estimulada e nem evitada, ou seja, é indiferente.

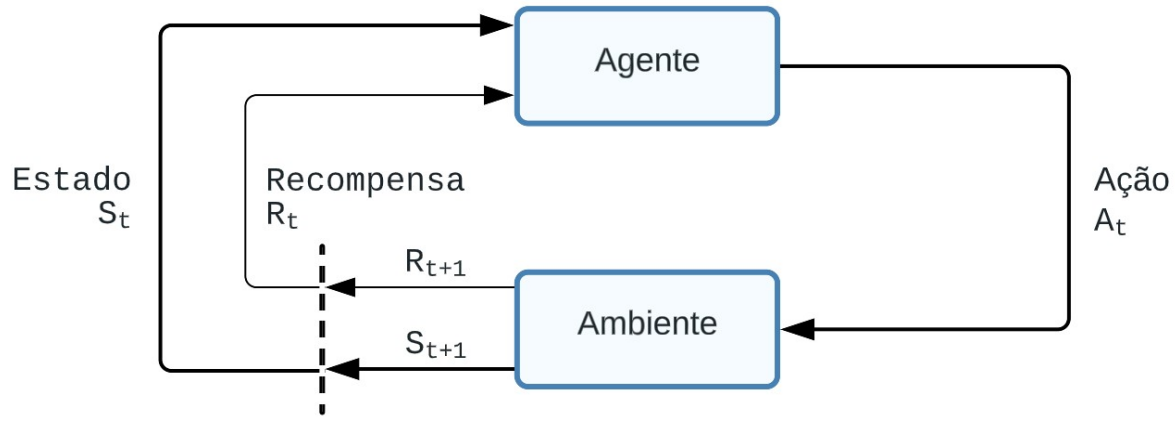
De modo menos informal, pode-se dizer que a interação apresentada anteriormente ocorre em uma sequência de passos em tempo discretos, ou seja, $t = 0, 1, 2, 3, ...$, e cada um dos tempos possui um estado, uma ação e uma recompensa associados, com exceção do $t = 0$, que não possui recompensa associada. Dessa forma, tem-se que em cada tempo t o agente recebe um estado $S_t \in \mathcal{S}$ do ambiente e, baseado em seu julgamento das informações recebidas, seleciona uma ação $A_t \in \mathcal{A}(s)$, em que $\mathcal{A}(s)$ representa o conjunto de ações disponíveis para a tomada de decisão quando o agente está no estado s. No tempo imediatamente posterior $t + 1$, dada a ação selecionada pelo agente, o ambiente emite um sinal de recompensa numérico $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$ e informa o novo estado $S_{t+1} \in \mathcal{S}$ do sistema. Essa dinâmica pode ser facilmente apresentada pela Figura 5.

Em um processo de decisão Markoviano finito, que é o que se apresenta neste tópico, tem-se os conjuntos de ações $\mathcal{A}$, estados $\mathcal{S}$ e recompensas $\mathcal{R}$ com uma quantidade finita de elementos. Para esse caso, Sutton e Barto (2018) afirmam que R_t e S_t possuem distribuições de probabilidade discretas bem definidas que dependem apenas do estado anterior e da ação anterior, ou seja, consegue-se obter a probabilidade de ocorrência de $r \in \mathcal{R}$ e $s' \in \mathcal{S}$ apenas condicionando a ocorrência de $a \in \mathcal{A}$ e $s \in \mathcal{S}$, o que dá origem à função que define a dinâmica de um processo de decisão Markoviano, conforme 2.14 a seguir:

$$p(s', r \mid s, a) \doteq Pr\{S_t = s', R_t = r \mid S_{t-1} = s, A_{t-1} = a\}, \tag{2.14}$$

para todo $s', s \in \mathcal{S}$, $r \in \mathcal{R}$ and $a \in \mathcal{A}(s)$.

Figura 5 – Representação da dinâmica de interação entre o agente e o ambiente em um processo de decisão Markoviano



Fonte: Sutton e Barto (2018), adaptado pelo autor.

A partir da equação 2.14 é possível computar diversas probabilidades envolvidas no ambiente estocástico por meio de operações básicas, como a utilização da propriedade da marginalização ou o uso da esperança matemática. Duas dessas funções de probabilidade obtidas por meio da equação 2.14 são especialmente importantes, pois estão diretamente ligadas à interação apresentada na Figura 5: a função de transição de estado e a função de recompensa. Observa-se na Figura 5 que quando o agente realiza uma ação o ambiente utiliza essa ação, juntamente com o estado anterior que é conhecido, para gerar uma recompensa e um novo estado. A recompensa e o novo estado são gerados por meio das funções de probabilidade obtidas por meio da equação 2.14.

2.4.2 Variáveis de estado

De acordo com Powell (2022), as variáveis de estado devem conter toda a informação que o agente precisa saber para que uma decisão possa ser tomada com base nela, ou seja, são os estímulos percebidos por um agente introduzido em um ambiente que são ponderados para a tomada de decisão. Formalmente, definindo-se $\mathcal{S}$ como o conjunto de todos os estados possíveis do sistema, tem-se que cada elemento $s \in \mathcal{S}$ é um estado especifico e seu valor remete a uma situação momentânea de todo o sistema.

As variáveis de estado podem assumir alguns tipos: físicas, informativas e crenças. As variáveis de estado físicas são aquelas que normalmente se referem à localização ou posição de algo dentro do ambiente. As variáveis informativas são aquelas que se relacionam com os conceito de quantidades e de parâmetros, e são conhecidas, como por exemplo a quantidade de

um dado item em um estoque de uma loja. Já as variáveis de crenças são aquelas informações que não são conhecidas por completo e são descritas na forma de distribuições de probabilidade.

Devido à propriedade markoviana apresentada anteriormente, espera-se que as variáveis de estado utilizadas no contexto de processos de decisão Markovianos possuam toda a informação histórica necessária em si mesmo, isto é, que não seja necessárias informações de estados anteriores para que a tomada de decisão seja tomada de forma efetiva, como por exemplo uma estimativa de alta de um preço de uma ação em um dado dia.

Um importante questão acerca das variáveis de estado é a sua dimensionalidade. Conforme citado no início deste tópico, estas variáveis devem conter toda a informação que o agente precisa para tomar uma decisão, contudo, como boa prática e com o objetivo de evitar inserir complexidade desnecessária ao problema, devem ser utilizadas apenas aquelas informações que influenciem minimamente a tomada de decisão. Assim, espera-se que informações desnecessárias para o agente, mesmo que definam de alguma forma algum atributo do ambiente, não estejam presentes nessas representações, evitando assim tornar o problema mais complexo.

2.4.3 Variáveis de decisão

As variáveis de decisão de referem às possíveis ações tomadas pelo agente. Essas variáveis podem ser binárias, discretas, contínuas ou ainda vetores com variáveis discretas e contínuas. As variáveis de decisão binárias são aquelas que assumem apenas dois valores possíveis, por exemplo decidir entre subir e descer, comprar e vender ou fazer e não fazer algo. As variáveis de decisão discretas são aquelas que podem assumir valores contidos em um conjunto de valores discretos, como por exemplo em um contexto de supermercado decidir qual o próximo produto comprar, ou em um contexto de um jogo de xadrez decidir qual peça movimentar. Já a variáveis de decisão contínuas são aquelas que assumem valores contínuos como por exemplo preços ou taxas.

Durante a evolução de um processo de decisão Markoviano, é importante notar que nem todas as ações estarão disponíveis a todo momento, pois existe uma dependência explicita entre o conjunto de decisões viáveis e o estado em que o sistema se encontra. Dado um estado $s \in \mathcal{S}$, define-se o conjunto de decisões viáveis no estado s como $\mathcal{A}(s)$. Dessa forma o conjunto contendo todas as decisões viáveis considerando todos os estados possíveis pode ser representado por $\mathcal{A} = \bigcup_{s \in \mathcal{S}} \mathcal{A}(s)$.

2.4.4 Épocas de decisão

Conforme já citado anteriormente, as épocas de decisão se referem aos momentos em que o agente recebe do ambiente toda a informação necessária para realizar uma ação, ou seja, o estado.

Um processo de decisão Markoviano pode ser caracterizado pela finitude ou não do conjunto das épocas de decisão. Quando a quantidade de épocas de decisão de um processo é finito, diz-se que este processo possui um horizonte finito; caso contrário, diz-se que possui um horizonte infinito.

Nos problemas com horizontes finitos, como o próprio termo indica, existe uma expectativa de que em algum momento o processo chegará ao fim. Assim, do ponto de vista do agente do sistema, deve-se buscar realizar seu objetivo até que a época de decisão terminal chegue, pois após este momento o processo de encerra. Assim é importante enfatizar que à medida que as épocas de decisão avançam a ação tomada pelo agente pode ser influenciada, isto é, existe uma dependência entre a ação e a época de decisão. Alguns exemplos clássicos de problemas de decisão sequencial com horizonte finito são jogos de xadrez, em que a época de decisão terminal é caracterizada pela vitória de algum dos jogadores ou pelo empate, e jogos de labirinto, em que a época de decisão terminal é caracterizada pela chegada do agente ao destino ou por falhar em encontrar a saída.

Já nos problemas que possuem horizontes infinitos, não se possui a expectativa de que o processo chegue ao fim, dessa forma, é possível perceber que a época de decisão em que o sistema se encontra não tem influência para a tomada de decisão, isto é, são independentes. Uma forma de racionalizar essa característica é percebendo que independente da ação tomada em um dado estado do sistema, sempre haverá infinitos estados posteriores para que o agente busque atingir o objetivo.

Uma outra importante característica dos processos de decisão Markovianos é quanto aos intervalos em que ocorrem as épocas de decisão. De acordo com Puterman (2005), quando consideramos o conjunto das épocas de decisão distribuído em uma reta real não negativa, simbolizando o tempo em que ocorrem, podemos classificá-las como discretas ou contínuas. Quando são classificadas como discretas, o agente deve tomar uma decisão em todas as épocas de decisão que ocorrerem. Já quando são classificadas como contínuas, as decisões do agente podem ocorrer de três formas distintas:

1. As decisões são tomadas em todas as épocas de decisão, o que implica que o agente deverá

tomar decisões continuamente a todo instante;

2. As decisões são tomadas em pontos aleatórios do tempo quando determinados eventos ocorrem, como por exemplo, tomar uma decisão sempre que houver entrada em um sistema de filas;
3. As decisões são tomadas em momento oportunos identificados pelo agente tomador de decisão. Assim, as decisões podem ser tomadas a qualquer momento.

O item 2 acima é o que será utilizado neste trabalho e para tanto será utilizada uma abordagem particular para a modelagem de tempo contínuo chamada de processo de decisão semimarkoviano, a qual será apresentada mais adiante.

2.4.5 Função de transição de estado

Conforme citado anteriormente, a partir da função 2.14, a qual define a dinâmica do processo de decisão Markoviano, é possível deduzir a função que define a probabilidade de transição para um dado estado:

$$p(s' \mid s, a) \doteq Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r \mid s, a). \tag{2.15}$$

A função 2.15 é obtida por meio do uso da propriedade de marginalização, em que foram marginalizadas as recompensas, isto é, foram consideradas todas as recompensas possíveis para que fosse possível excluir a variável da função. Assim é obtida uma função de três argumentos, sendo eles o estado atual $s \in \mathcal{S}$, a ação selecionada $a \in \mathcal{A}(s)$ e o estado posterior $s' \in \mathcal{S}$. Como se trata de uma função de probabilidade, tem-se a seguinte definição da função $p : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$.

Já foi citado anteriormente que o problemas de decisão sequencial podem ser determinísticos ou estocásticos. Esta característica do problema está diretamente ligada com a função de transição de estados, pois a estocasticidade do sistema pode ser momentaneamente explicitada por ela. Quanto o sistema é determinístico tem-se uma redução do contra-domínio da função para um conjunto discreto com os elementos 0 e 1, pois, dados os argumentos da função, tem-se certeza de qual será o seu retorno. Já no caso estocástico existe uma incerteza associada a cada combinação de argumentos.

Outro importante fator que possui impacto na função de transição de estados é se o ambiente possui a propriedade de ser estacionário. Quando o ambiente é estacionário, a função de transição de estados permanece a mesma ao longo do tempo; caso contrário, as distribuições

de probabilidade associadas mudam à medida que as épocas de decisão acontecem, o que insere considerável complexidade ao problema.

2.4.6 Função objetivo

Sabe-se que o objetivo de um problema de decisão Markoviano é maximizar recompensas ou minimizar penalidades, considerando os valores acumulados ao longo do tempo. Porém, esta não é uma tarefa fácil a depender das características do problema.

Quando o problema possui horizonte finito, isto é, quando existe uma época de decisão terminal, pode-se definir a recompensa acumulada da seguinte forma:

$$G_t \doteq R_{t+1} + R_{t+2} + \cdots + R_T, \tag{2.16}$$

em que T é o tempo em que ocorre o estado terminal do sistema e $R_t \in \mathbb{R}$ é a recompensa obtida na época de decisão t.

Já quando o problema possui horizonte infinito, tem-se que $T = \infty$, ou seja

$$G_t \doteq R_{t+1} + R_{t+2} + R_{t+3} + \ldots, \tag{2.17}$$

o que implica que a recompensa acumulada será ∞ ou $-\infty$ com considerável facilidade, a depender do objetivo do problema.

Considerando o caso em que a recompensa acumulada não converge, não faz sentido realizar a otimização das decisões sequenciais, uma vez que qualquer que sejam as decisões a recompensa final tenderá a um limite infinito. Buscando identificar meios de mensurar de forma eficiente a qualidade dos estados e ações, em relação às recompensas, garantindo a convergência da soma das recompensas acumuladas em um problema com horizonte infinito, foi introduzido o conceito de fator de desconto, que é normalmente representado pela letra grega γ. Garcia e Rachelson (2013) explicam que o fator de desconto γ representa o valor presente, ou peso, de uma recompensa recebida no tempo $t + 1$, considerando que o tempo atual é t. Pode-se generalizar o valor presente das recompensas futuras por meio do valor γ^τ, sendo τ a quantidade de tempos posteriores ao atual, ou seja, o tempo $t + \tau$:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2.18}$$

em que γ é chamado de fator de desconto quando $\gamma \in [0, 1)$.

Dessa forma, quando γ é um fator de desconto, ou seja, $0 \leq \gamma < 1$, e considerando que a sequência de recompensas $\{R_k\}$ é limitada, tem-se que a soma infinita em 2.18 converge para um valor finito.

Quando o valor do fator de desconto está próximo de 1, todas as recompensas futuras tendem a ser ponderadas igualmente, ou seja, o agente busca maximizar todas as recompensas ao mesmo tempo, fazendo com que se tenha uma visão de longo prazo. Já quando o fator de desconto é 0, tem-se o caso em que o agente é míope, isto é, o agente considera apenas a recompensa atual, desconsiderando todas as recompensas futuras. Assim o agente determina suas ações sem ponderar as eventuais recompensas recebidas no futuro, focando apenas no curto prazo.

A partir da equação 2.18 é possível perceber que as épocas de decisão subsequentes possuem um forte relacionamento entre si, além de haver claramente um relacionamento recursivo, conforme apresentado a seguir:

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \dots \tag{2.19}$$

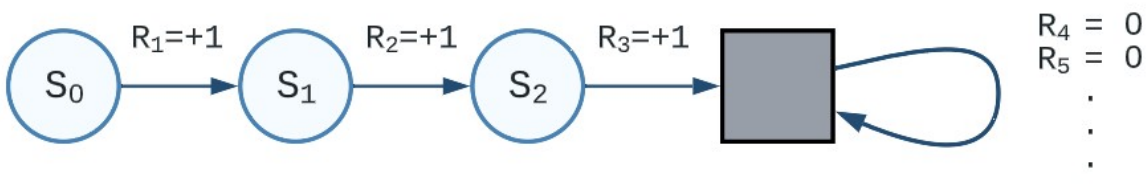
$$G_t \doteq R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + \gamma^2 R_{t+4} + \dots) \tag{2.20}$$

$$G_t \doteq R_{t+1} + \gamma G_{t+1} \tag{2.21}$$

O resultado obtido em 2.21 possui grande importância para a teoria e os algoritmos no campo do aprendizado por reforço.

Vale ressaltar que, apesar de ter sido apresentado o fator de desconto apenas no caso em que o problema possui horizonte infinito, também é possível, e por vezes adequado, usar o fator de desconto em problemas com horizonte finito. Sutton e Barto (2018) sugerem que se unifique os casos de horizonte finito e infinito considerando no caso dos problemas de horizonte finito a existência de um estado absorvente em que todas as épocas de decisão após ele retornam recompensa igual a zero.

Figura 6 – Estado absorvente em um problema de decisão sequencial



Fonte: Sutton e Barto (2018), adaptado pelo autor.

Na Figura 6 tem-se a representação de um problema de decisão sequencial com horizonte finito, mas considerando um estado absorvente, representado pelo quadrado cinza. Como é possível perceber, após a chegada ao estado terminal, todas as recompensas posteriores são iguais a zero. Assim é possível definir a equação 2.18 de uma maneira mais geral:

$$G_t \doteq \sum_{k=t+1}^{T} \gamma^{k+t+1} R_k \tag{2.22}$$

incluindo o caso em que ou $T = \infty$ ou $\gamma = 1$.

2.4.7 Políticas e funções de valor

Como será visto mais adiante, a maior parte das técnicas utilizadas para resolver problema de decisão sequencial precisam mensurar de alguma forma a qualidade de determinados estados do sistema ou de pares de ação-estado. Isso se dá porque é conveniente tomar ações que levem para estados mais promissores, viabilizando assim a maximização das recompensas acumuladas ao longo do tempo. As funções que buscam realizar essa mensuração são chamadas de funções de valor e são definidas de acordo com uma forma particular com a qual o agente atua no ambiente, também chamada de política.

Puterman (2005) cita que políticas também podem ser chamadas de plano de contingência, plano ou estratégia, e as define como uma especificação de uma regra de decisão que deve ser usada ao longo de todas as épocas de decisão. Powell (2022) afirma que políticas são como funções que mapeiam estados para decisões. De forma mais geral, mapeiam os estados para a probabilidade de selecionar alguma ação viável. Assim, considerando um agente que segue uma política π, tem-se que $\pi(a \mid s)$ é a probabilidade de se escolher a ação a dado que o estado atual é s.

Conforme citado anteriormente, as funções de valor são definidas com base na política a que estão vinculadas, isto é, a política que o agente está seguindo. De forma resumida, uma função de valor de um dado estado retorna o valor do retorno esperado quando se inicia o processo a partir desse estado e se toma todas as decisões posteriores com base na política associada à função de valor. De modo um pouco mais formal diz-se que o valor resultante de uma função de valor representada por $v_\pi(s)$, em que π é a política seguida pelo agente e s é o estado inicial do processo, é o retorno esperado quando se inicia o processo do estado s e se segue a política π.

Quando se considera o contexto de processos de decisão Markovianos, a função de

valor de um estado pode ser definida da seguinte forma:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s \right], \forall s \in \mathcal{S}, \tag{2.23}$$

em que $\mathbb{E}_\pi[\cdot]$ representa o valor esperado de uma variável aleatória considerando que o agente seguiu a política π.

A função de valor de estado apresentada em 2.23 é muito útil para mensurar o valor de um dado estado. Contudo, quando o agente precisa definir qual ação irá tomar em uma data época de decisão, é importante também mensurar quão boa é a decisão tomada. Para tanto, pode-se definir a função de valor da ação, representada por $q_\pi(s, a)$, cujo valor corresponde ao retorno esperado no caso do agente iniciar o processo no estado s, realizar a ação a e seguir a política π em todos as épocas de decisão subsequentes. A função de valor da ação pode ser definida formalmente da seguinte maneira:

$$q_\pi(s, a) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right], \forall s \in \mathcal{S}. \tag{2.24}$$

Conforme abordado anteriormente, a equação 2.21 possui uma grande importância para a área de aprendizado por reforço e da programação dinâmica. Isso se dá devido à importante relação de recursividade envolvida na equação, o que viabiliza calcular os valores esperados em 2.23 e 2.24. Com base nos estudos iniciados por Bellman (1957) em busca de formas de se obter funções que viabilizassem a construção de algoritmos eficientes para resolver problemas de decisão sequencial, foi possível deduzir uma importante equação, conforme mostrado adiante:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \tag{2.25}$$

$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \qquad \text{(por 2.21)} \tag{2.26}$$

$$= \mathbb{E}_\pi \left[R_{t+1} \mid S_t = s \right] + \gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_t = s \right] \tag{2.27}$$

A passagem de 2.26 para 2.27 é possível devido ao valor esperado ser um operador linear.

Para desenvolver a parte seguinte da equação, foi realizado o desenvolvimento de cada termo da soma separadamente, portanto, considerando o termo $\mathbb{E}_\pi \left[R_{t+1} \mid S_t = s \right]$, que representa o valor esperado da recompensa imediata seguindo a política π dado que o estado

atual é o estado s, tem-se

$$\mathbb{E}_\pi \left[R_{t+1} \mid S_t = s \right] = \sum_r r p(r \mid s) \tag{2.28}$$

$$= \sum_a \pi(a \mid s) \mathbb{E} \left[R_{t+1} \mid S_t = s, A_t = a \right] \tag{2.29}$$

$$= \sum_a \pi(a \mid s) \sum_r r p(r \mid s, a) \tag{2.30}$$

O desenvolvimento realizado entre 2.28 e 2.29 se dá devido ao termo $p(r \mid s)$ ser considerado uma distribuição de probabilidade marginal da distribuição que contem as variáveis aleatórias s' (estado seguinte) e a (ação tomada no estado atual) da seguinte forma

$$p(r \mid s) = \sum_{s'} \sum_a p(s', a, r \mid s) \tag{2.31}$$

$$= \sum_{s'} \sum_a p(a \mid s) p(s', r \mid s, a) \tag{2.32}$$

$$= \sum_{s'} \sum_a \pi(a \mid s) p(s', r \mid s, a) \qquad \text{(em que } p(a \mid s) \doteq \pi(a \mid s)) \tag{2.33}$$

Em relação ao segundo termo $\gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_t = s \right]$, que representa o valor esperado das recompensas futuras seguindo-se a política π e considerando que o estado inicial é o estado s, consideraremos G_{t+1} uma variável aleatória g que assume uma quantidade finita de valores, semelhante ao termo R_{t+1} do desenvolvimento anterior. Pode-se então desenvolver a equação da forma apresentada a seguir. Sabendo-se que

$$\gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_t = s \right] = \gamma \sum_g g p(g \mid s), \tag{2.34}$$

pode-se utilizar uma lógica similar à do primeiro termo da equação, ou seja, pode-se considerar que $p(g \mid s)$ é uma distribuição de probabilidade marginal de uma distribuição que possui as variáveis aleatórias s' (estado posterior), a (ação tomada no estado atual) e r (recompensa

imediata). Tem-se portanto a seguinte equação a distribuição $p(g \mid s)$:

$$p(g \mid s) = \sum_r \sum_{s'} \sum_a p(s', r, a, g \mid s) \tag{2.35}$$

$$= \sum_r \sum_{s'} \sum_a p(g \mid s', r, a, s) p(s', r, a \mid s) \tag{2.36}$$

$$= \sum_r \sum_{s'} \sum_a p(g \mid s', r, a, s) p(s', r \mid s, a) p(a \mid s) \tag{2.37}$$

$$= \sum_r \sum_{s'} \sum_a p(g \mid s', r, a, s) p(s', r \mid s, a) \pi(a \mid s) \qquad \text{(em que } p(a \mid s) \doteq \pi(a \mid s)\text{)} \tag{2.38}$$

$$= \sum_r \sum_{s'} \sum_a p(g \mid s') p(s', r \mid s, a) \pi(a \mid s) \tag{2.39}$$

Na equação 2.39 foi considerada a propriedade markoviana, isto é, o fato de que, dado s', a soma das recompensas futuras G_{t+1} é independente dos estados, ações e recompensas ocorridas antes do estado s', assim tem-se que $p(g \mid s', r, a, s) = p(g \mid s')$. Destaca-se que não foi citado que G_{t+1} é independente de r, a, s, mas sim que o estado s' guarda todas as informações necessária para sua mensuração.

Adicionando a equação desenvolvida entre 2.35 e 2.39 à equação do valor esperado apresentada em 2.34, tem-se o seguinte:

$$\gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_t = s \right] = \gamma \sum_g g \sum_r \sum_{s'} \sum_a p(g \mid s') p(s', r \mid s, a) \pi(a \mid s) \tag{2.40}$$

$$= \gamma \sum_g \sum_r \sum_{s'} \sum_a g p(g \mid s') p(s', r \mid s, a) \pi(a \mid s) \tag{2.41}$$

$$= \gamma \sum_r \sum_{s'} \sum_a \mathbb{E}[G_{t+1} \mid S_{t+1} = s'] p(s', r \mid s, a) \pi(a \mid s) \tag{2.42}$$

$$= \gamma \sum_r \sum_{s'} \sum_a v_\pi(s') p(s', r \mid s, a) \pi(a \mid s) \tag{2.43}$$

$$= \gamma \sum_a \pi(a \mid s) \sum_{s'} v_\pi(s') \sum_r p(s', r \mid s, a) \tag{2.44}$$

$$= \gamma \sum_a \pi(a \mid s) \sum_{s'} v_\pi(s') p(s' \mid s, a) \tag{2.45}$$

Por fim, combinando os dois termos, deve-se encontrar a equação que se busca:

$$v_\pi(s) \doteq \mathbb{E}_\pi \left[G_t \mid S_t = s \right] \tag{2.46}$$

$$= \mathbb{E}_\pi \left[R_{t+1} + \gamma G_{t+1} \mid S_t = s \right] \tag{2.47}$$

$$= \mathbb{E}_\pi \left[R_{t+1} \mid S_t = s \right] + \gamma \mathbb{E}_\pi \left[G_{t+1} \mid S_t = s \right] \tag{2.48}$$

$$= \sum_a \pi(a \mid s) \sum_r r p(r \mid s, a) + \gamma \sum_a \pi(a \mid s) \sum_{s'} v_\pi(s') p(s' \mid s, a) \tag{2.49}$$

$$= \sum_a \pi(a \mid s) \left[\sum_r r p(r \mid s, a) + \gamma \sum_{s'} v_\pi(s') p(s' \mid s, a) \right] \tag{2.50}$$

$$= \sum_a \pi(a \mid s) \left[\sum_{s'} \sum_r p(s', r \mid s, a) r + \gamma \sum_{s'} \sum_r p(s', r \mid s, a) v_\pi(s') \right] \tag{2.51}$$

$$= \sum_a \pi(a \mid s) \sum_{s'} \sum_r p(s', r \mid s, a) \left[r + \gamma v_\pi(s') \right] \tag{2.52}$$

$$= \sum_a \pi(a \mid s) \sum_{s', r} p(s', r \mid s, a) \left[r + \gamma v_\pi(s') \right] \tag{2.53}$$

A equação 2.53 é chamada de equação de Bellman para a função de valor do estado e a partir dela será possível computar, aproximar e aprender v_π, sendo portanto uma ferramenta fundamental para a criação e análise de algoritmos de programação dinâmica e aprendizado por reforço. A equação guarda a relação de recursividade já citada, mostrando uma forte relação entre o valor do estado atual e o valor estado seguinte. Além disso possui o termo $\pi(a \mid s)$ ponderando as probabilidades de ocorrência de todas os pares de estados futuros e recompensas. É importante ressaltar que a distribuição $p(s', r \mid s, a)$ é o que representa toda a dinâmica do ambiente. Essa dinâmica pode ser conhecida a priori ou desconhecida. Quando é conhecida, é possível a utilização de métodos de programação dinâmica para computar políticas ótimas; já quando é desconhecida, é necessário recorrer para métodos com aproximação para a identificação de políticas ótimas, como o aprendizado por reforço.

2.5 Processo de decisão semimarkoviano

Na seção 2.4.4 foi abordada, entre outras questões, a caracterização de processos de decisão Markovianos por meio do intervalo de tempo em que as épocas de decisão ocorrem. Muitos problemas reais podem ser modelados considerando as épocas de decisão como intervalos discretos, contudo, muitos outros necessitam que seja feita uma modelagem considerando as épocas de decisão contínuas. Foi visto que no caso em que as épocas de decisão são classificadas como contínuas, existem três formas de se definir os momentos em que o agente deve tomar

alguma decisão. Na primeira, as decisões ocorrem de forma contínua; na segunda, as decisões ocorrem quando algum evento específico é identificado pelo sistema; e na terceira, as decisões ocorrem a qualquer momento a depender do julgamento do agente.

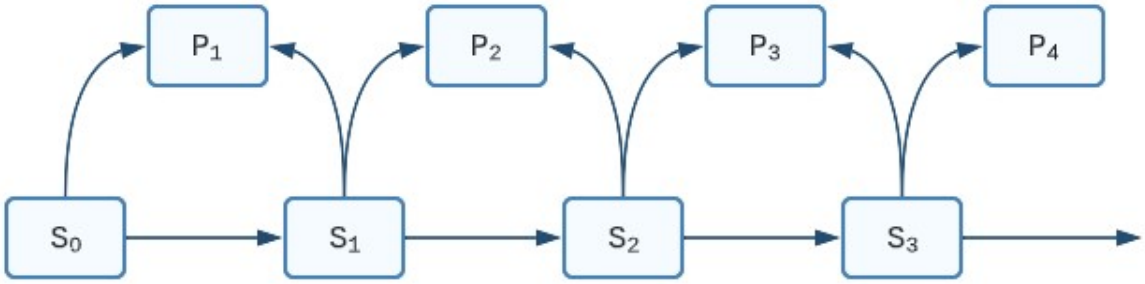
Neste trabalho tomou-se como referência a abordagem em que as decisões são realizadas quando determinados eventos ocorrem no sistema. Como neste caso as decisões não ocorrem em intervalos idênticos, utilizar uma abordagem baseada em processos de decisão Markovianos pode não ser adequado, pois os fatores influenciados pelo tempo seriam desconsiderados na modelagem. Por este motivo, decidiu-se utilizar uma abordagem baseada em processos de decisão semimarkovianos.

De acordo com Gallager (2013), um processo de decisão semimarkoviano é uma generalização de processos de decisão Markovianos em que o intervalo de tempo entre as épocas de decisão são regidos por uma distribuição de probabilidade arbitrária, isto é, o tempo entre as transições é uma variável aleatória. Gosavi (2015) cita que por trás de um processo de decisão semimarkoviano há uma cadeia de Markov chamada de "cadeia de Markov embutida", e que a diferença principal entre um processo Markoviano e um semimarkoviano é referente ao tempo em que se leva para a realização das transições de estados. Para Sutton *et al.* (1999) processos de decisão semimarkovianos são um tipo especial de processo de decisão Markoviano apropriados para modelar sistemas com tempo contínuos e eventos discretos, tendo sido este o principal motivo pela decisão de utilização dessa abordagem neste trabalho.

Vale ressaltar que os processos de decisão semimarkovianos também pode ocorrer de maneira determinística, isto é, apesar dos tempos entre as épocas de decisão não serem necessariamente unitários e iguais, eles podem ser identificados de forma exata por meio de uma função. Uma outra característica é que quando a distribuição de probabilidade que modela o tempo entre as épocas de decisão é uma distribuição exponencial, diz-se que o processo estocástico se refere a um processo Markoviano de tempo contínuo.

Nos processos de decisão semimarkovianos, tem-se que as variáveis aleatórias referentes aos tempos de permanência nos estados P_n, com $n \geq 1$, são dependentes apenas do estado atual e do estado posterior que os sistema assume, isto é, S_{n-1} e S_n, assim nota-se uma independência do P_n com todos os P_m, com $m < n$, e com todos os S_m, com $m < n - 1$, conforme Figura 7.

Figura 7 – Dependência dos intervalo de permanência em um processo de decisão semimarkoviano



Fonte: Gallager (2013), adaptado pelo autor.

2.6 Aprendizado por reforço

O aprendizado por reforço é uma área do aprendizado de máquina que possui ênfase na tomada de decisão sequencial. O principal objetivo do aprendizado por reforço é identificar políticas ótimas em um dado ambiente, isto é, políticas que levem o agente decisor a atingir a melhor recompensa possível quando considerado o processo de decisão como um todo. Para tanto, são utilizados algoritmos para estimação de funções de valor v associadas à política π em questão, as quais são representadas por v_π. Em geral a estimação das funções de valor ocorrem por meio da interação do agente com o ambiente, por exemplo identificando as melhores ações por tentativa e erro. Outras possibilidades além de aproximação de funções de valor é a aproximação direta de políticas ótimas ou do modelo, que é representado pela função de transição de estado e a função de recompensa.

É importante ressaltar que a aplicação de técnicas de aprendizado por reforço são especialmente interessantes quando não se possui um modelo da dinâmica do ambiente, ou seja, quando não se conhece a forma como o ambiente de comporta dadas as decisões do agente. Como já foi dito anteriormente, quando se possui um modelo da dinâmica do ambiente é possível utilizar técnicas de programação dinâmica para a identificação de políticas ótimas. Como este não é o caso que será tratado neste trabalho, é necessário então recorrer para técnicas de aprendizado por reforço.

Uma das principais ideias introduzidas pelo aprendizado por reforço foi a utilização de métodos de Monte Carlo juntamente com toda a teoria já existente acerca da programação dinâmica, motivo pelo qual os termos aprendizado por reforço e programação dinâmica aproximada são intercambiáveis. Métodos que utilizam essas duas abordagens de forma combinada são conhecidos como métodos de aprendizagem por diferença temporal (*Temporal-Difference*

Learning) e permitem o aprendizado direto por meio da experiência do agente no ambiente, motivo pelo qual é possível aproximar a política sem a necessidade de um modelo a priori do ambiente.

Para avaliar se uma política é melhor ou pior que uma outra política, avalia-se o retorno esperado quando se utiliza uma delas até o fim do processo ou indefinidamente ao longo de um horizonte infinito. Quando uma política π possui um valor esperado superior a uma política π', diz-se que a política π é superior ou melhor que a política π'.

Como visto anteriormente, uma forma de mensurar valores esperados quando se segue uma política específica é por meio da funções de valor. Consequentemente, tem-se que $\pi > \pi' \leftrightarrow v_\pi(s) > v_{\pi'}(s), \forall s \in \mathcal{S}$.

A política ótima possui a característica de ser sempre melhor ou igual a todas as demais políticas possíveis, o que implica possuir uma função de valor maior ou igual que as demais, e é denotada por π_*. Assim como a política ótima, pode-se também representar a função de valor de estado ótima por v_* e pode-se representá-la da seguinte forma:

$$v_*(s) \doteq \max_\pi v_\pi(s), \qquad \forall s \in \mathcal{S}. \tag{2.54}$$

Percebe-se pela definição que políticas ótimas também possuem função de valor de estado ótima. Estas políticas também compartilham da mesma função de valor de ação q_* e estas também são ótimas:

$$q_*(s,a) \doteq \max_\pi q_\pi(s,a), \qquad \forall s \in \mathcal{S}, a \in \mathcal{A}. \tag{2.55}$$

É possível também representar a função de valor de ação q_* em função de v_*, pois a função de valor de ação ótima representa o resultado esperado de se tomar uma ação a num dado estado s e depois seguir indefinidamente a política ótima:

$$q_*(s,a) \doteq \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a]. \tag{2.56}$$

Salienta-se que é possível existir mais de uma política ótima num mesmo contexto, isto é, o agente pode tomar decisões distintas em determinadas épocas de decisão. Contudo, por serem políticas ótimas, as funções de valor de estado e ação associados possuem valores iguais.

Por meio das equações já apresentadas, é possível então deduzir uma importante equação para o campo da programação dinâmica e aprendizado por reforço, a equação da otimalidade de Bellman. Sutton e Barto (2018) apresentam a dedução da equação da otimalidade

de Bellman para v_* da seguinte forma:

$$v_*(s) = \max_{a \in \mathcal{A}} q_{\pi_*}(s,a) \tag{2.57}$$

$$= \max_a \mathbb{E}_{\pi_*}[G_t \mid S_t = s, A_t = a] \tag{2.58}$$

$$= \max_a \mathbb{E}_{\pi_*}[R_{t+1} + \gamma G_{t+1} \mid S_t = s, A_t = a] \tag{2.59}$$

$$= \max_a \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \tag{2.60}$$

$$= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_*(s')] \tag{2.61}$$

De forma similar pode-se obter a equação da otimalidade de Bellman para q_*:

$$q_*(s,a) = \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a\right] \tag{2.62}$$

$$= \sum_{s',r} p(s', r \mid s, a)\left[r + \gamma \max_{a'} q_*(s', a')\right] \tag{2.63}$$

Ao se observar detalhadamente a equação 2.61, é possível perceber que esta é na verdade um sistema de equações, sendo uma equação por estado. Além disso, esse sistema de equações possui uma solução única. Se um problema tem n estados, então o sistema de equações representado pela equação de otimalidade para a função de valor de estado 2.61 possui n equações e n incógnitas.

Um importante resultado dos estudos acerca de equações de otimalidade é que políticas gulosas com respeito a v_* são políticas ótimas. Uma politica gulosa pode ser descrita de duas formas distintas, basicamente. A primeira é como uma função de 2 argumentos $\pi(s,a)$ que possui em seu contra domínio apenas os valores 0 e 1, os quais representam a probabilidade de se tomar uma ação a quando o agente observa o estado s. Considerando uma política gulosa com a respeito a v, tem-se a equação 2.64.

$$\pi(s,a) = \begin{cases} 1, \text{ se } a = \arg\max_a \sum_{s',r} p(s', r \mid s, a)\,[r + \gamma v(s')] \\ 0, \text{ caso contrário} \end{cases} \tag{2.64}$$

em que $\arg\max_a$ sinaliza que a ação a que será escolhida é a que maximiza a equação que segue.

Uma outra forma de representar uma política gulosa é como uma função que recebe como argumento um estado e retorna um ação, conforme equação 2.65 a seguir. Neste caso tem-se uma política gulosa com relação a q, pois sempre toma a ação a que maximiza a função:

$$\pi(s) = \arg\max_a q_\pi(s,a). \tag{2.65}$$

Sabendo disso, objetivando maximizar a recompensa acumulada esperada, uma vez que se tenha identificado a equação de valor de estados ótima, basta agir de forma gulosa com relação a v_*, isto é, selecionando a cada passo a ação que leva ao maior retorno esperado, ou seja, a ação ótima apresentada nas equações 2.66-2.68:

$$a_*(s) = \arg\max_a q_*(s,a) \tag{2.66}$$

$$= \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right] \tag{2.67}$$

$$= \arg\max_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s') \right]. \tag{2.68}$$

É possível notar por meio das equações 2.66-2.68 que a equação 2.66 possui uma vantagem computacional em relação às demais, uma vez que não é necessário computar todos os casos dos estados posteriores, além de não ser necessário ter conhecimento a priori da dinâmica do ambiente. Portanto, sabendo-se q_*, para se obter uma política ótima basta escolher a ação que maximiza q_* a cada época de decisão.

Vale pontuar que utilizar uma política gulosa com respeito a v_* e q_* não a torna uma política míope, pois as recompensas futuras esperadas, e por vezes descontadas, estão sendo computadas nessas funções. Isso quer dizer que mesmo escolhendo ações que maximizem de forma imediata uma dessas funções, indiretamente as recompensas futuras estão sendo ponderadas.

2.6.1 Q-Learning

O algoritmo *Q-Learning*, apresentado por Watkins (1989), foi um dos mais famosos e largos passos dados no estudo do aprendizado por reforço. Seu objetivo é estimar a função de valor da ação ótima q_* utilizando a equação da otimalidade de Bellman por meio de iterações. No campo da programação dinâmica e aprendizado por reforço esse procedimento é chamado de iteração de valor. Um outro uso para o termo *Q-Learning* é apresentado por Clifton e Laber (2020) que citam que atualmente também é utilizado para se referir a uma classe geral de problemas de aprendizado por reforço amplamente utilizado nos campos da estatística e inteligência artificial. Porém, neste trabalho o termo será utilizado para se referir apenas ao algoritmo.

Uma das principais características do algoritmo *Q-Learning* é que ele é *off-policy*, isto é, a política que é utilizada para realizar a iteração do algoritmo objetivando a convergência é diferente da política de comportamento, que é a política que o agente utiliza para a tomada de

decisão. Sutton e Barto (2018) afirmam que isso simplifica bastante a análise do algoritmo e viabiliza provas de convergência precoces. Apesar disso, uma condição necessária para que o algoritmo tenha uma correta convergência é que todos os pares estado-ação sejam suficientemente atualizados.

Além disso, uma das principais vantagens do algoritmo *Q-Learning* é a sua capacidade de se adaptar facilmente a uma variedade de cenários do mundo real que requerem testes experimentais em ambientes simulados. Isso ocorre porque o algoritmo pode interagir com amostras de transições, sejam elas observadas ou geradas por simuladores.

As iterações do *Q-Learning* são realizadas com base na equação seguinte:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right], \tag{2.69}$$

em que $\alpha \in [0, 1]$ é um termo que controla a intensidade do passo da iteração e , $\gamma \in [0, 1]$ controla o quão as recompensas futuras contribuirão no valor do par estado-ação, R_{t+1} é a recompensa imediata e Q é uma aproximação da função q_*, A_t é a ação tomada no tempo t e S_t é o estado no tempo t.

Na equação à direita em 2.69, é possível notar que o trecho final da equação, $R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)$ é uma espécie de cálculo de erro onde é realizada uma comparação entre o valor da função Q descontada por um fator γ escolhendo a ação a que a maximize no estado posterior S_{t+1} com a função Q considerando o estado atual S_t e a ação atual A_t, além de somar o valor da recompensa imediata. Este termo é conhecido como erro de diferença temporal e é com base neste erro que o algoritmo realiza a convergência da função Q iterativamente. O algoritmo do *Q-Learning* está detalhado no Algoritmo 1.

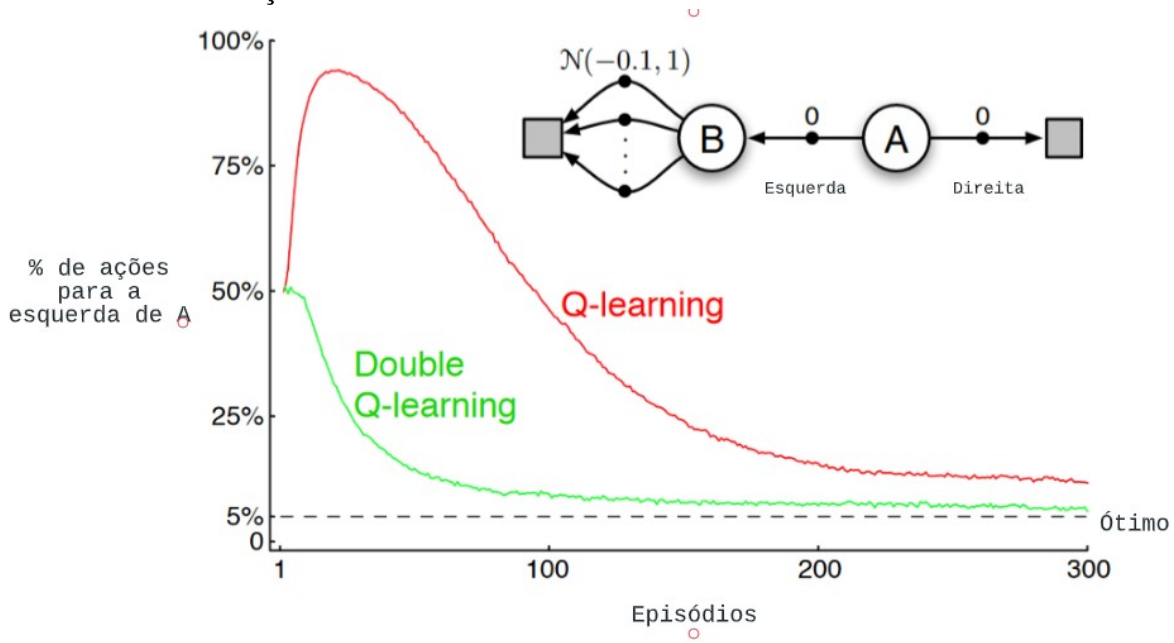
Um problema que pode eventualmente surgir com a utilização do *Q-Learning* é a chamada maximização de viés. Este problema é induzido pelo uso do operador de maximização e muitos algoritmos de aprendizado por reforço que fazem uso de operações de maximização estão sujeitos a sofrer problemas com ele. De forma intuitiva, pode-se dizer que devido ao uso do operador de maximização o modelo tende a ser excessivamente otimista, considerando de forma exagerada eventuais ruídos.

Um exemplo bastante intuitivo deste problema é apresentado por Sutton e Barto (2018). Para facilitar o entendimento considere o Gráfico 1. Pelo diagrama na área superior direita, é possível notar que existem dois possíveis estados não terminais, A e B, e dois estados terminais, representados pelos quadrados acinzentados. Neste processo de decisão o agente sempre inicia em A e pode decidir entre ir para a esquerda ou ir para a direita. Se o agente decidir

Algoritmo 1: Q-Learning para estimar $\pi \approx \pi_*$

Entrada: $\alpha \in (0, 1], \epsilon > 0$
início
  Inicialize $Q(s, a)$, para todo $s \in \mathcal{S}$ e $a \in \mathcal{A}(s)$, aleatoriamente
  para *cada episodio* **faça**
    Inicialize S
    para *cada passo do episodio* **faça**
      Escolha A para S usando uma política derivada de Q ($\epsilon-$greedy, por exemplo)
      Execute a ação A, observe R, S'
      $Q(S, A) \leftarrow Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A)\right]$
      $S \leftarrow S'$
    fim
  fim
fim

Gráfico 1 – Maximização de viés



Fonte: Sutton e Barto (2018), adaptado pelo autor.

ir para a direita ele receberá uma recompensa de 0 e o processo se encerrará devido ao estado terminal alcançado. Se o agente decidir ir para a esquerda ele também receberá a recompensa de 0, mas atingirá o estado B. No estado B existem inúmeras ações possíveis e as recompensas dessas ações seguem uma distribuição normal com média -0,1 e variância 1. Sabe-se portanto que, quando o agente escolhe ir para a esquerda, ele terminará com uma recompensa média de -0,1, que é pior que 0. No entanto, utilizando o *Q-Learning* como exemplo, ao estimar os valores das ações tomadas no estado B, devido à variância associada à distribuição normal, existirão valores acima e abaixo de -0,1. Como o *Q-Learning* utiliza o operador de maximização para

realizar atualização do valor Q, tenderá a escolher ir para a esquerda em direção ao estado B, pois a recompensa máxima das ações disponíveis neste estado em geral são superiores a zero, o que acarreta a já citada maximização de viés. Pelo Gráfico 1 é possível notar que nos episódios iniciais o *Q-Learning* possui uma grande tendência de escolher ir para a esquerda, o que é de uma forma geral indesejável.

Uma técnica que visa evitar o problema da maximização de viés é a dupla aprendizagem. A estratégia da dupla aprendizagem é manter duas funções Q_1 e Q_2, ambas aproximações de q e utilizá-las de maneira composta visando reduzir o viés. Considerando o caso em que Q_1 seja a função responsável por determinar a ação que maximiza a recompensa esperada, a composição de funções

$$Q_2(A^*) = Q_2(\max_a Q_1(a))$$

será responsável por evitar a maximização de viés, pois

$$\mathbb{E}[Q_2(A^*)] = q(A^*).$$

Quando se utiliza a técnica de dupla aprendizagem juntamente com o algoritmo *Q-Learning*, obtém-se o *Double Q-Learning*. De forma geral essa versão do algoritmo realiza os mesmo passos que o original, com apenas uma alteração principal na função de iteração, conforme equação 2.70:

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q_2(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)) - Q_1(S_t, A_t) \right]. \tag{2.70}$$

Uma possível versão do algoritmo do *Double Q-Learning* é apresentada a seguir no Algoritmo 2.

No Algoritmo 2, foi escolhido realizar a atualização de Q_1 e Q_2 na mesma frequência, isto é, na metade dos passos atualiza-se Q_1 e na outra metade Q_2. Outra particularidade é que foi escolhida um política gulosa em $Q_1 + Q_2$, mas outras escolhas também seriam possíveis.

Crites e Barto (1998) afirmam que em vários problemas não é adequado utilizar o fator de desconto γ de maneira fixa, como é feito nos casos que tratam o problema como um processo de decisão Markoviano, em que o tempo é considerado discreto. Esses problemas normalmente consideram tempo contínuo, em que as épocas de decisão podem ocorrer em tempos variáveis. Portanto se faz necessário considerar o tempo de permanência nos estados para computar a função Q. Bradtke e Duff (1994) propuseram uma versão do Q-Learning adequada para problema que consideram o tempo contínuo, como no caso da abordagem de processos de

Algoritmo 2: Double Q-Learning para estimar $Q_1 \approx Q_2 \approx q_*$

Entrada: $\alpha \in (0, 1], \epsilon > 0$
início
  Inicialize $Q_1(s, a)$ e $Q_2(s, a)$, para todo $s \in \mathcal{S}$ e $a \in \mathcal{A}(s)$, aleatoriamente
  para *cada episodio* **faça**
    Inicialize S
    para *cada passo do episodio* **faça**
      Escolha A para S usando uma política $\epsilon-$greedy em $Q_1 + Q_2$
      Execute a ação A, observe R, S'
      Com probabilidade 0.5:
        $Q_1(S, A) \leftarrow Q_1(S, A) + \alpha \left[R + \gamma Q_2(S', \arg\max_a Q_1(S', a)) - Q_1(S, A)\right]$
      caso contrário:
        $Q_2(S, A) \leftarrow Q_2(S, A) + \alpha \left[R + \gamma Q_1(S', \arg\max_a Q_2(S', a)) - Q_2(S, A)\right]$
      $S \leftarrow S'$
    fim
  fim
fim

decisão semimarkovianos. Mais adiante neste trabalho, consideraremos a iteração do Q-learning conforme apresentado em Du *et al.* (2020), isto é, da seguinte forma:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma^{t'-t} \max_a Q(S_{t+1}, a) - Q(S_t, A_t)\right]. \tag{2.71}$$

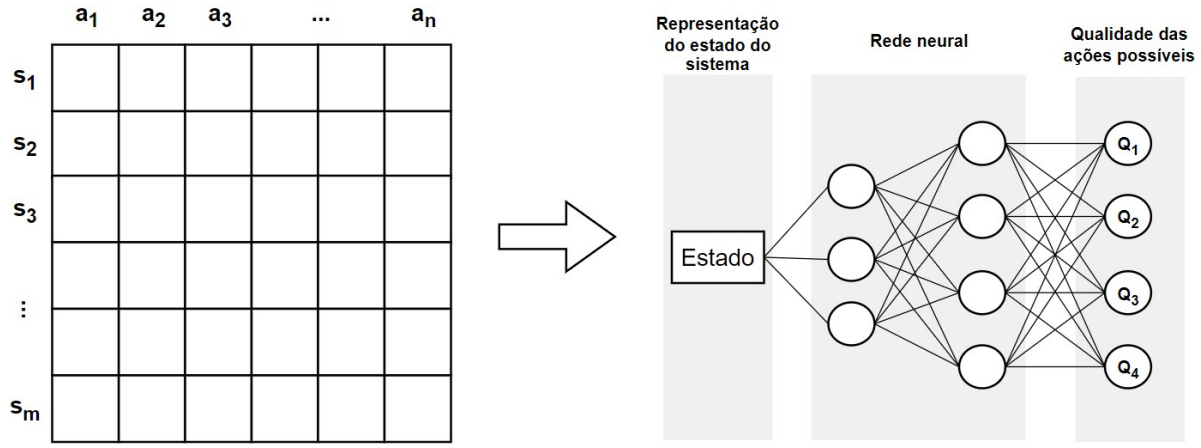
Note que a única alteração em relação à função apresentada anteriormente ocorre no fator de desconto, que passa a ser elevado pelo período de permanência em um estado, que é contabilizado subtraindo o tempo da época posterior t' pelo tempo da época atual t. O mesmo pode ser aplicado ao *Double Q-Learning*.

2.7 Aprendizado por reforço profundo

O termo aprendizado por reforço profundo se refere à utilização de aprendizado por reforço em conjunto com redes neurais profundas para aproximar qualquer um de seus componentes: função de valor, $\hat{v}(s; \boldsymbol{\theta})$ ou $\hat{q}(s, a; \boldsymbol{\theta})$, política $\pi(a \mid s; \boldsymbol{\theta})$, e modelo (referente à função de transição de estado e à função de recompensa), conforme Li (2018).

Com a utilização do aprendizado por reforço profundo tem sido possível resolver problemas que antes eram intratáveis devido à grande quantidade de estados e ações possíveis, acarretando explosão de dimensionalidade no espaço de estados e ações, de acordo com Arulkumaran *et al.* (2017). A Figura 8 demonstra de uma forma mais clara a principal diferença entre o aprendizado por reforço clássico e o aprendizado por reforço profundo. Na figura à esquerda, que retrata uma representação de uma estrutura de dados necessária para a mensuração dos valores Q

Figura 8 – Aprendizado por reforço e aprendizado por reforço profundo



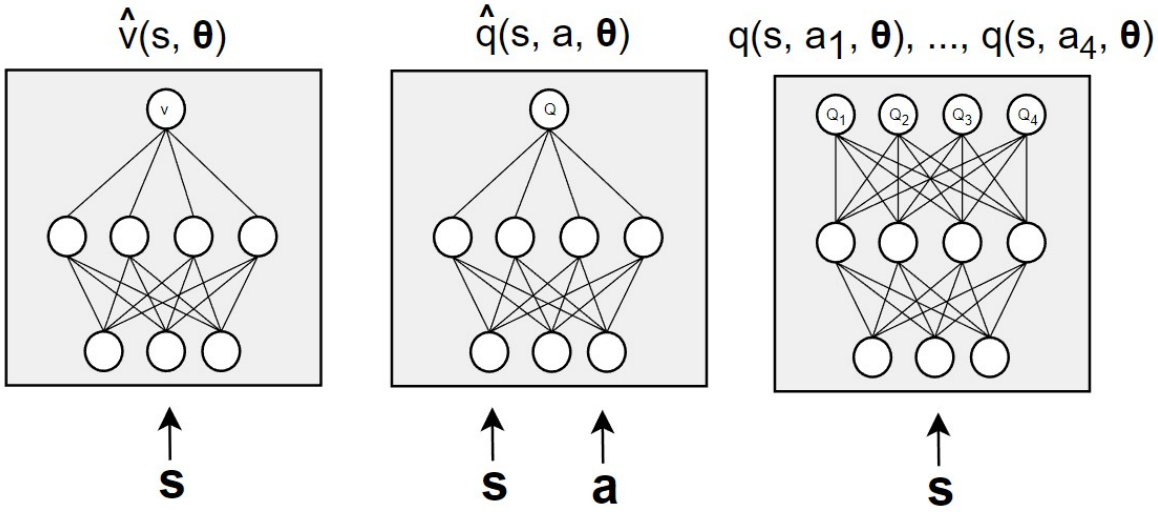
Fonte: Elaborada pelo autor.

em uma abordagem utilizando aprendizado por reforço clássico, é possível perceber que quando a quantidade de estados possíveis cresce, essa estrutura de dados ficará com uma dimensão cada vez maior, o que por vezes inviabiliza o tratamento dos dados desta maneira. Já na representação à direita, que se trata do aprendizado por reforço profundo, tem-se entre o estado e as ações possíveis a utilização de uma rede neural para a aproximação da função Q, mapeando os valores Q para cada estado que é utilizado como entrada da rede.

Nos dias atuais, diversos problemas que envolvem espaços de estados e ações de alta dimensionalidade como, por exemplo, jogos de video game, em que os estados podem ser representados pelos pixels da tela, ou problemas do campo da robótica, em que os estados possíveis podem ser representados pela combinação de inúmeros valores coletados por sensores, já estão sendo tratados com certa naturalidade por meio do aprendizado por reforço profundo, uma vez que a área está avançando com considerável velocidade.

Existem algumas abordagens clássicas para utilizar redes neurais como aproximadores das funções de valor. Na Figura 9 estão apresentadas as três principais abordagens. Na primeira representação à esquerda, utiliza-se como entrada para a rede neural um vetor de representação do estado observado pelo agente e se tem como saída o valor da função de valor do estado em questão. Neste caso está ocorrendo uma aproximação da função de valor de estado parametrizada pelos parâmetros da rede neural $\boldsymbol{\theta}$. No segundo caso da esquerda para a direita, é utilizado o estado e a ação como entrada para a rede neural. Na prática, normalmente realiza-se uma concatenação de dois vetores, um que representa o estado e outro que representa a ação. A saída da rede neural neste caso é o valor Q referente ao par estado-ação que foi utilizado como entrada da rede. Esta abordagem é utilizada principalmente em problemas que possuem a

Figura 9 – Abordagens para aproximação das funções de valor



Fonte: Elaborada pelo autor.

quantidade variável de ações possíveis, isto é, não se possui uma quantidade de ações fixas em todas as épocas de decisão. No terceiro caso da esquerda para a direita, tem-se uma abordagem que é normalmente utilizada para solucionar problemas que possuem um conjunto fixo de ações possíveis. Neste caso o estado é utilizado como entrada para a rede neural e cada neurônio de saída da rede retorna o valor Q para a respectiva ação.

Apesar do grande esforço da comunidade científica para o desenvolvimento de técnicas de aprendizado por reforço profundo robustas, durante bastante tempo aparentou ser inviável realizar o treinamento de forma estável em conjuntos de dados de alta dimensionalidade. Em 2013, Mnih *et al.* (2013) apresentaram um importante trabalho propondo um modelo de aprendizado por reforço profundo que fazia uso de uma estrutura de dados de alta dimensionalidade. Neste trabalho o modelo recebia como entrada um sinal visual com resolução 210×160, três canais de cores e uma frequência de 60Hz e o objetivo do agente era jogar jogos de Atari 2600 em situações que eram difíceis para jogadores humanos. Os testes foram realizados em sete jogos. Em seis deles o modelo performou melhor que abordagens anteriores e em três jogos o agente superou jogadores humanos especialistas.

Apesar deste trabalho ter tido grande importância, a verdadeira explosão do campo de pesquisa ocorreu em 2015, quando Mnih *et al.* (2015) publicaram o trabalho entitulado "*Human-level control through deep reinforcement learning*". De acordo com Arulkumaran *et al.* (2017) este foi o primeiro trabalho a demonstrar de forma convincente que agentes treinados por meio de de aprendizado por reforço profundo podem resolver problemas que possuem um espaço de estados de alta dimensionalidade considerando apenas sinais de recompensa. Pouco tempo

depois, em 2016, Silver *et al.* (2016) publicaram um trabalho propondo o AlphaGo, um agente treinado com aprendizado por reforço profundo que foi capaz de vencer o campeão mundial de Go.

Desde então a comunidade científica vem se dedicando ao desenvolvimento de novos métodos no campo de aprendizado por reforço utilizando aprendizado profundo, visto que existe uma ampla gama de problemas reais que podem ser beneficiados com a sua utilização. Diversos algoritmos de aprendizado por reforço "puro" já tiveram publicadas suas versões utilizando aprendizado profundo com o objetivo de superar dificuldades, principalmente relacionadas à alta dimensionalidade dos dados e às dificuldades para a aproximação de funções muito complexas.

2.7.1 Deep Q-Learning

O uso de algoritmos de aprendizagem por reforço em conjunto com redes neurais profundas possui diversos desafios. Um dos principais desafios enfrentados é que esses algoritmos sofrem com considerável instabilidade, sendo que muitas vezes podem não atingir a convergência, quando fazem uso de aproximadores não lineares para a função de valor de ação, como é o caso de redes neurais profundas. O *Deep Q-Learning*, algoritmo proposto por Mnih *et al.* (2015), buscou reduzir esses problemas utilizando algumas estratégias que até então não haviam sido aplicadas com tanto sucesso. Um dos grandes feitos desse algoritmo foi superar todos os algoritmos anteriores e performar de maneira competitiva em 49 (quarenta e nove) jogos de Atari 2600 sem a necessidade de alteração da arquitetura de rede neural e dos hiperparâmetros do modelo.

Mnih *et al.* (2015) identificaram que a instabilidade presente nesses algoritmos tem diversas causas, mas pontuaram algumas bastante importantes: a correlação associada à sequencia de observações, o fato de que pequenas atualizações na função de aproximação Q podem mudar significantemente a política, mudando a distribuição dos dados, e a forte correlação entre os valores de ação e os valores alvo $r + \gamma \max_{a'} Q(s', a')$.

O primeira abordagem buscou resolver o problema da correlação associada à sequencia dos dados. Para tanto, foi proposta a utilização de uma técnica inspirada em um mecanismo biológico chamado *experience replay*. Essa técnica consiste em armazenar toda a experiência do agente (estados ocorridos, ações realizadas e recompensas recebidas) em uma estrutura de dados e, com uma frequência ajustável, realizar o treinamento do modelo com amostras aleatórias desse conjunto de dados. Assim, observou-se que a correlação com a sequência de dados foi removida

e além disso percebeu-se uma suavização das mudanças na distribuição dos dados. A segunda abordagem, que visou resolver o problema da forte correlação entre o valor Q e o alvo, consistiu em realizar a atualização da aproximação dos valores Q para o alvo apenas periodicamente.

As *Deep Q-Networks* são normalmente representadas por $Q(s, \cdot; \boldsymbol{\theta})$, em que $\boldsymbol{\theta}$ são os parâmetros de rede neural. Conforme Van-Hasselt *et al.* (2015), o alvo dessas redes é

$$Y_t^{DQN} = R_{t+1} + \gamma \max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t). \tag{2.72}$$

Já a *Double DQN*, que busca resolver o problema do super otimismo da DQN, conforme abordado anteriormente, possui o seguinte alvo:

$$Y_t^{DoubleDQN} = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q(S_{t+1}, a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t), \tag{2.73}$$

em que $\boldsymbol{\theta}_t$ são os parâmetros da rede neural que está sendo treinada online e $\boldsymbol{\theta}'_t$ os parâmetros da rede neural alvo.

Para o caso em que o problema é modelado como um processo de decisão Markoviano, a função de perda utilizada para o treinamento da *Double DQN* é portanto:

$$\mathcal{L}_t(\boldsymbol{\theta}_t) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \left[\left(r + \gamma Q(s', \arg\max_a Q(s', a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t) - Q(s, a; \boldsymbol{\theta}_t) \right)^2 \right], \tag{2.74}$$

em que $(s, a, r, s') \sim U(D)$ indica que as amostras referentes às experiencias anteriormente obtidas pelos agentes são amostradas a partir do conjunto D de maneira aleatória seguindo uma distribuição uniforme.

Já no caso de ser formulado como um processo de decisão semimarkoviano, deve-se considerar o período de permanência no estado. O tamanho desse período afeta diretamente o fator de desconto γ, pois a equação fica da seguinte forma:

$$\mathcal{L}_t(\boldsymbol{\theta}_t) = \mathbb{E}_{(s,a,r,s')\sim U(D)} \left[\left(r + \gamma^{t'-t} Q(s', \arg\max_a Q(s', a; \boldsymbol{\theta}_t); \boldsymbol{\theta}'_t) - Q(s, a; \boldsymbol{\theta}_t) \right)^2 \right], \tag{2.75}$$

em que t é o tempo em que ocorreu a observação atual e t' o tempo em que ocorreu a observação seguinte. Importante notar que o fator de desconto γ sempre está elevado ao tempo de permanência no estado, sendo que, quando se está utilizando uma abordagem markoviana, tem-se como premissa o tempo discreto em que período de cada época de decisão tem tamanho fixo de uma unidade de tempo e por este motivo a potência não é apresentada nessas equações.

3 TRABALHOS RELACIONADOS

Com o surgimento de tecnologias como GPS (Sistema de Posicionamento Global) criou-se a possibilidade de realização de melhorias em serviços que possuem a localização como uma de suas variáveis. O problema de alocação de veículos dinâmico e estocástico é diretamente impactado pela localização dos veículos da frota e dos chamados que surgem no sistema. Em Liao (2001) um estudo foi feito com 3 grandes companhias de táxi de Singapura que migraram de um sistema de rádio para um sistema baseado em satélite que possibilitou realizar o monitoramento e identificar a localização dos veículos da frota, bem como identificar os veículos mais próximos dos chamados. Neste trabalho, uma política míope foi utilizada para a alocação de veículos a chamados, que para a época foi responsável por uma melhoria nos serviços prestados pelas empresas de táxi.

Seow *et al.* (2010) afirmaram neste trabalho que as abordagens utilizadas até o momento da publicação do trabalho eram focadas em aumentar a satisfação do usuário de forma local, alocando sequencialmente os chamados aos veículos mais próximos considerando uma política "First in First out", que é baseada em filas. Objetivando aumentar a satisfação dos usuários de forma global, bem como aumentar a satisfação dos motoristas do serviço, propuseram um sistema multi-agente colaborativo. A abordagem considerou janelas de tempo em que era possível se gerar um problema de alocação em que os recursos são os veículos da frota e a demanda são os chamados. Posteriormente era então utilizada uma técnica para resolver um problema de alocação linear de forma colaborativa, em que os agentes compartilham informações entre si, buscando atingir a melhor qualidade do serviço como um todo.

Zhang *et al.* (2017) formularam um problema de alocação cuja função objetivo consiste na soma das probabilidades de aceitação pelos motoristas. Essas probabilidades são obtidas por meio de um modelo de regressão logística, contudo, como a função objetivo do problema de alocação é não linear, foi utilizada uma heurística para resolver. Bertsimas *et al.* (2019) formularam o problema como um problema de coleta e entrega (pickup and delivery) e desenvolveram uma estratégia de re-otimização em que problemas estáticos de otimização inteira mista são resolvidos em intervalos de tempo fixos. Neste trabalho foi assumido que os veículos sempre aceitam as alocações.

Miao *et al.* (2016) propuseram uma abordagem para reposicionamento de veículos considerando tanto os dados históricos como dados em tempo real para treinar modelos que são capazes de predizer a demanda futura. Neste trabalho a região geográfica é dividida em

uma quantidade finita de sub-regiões e os veículos ociosos são orientados a realizar a ação de reposicionamento para um dessas regiões com base na demanda futura esperada. O objetivo é que os veículos possam suprir a demanda de forma mais rápida.

Wen *et al.* (2017) trataram o problema de alocação de veículos dinâmico e estocástico também pelo ponto de vista de reposicionamento de veículos. Esse tipo de abordagem prioriza manter os veículos distribuídos de tal forma que a demanda futura possa ser atendida de maneira eficiente. Para tanto, utilizaram aprendizado por reforço profundo por meio do algoritmo "Deep Q-Learning" para identificar uma boa política de decisão. Os experimentos apresentados neste estudo mostraram que a técnica utilizada possui grande potencial neste campo, que até o momento era dominado por modelos de pesquisa operacional pura. As melhorias mostradas foram principalmente em relação ao tempo de espera dos clientes. Apesar dos bons resultados obtidos, Wen *et al.* (2017) não levaram em consideração a tolerância de espera do clientes, isto é, assumiram que os clientes poderiam aguardar o atendimento por um período infinito, o que não é realista considerando atualmente que existem diversas alternativas de serviços de mobilidade urbana disponível, facilitando a troca da plataforma do serviço por parte dos usuários insatisfeitos.

Wang *et al.* (2014) também buscaram resolver o problema utilizando aprendizado por reforço profundo, mas objetivando adicionar estabilidade ao treinamento do agente fizeram utilização do algoritmo Double-DQN. Neste trabalho tanto as ações como os estados foram utilizados como entrada da rede neural, que tem como saída um único neurônio referente ao valor Q estimado. As ações são discretizadas devido ao grande espaço de decisões viáveis. Quando aproximada a função de valor, esta foi utilizada para computar os pesos das arestas do grafo bipartido referente ao problema de alocação entre os chamados e veículos. Por fim, os autores utilizaram o algoritmo de Kuhn-Munkres para resolver o problema de alocação. No trabalho de Wang *et al.* (2014) não fica claro se foi considerada uma tolerância máxima de espera por parte dos clientes. Tang *et al.* (2019) avançam no trabalho propondo agora uma formulação com base em processos de decisão semimarkovianos, além de propor uma abordagem bastante robusta para representação do estados do sistema. Apesar disso, continuam tratando o problema final como um problema de alocação em que a função de valor aprendida pelas redes neurais é utilizada como os pesos das arestas dos grafos bipartidos.

Xu *et al.* (2018) também propuseram resolver o problema fazendo uso de técnicas de aprendizado por reforço. Visto que neste trabalho o problema foi formulado como um problema

de decisão sequencial, a modelagem utilizou processos de decisão Markovianos como base. As funções de valor são computadas com base nas informações históricas do mês anterior usando programação dinâmica em seu formato tabular. As decisões tomadas pelo serviço são provenientes de soluções de problemas de alocação lineares em que a função de custo é a soma das aproximações da funções de valor dos veículos.

Liang *et al.* (2022) propuseram uma formulação do problema como um processo de decisão Markoviano. Assim como no trabalho de Wang *et al.* (2014) e Tang *et al.* (2019), neste trabalho os autores propuseram aprender as funções de valor associadas às decisões necessárias para o processo e utilizar as políticas aprendidas como entrada do simulador. O simulador, que utilizou dados reais disponibilizados pelo governo de Nova York, reserva dois momentos distintos para a tomada de decisão. O primeiro momento é utilizado para resolver o problema de alocação, em que é resolvido utilizando um modelo de programação matemática que enxerga o estado do sistema como um todo e utiliza a função de valor aprendida pelas redes neurais para calcular a função objetivo. Já no segundo momento é utilizada uma estratégia de realocação dos veículos que não estão ocupados. Neste trabalho a abordagem proposta foi comparada com diversas abordagens, como modelos centralizados para atribuir o veículo mais próximo aos chamados ou para maximização de lucro, além de técnicas de aprendizado por reforço como DQN (Deep Q-Learning) e A2C (Advantage Actor Critic). Os resultados dos experimentos mostraram que a abordagem proposta, que se baseia em programação centralizada e diferença temporal, foi superior a todas as outras abordagens testadas no que diz respeito à taxa de chamados concluídos, lucro médio e tempo de alocação.

Já pensando em cenários em que o problema de alocação de veículos dinâmico e estocástico pode ser aplicado utilizando veículos elétricos autônomos, Kullman *et al.* (2021) desenvolveram uma abordagem utilizando aprendizado por reforço profundo em que a cada época de decisão um agente central deve decidir se o veículo elétrico deve atender um chamado ou se reposicionar para realizar a recarga. Nesse trabalho os autores utilizaram *Deep Q-Networks* para aproximar as funções de valor e compararam os resultados com políticas míopes utilizando re-otimização.

4 FORMULAÇÃO DO PROBLEMA

Neste capitulo será apresentada a modelagem proposta para resolver o PAVDE e serão apresentadas as implementações dos componentes utilizados nos experimentos.

4.1 Descrição do problema como um processo de decisão semimarkoviano

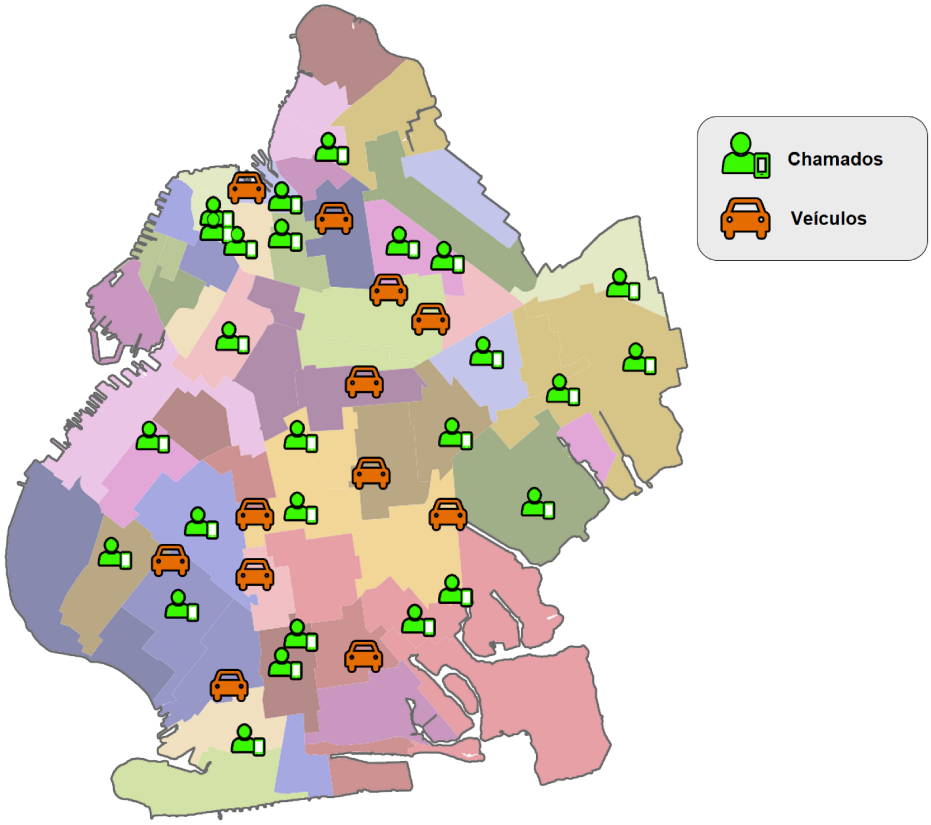
Apesar do problema deste trabalho ter forte relação com problemas clássicos que são frequentemente formulados como problemas de programação linear, devido à sua natureza dinâmica e estocástica este tipo de modelagem se torna inviável. Por este motivo, decidiu-se formular o problema utilizando o arcabouço teórico que envolve uma generalização dos processos de decisão Markovianos, mais especificamente processos de decisão semimarkovianos, tornando natural a aplicação de técnicas de aprendizado por reforço, que são adequadas para este tipo de problema. Nas seções adiante serão definidos os elementos que compõem a modelagem realizada.

4.1.1 Ambiente

O ambiente é representado pelo espaço onde os chamados podem se originar. Para representar a localização dos veículos e chamados foi utilizado um sistema de coordenadas cartesianas simples. A Figura 10 ilustra um exemplo de ambiente com uma distribuição espacial de veículos e solicitações em um determinado momento. É importante observar que nosso modelo pressupõe um conhecimento completo do ambiente, permitindo que o sistema utilize informações sobre todos os veículos e solicitações a qualquer momento, como localizações e status atual. Devido ao alto grau de informatização dos sistemas atuais de monitoração de veículos, essa é uma suposição realista.

Baseando-se na Figura 10 e objetivando encontrar uma boa solução para o problema de alocar todos os veículos disponíveis aos chamados que estão aguardando, seria necessário resolver um problema de alocação generalizado (quando considerada a restrição de tempo de espera dos cliente), tornando necessário recorrer a heurísticas, a depender do tamanho da instância do problema. Isso ocorre quando o problema é modelado como um processo de decisão Markoviano utilizando uma abordagem por *matching*. Neste trabalho a proposta é modelar o problema como um processo de decisão semimarkoviano considerando uma abordagem baseada em eventos, assim, na Figura 11 é apresentada uma representação espacial do ambiente quando

Figura 10 – Representação espacial de um momento específico no ambiente



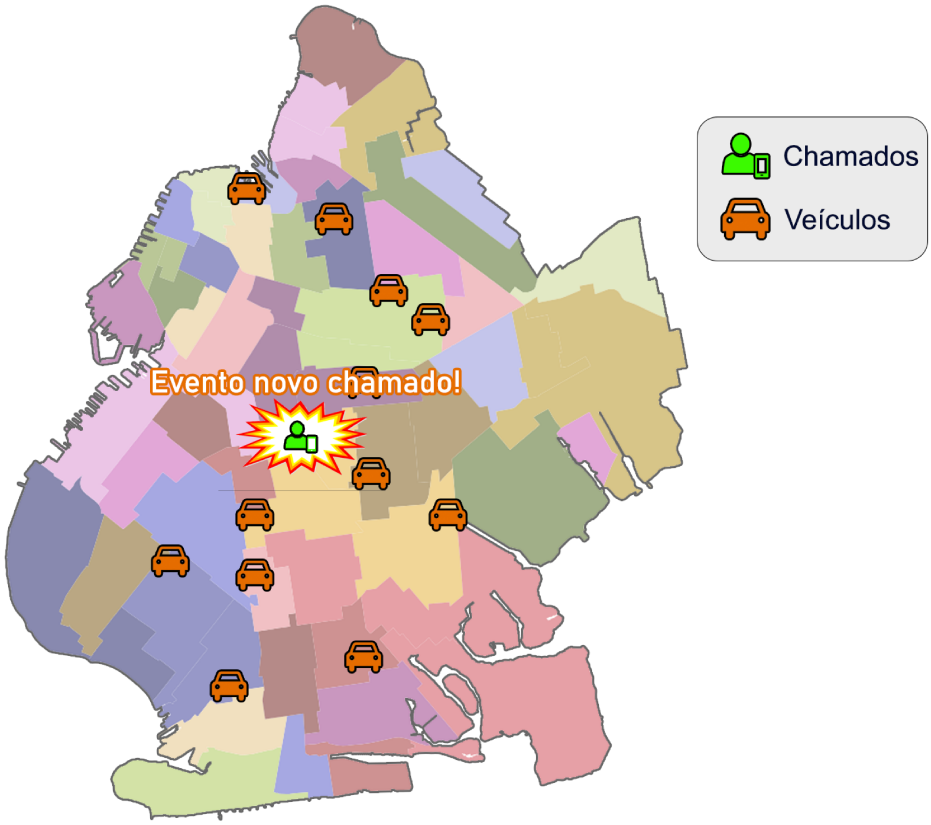
Fonte: Elaborada pelo autor.

há a ocorrência de um evento que acarreta uma época de decisão do sistema. Neste caso, tem-se uma situação em que um novo chamado chegou ao sistema e deve-se escolher qual veículo irá atendê-lo. Assim, observa-se uma simplificação do problema de alocação, uma vez que o problema se torna do tipo um-para-muitos.

No caso mostrado na Figura 12 tem-se a situação em que um veículo termina uma corrida e deve decidir qual cliente irá atender em seguida, isto é, a ocorrência de um evento "Veículo Livre". Assim como no caso anterior, tem-se uma simplificação do problema, pois nessa abordagem haverá sempre um único veículo para alocar aos chamados que estão aguardando.

Uma abordagem baseada em processos de decisão Markovianos baseada em *matching* pressupõe que as épocas de decisão ocorrem em períodos igualmente espaçados. Dessa forma, como citado anteriormente, é comum que em cada época de decisão se tenha n veículos livres e m chamados em espera, sendo $n \geq 0, m \geq 0$, conforme Figura 13. Um dos pontos negativos dessa abordagem é que o espaço de decisões pode se tornar muito grande quando há uma frota de veículos grande e um grande fluxo de entrada de chamados, o que dificulta a identificação de boas decisões, além de eventualmente inviabilizar a resolução dos problemas de

Figura 11 – Representação espacial do evento “Novo Chamado” em uma abordagem baseada em eventos



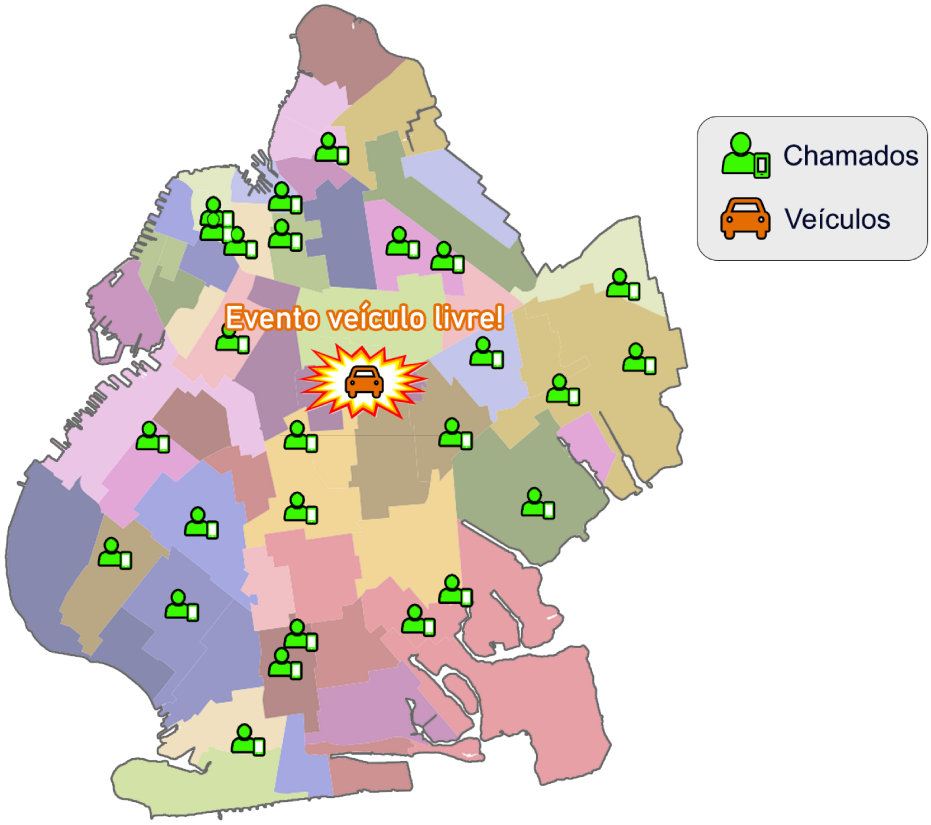
Fonte: Elaborada pelo autor.

alocação em tempos curtos, podendo aumentar a latência da resposta do sistema. Outro problema é que devido aos tempos das épocas de decisão serem fixos tem-se um acréscimo de tempo ocioso no sistema, pois uma vez que já se tem pelo menos um veículo livre e pelo menos um chamado em espera, já existe a possibilidade de realizar uma alocação. Nesta configuração é necessário aguardar até a ocorrência da época de decisão para realizar a alocação.

Note que no tempo $t = 1$ da Figura 13 algumas alocações poderiam ter sido antecipadas, como por exemplo quando o primeiro veículo fica livre. Neste momento o sistema poderia ter verificado se há compatibilidade entre o veículo livre e um dos dois chamados em espera, contudo, devido à limitação presente em uma modelagem que define períodos fixos para a ocorrência das épocas de decisão, tem-se que aguardar até o momento $t = 1$ para realizar as alocações.

Já na modelagem baseada em eventos, modelando o problema como um processo de decisão semimarkoviano, não se possuem as restrições apresentadas. Um exemplo do impacto de se modificar a modelagem do problema de um processo de decisão Markoviano para semimarkoviano é apresentado na Figura 14.

Figura 12 – Representação espacial do evento “Veículo Livre”

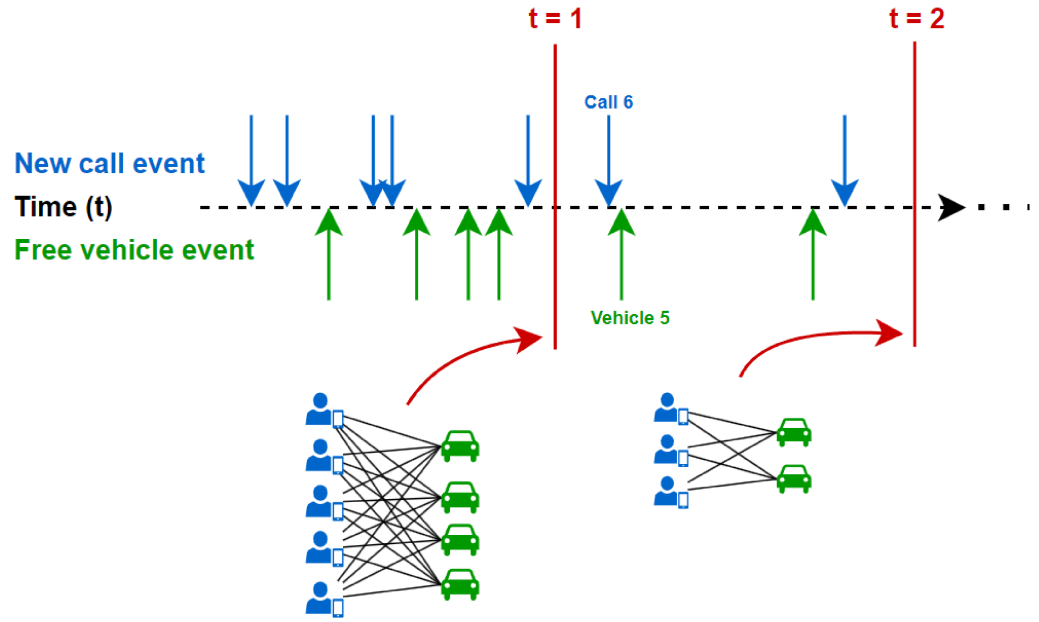


Fonte: Elaborada pelo autor.

A modelagem do problema baseada em eventos considerando tempo contínuo e épocas de decisão com períodos variáveis possui duas grandes vantagens em relação à modelagem baseada em épocas de decisão que correm em intervalos fixos. É possível perceber que as épocas de decisão, exemplificadas com as linhas vermelhas tracejadas na vertical na Figura 14, podem ocorrer em qualquer ponto do tempo, bastando apenas que alguma condição dispare um evento pré-definido. Dessa forma espera-se reduzir o tempo de espera dos chamados que chegam ao sistema, uma vez que não há necessidade de aguardar a próxima época de decisão fixada, como no caso da modelagem considerando intervalos de época de decisão fixos.

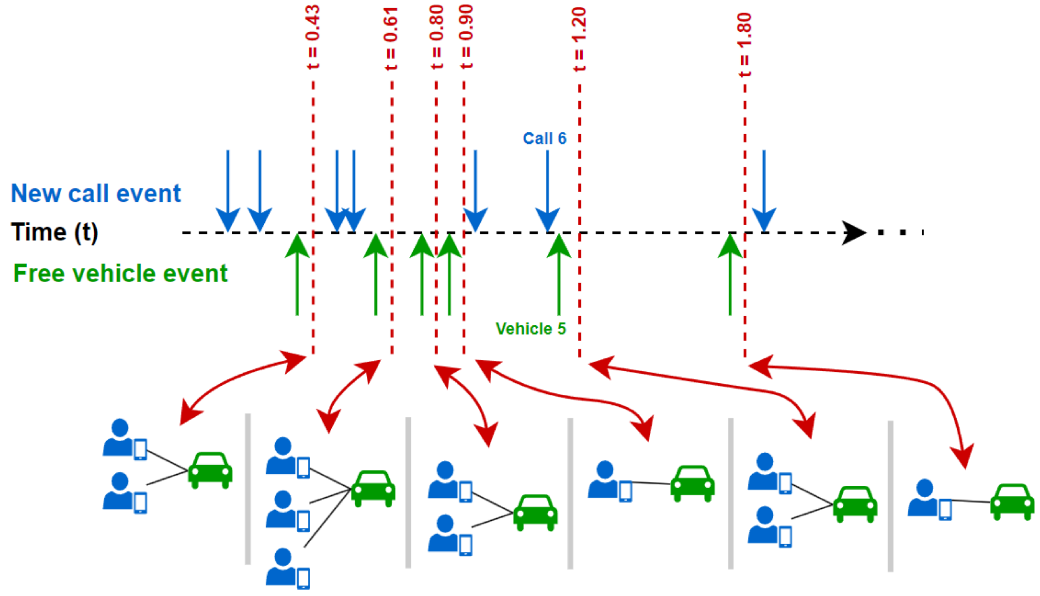
Outra importante vantagem dessa modelagem é que os problemas de alocação se reduzem a problemas um-para-muitos, onde se possui ou somente um veículo livre ou somente um chamado aguardando, diminuindo a complexidade dos problemas de alocação mesmo em situações com grandes quantidades de veículos livres e chamados em espera. Assim espera-se reduzir o custo computacional e possibilitar que o sistema faça sugestões de alocações com latências consideravelmente mais baixas.

Figura 13 – Representação das épocas de decisão em um processo de decisão Markoviano



Fonte: Elaborada pelo autor.

Figura 14 – Representação das épocas de decisão em um processo de decisão semimarkoviano



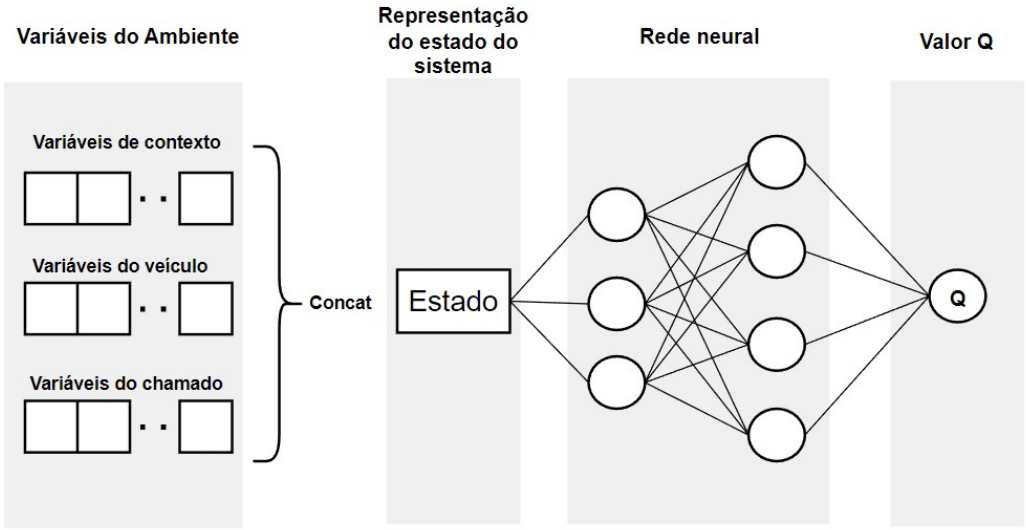
Fonte: Elaborada pelo autor.

4.1.2 Espaço de estados e espaço de ações

Os estados e ações que serão enfrentados pelos agentes são dependentes do evento disparado. Por exemplo, quando um veículo fica livre, um evento será disparado. Neste caso o estado será definido pelas características do próprio veículo, como a sua localização, e as ações serão definidas pelo conjunto de chamados aguardando atendimento. Já quando o evento

disparado é referente a um novo chamado, o estado será definido pelas características do próprio chamado e as ações viáveis serão os veículos da frota. Dessa forma, o sistema é capaz de prover uma representação de estados e ações adequada para que os agentes possam tomar decisões compatíveis com a época de decisão. Neste trabalho, essas representações serão utilizadas de forma combinada. Isto se dá devido à natureza do problema, que não garante uma quantidade fixa de decisões viáveis para cada época de decisão. Assim, será utilizada a concatenação das representações como entrada, incluindo-se variáveis de contexto, para o método que utilizaremos para realizar o treinamento do agentes decisores. Na Figura 15 é apresentado como será realizado o fluxo desses dados para ambos os agentes.

Figura 15 – Representação do vetor de entrada das redes neurais atreladas aos agentes decisores



Fonte: Elaborada pelo autor.

Na Quadro 1 são apresentadas as variáveis referentes aos veículos que foram utilizadas para realização do treinamento dos agentes. Ressalta-se que no caso das variáveis que se referem à localização dos veículos e dos chamados foi utilizado um sistema de coordenadas cartesianas simples.

Quadro 1 – Vetor de variáveis do veículo

Índice	Variável	Descrição
1	Coordenada X da Localização do Veículo	Coordenada X da localização atual do veículo. Representada em metros.
2	Coordenada Y da Localização do Veículo	Coordenada Y da localização atual do veículo. Representada em metros.
3	Coordenada X do Destino do Veículo	Coordenada X da localização para a qual o veículo está se locomovendo. Caso o veículo não esteja atendendo um chamado, é igual à localização atual. Representada em metros.
4	Coordenada Y do Destino do Veículo	Coordenada Y da localização para a qual o veículo está se locomovendo. Caso o veículo não esteja atendendo um chamado, é igual à localização atual. Representada em metros.
5	Tempo para encerrar o serviço	Tempo estimado para o veículo chegar à localização de destino do chamado que está atendendo. Caso o veículo não esteja ocupado a variável assume valor 0.
6	Probabilidade de Cancelamento	Probabilidade de não aceitação de uma alocação por parte do veículo.
7	Status do Veículo	Assume o valor 0 quando o veículo está ocioso e 1 quando o veículo está ocupado.

Fonte: Elaborada pelo autor.

As variáveis referentes aos chamados que foram utilizadas para treinar os agentes estão apresentadas na Quadro 2.

Quadro 2 – Vetor de variáveis do chamado

Índice	Variável	Descrição
1	Coordenada X da Origem do Chamado	Coordenada X da localização de origem do chamado. Representada em metros.
2	Coordenada Y da Origem do Chamado	Coordenada Y da localização de origem do chamado. Representada em metros.
3	Coordenada X do Destino do Chamado	Coordenada X da localização de destino do chamado. Representada em metros.
4	Coordenada Y do Destino do Chamado	Coordenada Y da localização de destino do chamado. Representada em metros.
5	Horário que o chamado chegou ao sistema	Horário que o cliente realizou o chamado.

Fonte: Elaborada pelo autor.

Na Quadro 3 são apresentadas as variáveis de contexto utilizadas.

Quadro 3 – Vetor de variáveis de contexto

Índice	Variável	Descrição
1	Razão recurso/demanda	Quantidade de veículos na frota dividido pela quantidade de chamadas que entraram no sistema nos últimos 15 minutos. Esta variável é atualizada pelo sistema a cada 15 minutos.
2	Variável temporal cíclica 1	Momento da semana codificado da seguinte forma: $\sin(2\pi(m/10080))$ onde m representa o minuto da semana.
3	Variável temporal cíclica 2	Momento da semana codificado da seguinte forma: $\cos(2\pi(m/10080))$ onde m representa o minuto da semana.

Fonte: Elaborada pelo autor.

Conforme apresentado na Figura 15, cada entrada da rede neural que utilizaremos para realizar o treinamento dos agentes é formada pela concatenação dos vetores de variáveis de contexto, veículo e chamado. Cada concatenação representa uma alocação em um dado momento. Os agentes recebem como entrada $n \geq 1$ tuplas que representam, cada uma, uma alocação específica, e deverá escolher entre elas aquela que for mais vantajosa para o sistema. Ressalta-se que a representação foi padronizada para ambos os agentes.

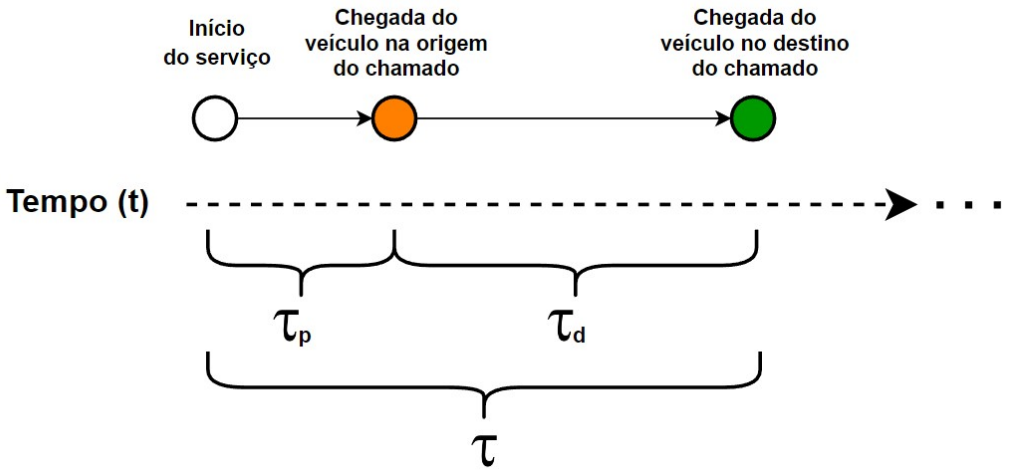
4.1.3 Função de recompensa

Quando uma alocação é realizada, inicia-se um serviço. O tempo do serviço abrange tanto o tempo de duração para o veículo chegar até a localização de origem do chamado, como o tempo para chegar até o destino, conforme apresentado na Figura 16.

Conforme citado anteriormente, o objetivo principal do trabalho é identificar políticas que minimizem o atraso médio. Essa tarefa seria fácil de ser resolvida considerando uma política que somente aceitasse chamados que possuem a localização de origem muito próxima de um veículo ocioso, contudo isso impactaria fortemente o percentual de cancelamentos de chamados e a receita obtida pela operadora do serviço. Assim, considerou-se a necessidade também de manter níveis controlados de cancelamentos de chamados e de receita obtida.

Para tanto, o racional utilizado para a definição da função de recompensa partiu da ideia de que ao reforçarmos ações que maximizem a quantidade de tempo em serviço, os agentes tendem a reduzir os tempos de atrasos, pois quanto maior o atraso, menos tempo os veículos estarão atendendo os chamados. Assim, à medida que se aumenta a receita e a quantidade de

Figura 16 – Tempos envolvidos no processo de atendimento de um chamado



Fonte: Elaborada pelo autor.

chamados atendidos, indiretamente influenciado pela maximização da recompensa, reduz-se o atraso médio dos chamados. A função de recompensa utilizada neste trabalho é apresentada a seguir:

$$R = \tau_d^{(i)} + b, \tag{4.1}$$

sendo $\tau_d^{(i)}$ a duração estimada do percurso da origem até o destino do chamado i e b um hiperparâmetro do modelo que controla o balanceamento entre a quantidade de tempo em serviço e a quantidade de chamados atendidos. Quanto maior for o valor b, mais o sistema tenderá a escolher chamados mais curtos, pois ele atua como uma espécie de taxa base, tornando o número de chamados atendidos cada vez mais importante. Quando o valor b for pequeno, então a quantidade de chamados não será tão importante, incentivando o agente a escolher chamados mais longos para maximizar a recompensa.

No contexto do problema tratado, observa-se que a função de recompensa é obtida em função do tempo de permanência em serviço dos veículos. Apesar disso, o tempo em que o veículo demora para chegar à localização de origem do chamado também influencia indiretamente a recompensa futura, pois à medida que se deseja maximizar o tempo em serviço, deseja-se também reduzir o tempo de atraso. Desta forma, considerando que a recompensa é distribuída uniformemente durante todo o tempo que um veículo demora para atender um chamado, isto é, $\tau = \tau_d^{(i)} + \tau_p^{(i)}$, foi utilizada a recompensa acumulada descontada apresentada a

seguir no processo de treinamento dos agentes:

$$R_{disc} = \frac{R}{\tau} + \gamma\frac{R}{\tau} + \gamma^2\frac{R}{\tau} + \cdots + \gamma^{\tau-1}\frac{R}{\tau} \tag{4.2}$$

$$= \frac{R(\gamma^\tau - 1)}{\tau(\gamma - 1)}, \tag{4.3}$$

sendo R a recompensa não descontada (em função apenas do percurso do chamado), γ o fator de desconto e τ o tempo de duração de todo o processo de atendimento, conforme mostrado na Figura 16.

Observando a função referente à recompensa descontada apresentada na Equação 4.2, é possível notar que ela está em função do tempo de serviço e do tempo de atraso, pois τ considera o tempo de todo o percurso do motorista até a origem do chamado e posteriormente até o destino. A função de recompensa descontada foi a função utilizada para realizar o treinamento dos agentes implementados neste trabalho.

4.1.4 Dinâmica do ambiente

A dinâmica do ambiente evolui progressivamente com o passar do tempo. As mudanças de estado são efetuadas em momentos discretos. Dessa forma, é possível que o ambiente se mantenha em um estado específico por um período indeterminado. Tomando como exemplo a localização dos veículos, esta é uma componente do estado do sistema e consideramos que alterações ocorrem somente em momentos específicos, como quando um veículo atinge a origem ou o destino de um pedido.

O cenário inclui quatro principais elementos aleatórios: As solicitações de corrida emergem de forma imprevisível em locais e tempos distintos; a transição dos veículos entre diferentes pontos pode variar em duração; há uma possibilidade dos motoristas declinarem um pedido feito pelo agente de decisão; e os passageiros possuem uma expectativa variável, porém desconhecida para os agentes, quanto ao tempo máximo que estão dispostos a aguardar.

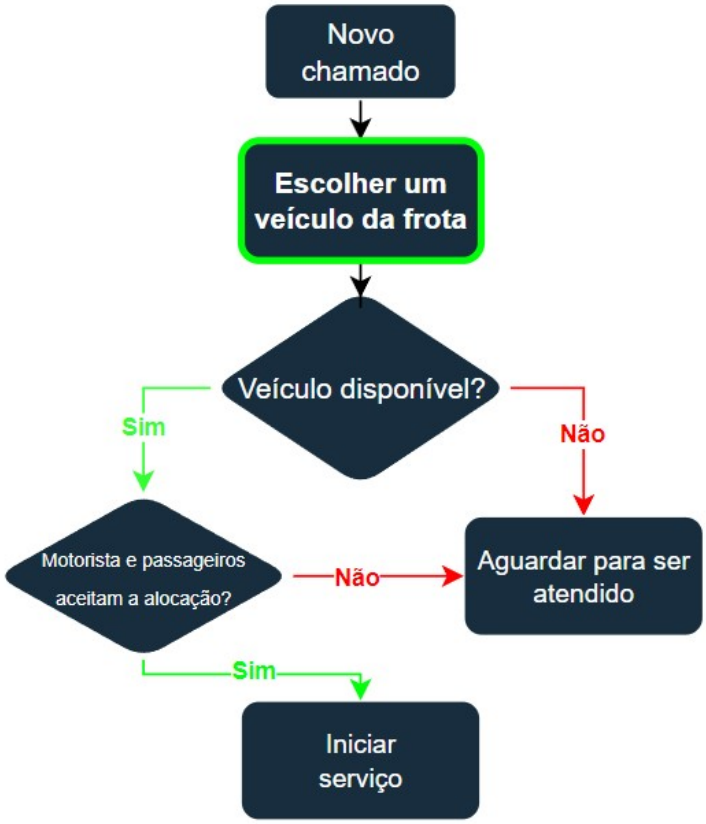
4.2 Desenvolvimento do ambiente de simulação

Dado que o problema foi modelado utilizando uma abordagem baseada em eventos, o ambiente de simulação foi implementado utilizando simulação de eventos discretos. Foi definido que os dois eventos que podem disparar um momento de decisão no ambiente são a chegada de uma nova solicitação no sistema, denominado de “Novo chamado”, e o término de uma corrida, denominado “Veículo livre”, sinalizando que o veículo está livre para atender novos chamados.

No evento do tipo “Novo chamado” o solicitante da corrida deve informar a origem, que é o ponto em que se encontra no momento da solicitação, e o destino, que é a localização para a qual se deseja ir. Já no evento do tipo “Veículo livre” não há necessidade de disponibilização de informações por parte do motorista, uma vez que esses são monitorados pelo sistema, que possui todas as informações relevantes para a tomada de decisão.

Ao chegar um chamado no sistema, imediatamente o sistema realiza uma varredura em todos os veículos da frota buscando identificar aquele que possui melhor condição de atender o cliente visando a minimização do atraso médio de todo o ambiente. Caso o veículo sugerido pelo sistema esteja disponível para atendimento, o sistema tentará realizar esta alocação, em que a efetivação depende apenas da aceitação da viagem por ambas as partes. Caso contrário o sistema não deve sugerir nenhuma ação, mantendo o chamado aguardando no *pool* de chamados em espera. O fluxo do processo que ocorre quando há a chegada de um novo chamado no sistema está apresentado na Figura 17.

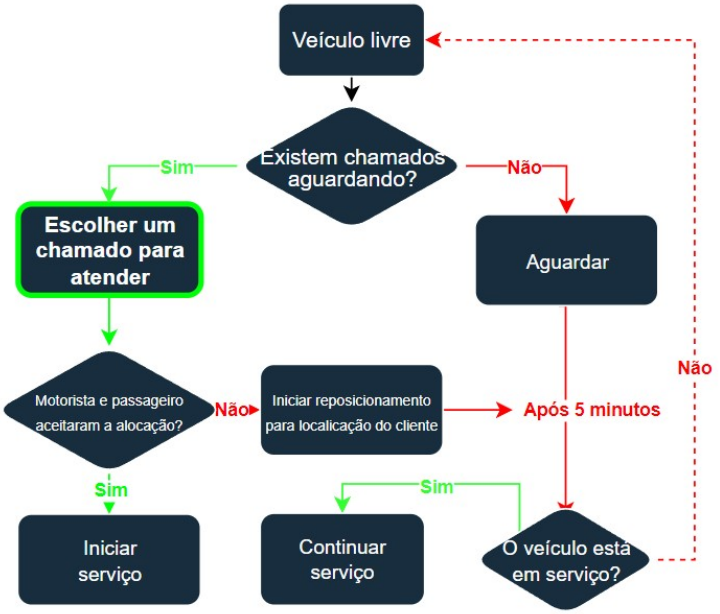
Figura 17 – Fluxo básico do evento “Novo chamado”



Fonte: Elaborada pelo autor.

Após a constatação do acionamento do evento “Veículo livre”, o sistema consulta o *pool* de chamados em espera para verificar se existe algum chamado aguardando atendimento

Figura 18 – Fluxo básico do evento “Veículo livre”



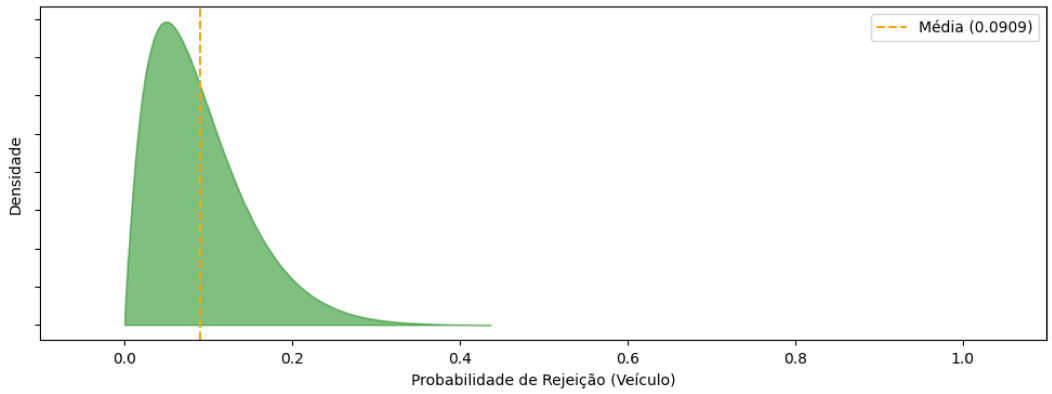
Fonte: Elaborada pelo autor.

para então prosseguir no fluxo do processo. Caso existam chamadas aguardando para serem atendidas, é iniciada uma época de decisão em que o agente decisor precisa decidir a qual chamado o veículo será alocado para então realizar o serviço. Após o sistema sugerir o chamado que o veículo deverá atender, ocorre a verificação se o passageiro e o motorista aceitaram a alocação, assim como no evento “Novo chamado”. Caso ambos aceitem a alocação sugerida pelo sistema, é iniciada a viagem e o fluxo se encerra quando o motorista deixa o passageiro na localização de destino e o evento “Veículo livre” é acionado novamente. Caso algum dos atores não aceite a alocação, o agente responsável irá sugerir que o motorista inicie uma ação de reposicionamento para a localização do chamado em que a alocação não foi concluída. A expectativa é que alguma ocorrência do evento “Nova chamado” selecione o veículo que está realizando reposicionamento nos 5 (cinco) minutos seguintes. No caso do veículo não ser alocado a nenhum chamado durante o período, o evento “Veículo livre” é acionado novamente e o fluxo é reiniciado. Já no caso de o veículo ser alocado a algum chamado durante o período, este deve interromper a ação de reposicionamento, se for o caso, e se locomover em direção à origem do chamado em questão. O fluxo básico do evento “Veículo livre” é apresentado a seguir na Figura 18. A linha vermelha tracejada representa o reinício do fluxo.

A alocação realizada pelo sistema é, nesta modelagem, uma sugestão para ambos os atores da alocação (o passageiro e o motorista), por este motivo considerou-se que ambos podem aceitar ou rejeitar a alocação sugerida, motivo pelo qual há uma etapa no fluxo do processo que

faz esta verificação. Para o sistema, existe uma probabilidade de cancelamento associada a cada veículo, sendo que, no ambiente de simulação foi amostrado um valor entre 0 e 1 para cada veículo a partir de uma distribuição beta com parâmetros $\alpha = 2$ e $\beta = 20$, conforme Figura 19, representando a probabilidade de cancelamento fixa para cada um e cada vez que ocorre uma tentativa de alocação por parte do sistema é realizado um experimento de Bernoulli em que a probabilidade de sucesso é a probabilidade que foi amostrada anteriormente. Em ambiente de produção tal probabilidade pode ser inferida diretamente por meio da frequência histórica de cancelamentos de cada motorista. Em relação ao passageiro, o cancelamento pode ocorrer devido a uma previsão de período de espera que extrapole sua tolerância máxima. Para fins de simulação, a tolerância máxima de espera dos clientes foi amostrada a partir de uma distribuição gama, que possui como suporte números positivos. Os parâmetros definidos para a distribuição foram $\alpha = 30$ e $\beta = 1$, e a forma obtida está apresentada na Figura 20. Ressalta-se que o período máximo de espera considera todo o tempo entre o momento da solicitação de corrida e a chegada do veículo à localização de origem.

Figura 19 – Distribuição da taxa de cancelamento de corridas por parte dos motoristas (distribuição beta com $\alpha = 2$ e $\beta = 20$)

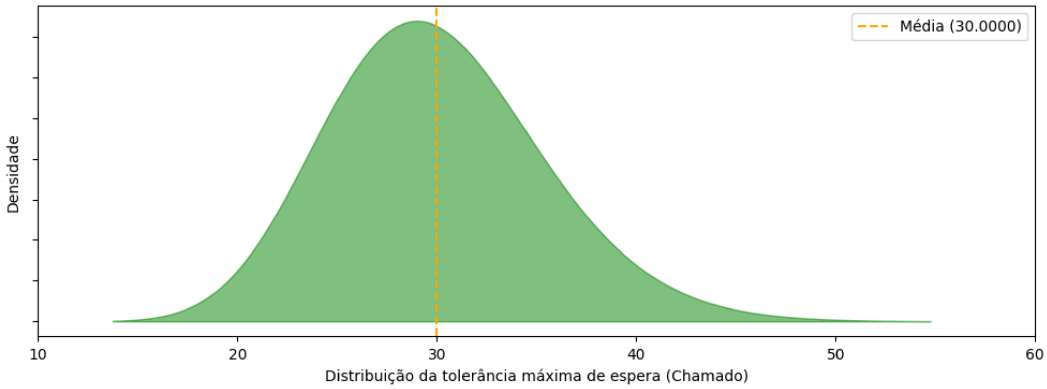


Fonte: Elaborada pelo autor.

A simulação realizada considerou tanto dados reais como dados simulados provenientes das distribuições de probabilidade previamente apresentadas. O treinamento dos agentes foi realizado simultaneamente à medida que o ambiente de simulação evoluía. Na Figura 21 é apresentado como foi organizado o fluxo de treinamento completo.

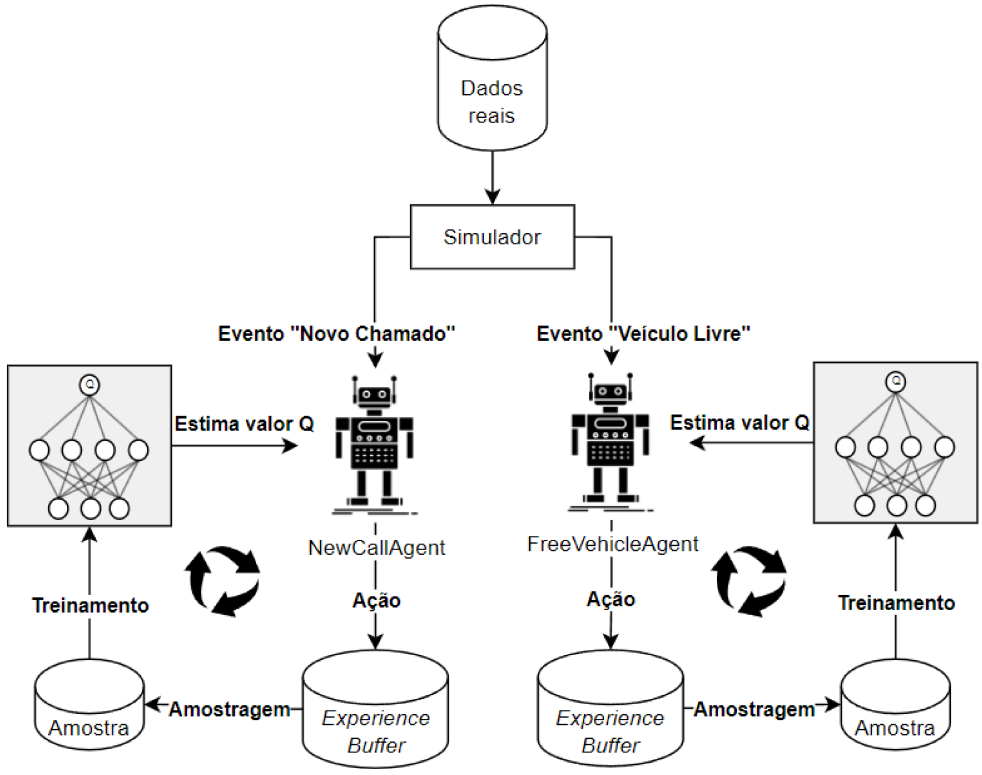
À medida que os eventos vão ocorrendo, os agentes vão tomando decisões. As experiências obtidas vão sendo armazenadas em um banco de dados. Este banco de dados é responsável pela amostragem aleatória de experiências passadas e treinamento dos agentes para a estimação do valor Q obtido. Este ciclo ocorre até a convergência das redes neurais que estão

Figura 20 – Distribuição do tempo máximo de espera por parte dos solicitantes (distribuição gama com $\alpha = 30$ e $\beta = 1$)



Fonte: Elaborada pelo autor.

Figura 21 – Fluxo geral de treinamento dos agentes



Fonte: Elaborada pelo autor.

aproximando as funções de valor de ação.

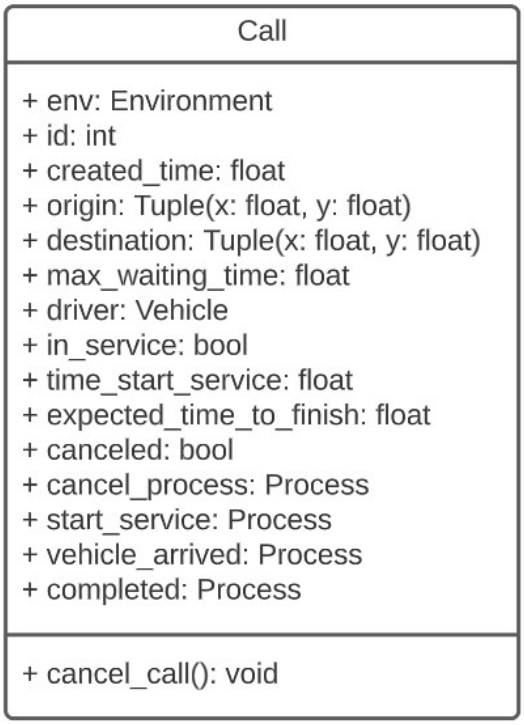
Todas as implementações realizadas neste trabalho utilizam a linguagem Python na versão 3.7.13. As principais bibliotecas utilizadas estão apresentadas na Tabela 4. Como partes principais da implementação realizada, podemos destacar as entidades básicas envolvidas no problema, os agentes e o ambiente. Cada um desses possui um papel importante para a realização dos experimentos e portanto serão detalhados separadamente a seguir.

Quadro 4 – Principais bibliotecas utilizadas

Biblioteca	Versão
NumPy	1.21.6
Pandas	1.3.5
GeoPandas	0.10.2
Simpy	4.0.1
Matplotlib	3.5.1
PyTorch	1.8.2

Fonte: Elaborada pelo autor.

Figura 22 – Diagrama da classe *Call* (Chamado)



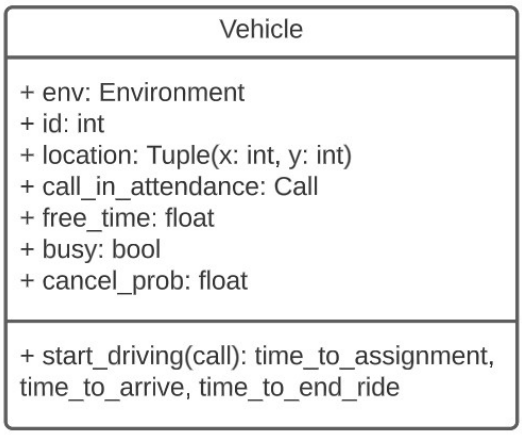
Fonte: Elaborada pelo autor.

4.2.1 Entidades básicas do ambiente

Como entidades básicas do ambiente temos o objeto chamado (*Call*) e o objeto veículo (*Vehicle*). Cada uma das entidades possui significativa importância para os momentos de decisão que serão abordados posteriormente.

O diagrama apresentado na Figura 22 detalha os atributos e métodos da classe que modela o comportamento dos chamados. Como é possível observar no diagrama da classe dos chamados, os atributos relacionados ao tempo estão implementados como o tipo *float*. Isso ocorre devido ao ambiente considerar como unidade básica de tempo o minuto, ou seja, representa-se uma hora como 60 unidades de tempo, duas horas como 120 unidades de tempo, 30 segundos com 0,5 unidades de tempo e assim por diante.

Figura 23 – Diagrama da classe *Vehicle* (Veículo)



Fonte: Elaborada pelo autor.

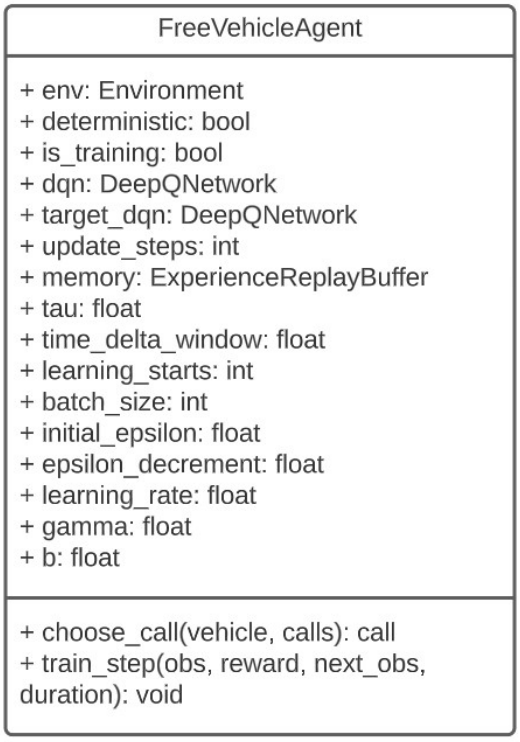
Destaca-se também o atributo “cancel_process” que é do tipo *Process*. Trata-se de um processo que é iniciado quando ocorre a instanciação de um objeto do tipo *Call* e seu objetivo é realizar o monitoramento do tempo máximo que uma chamada pode esperar, representado pelo atributo “max_waiting_time”. Quando o processo identifica que o tempo limite foi atingido, a chamada é cancelada por meio do método “cancel_call‘’, sendo o atributo “canceled” alterado para o valor verdadeiro. Quanto aos outros processos, estes são utilizados para sinalizar ao agente a ocorrência de momentos importantes que são utilizados para o treinamento dos modelos.

O diagrama da classe que representa os veículos do ambiente está apresentado na Figura 23. É perceptível que este diagrama possui uma estrutura bastante parecida com a anterior, contudo possui especifidades importantes. Como se pode observar, a partir da classe “Vehicle” é possível acessar o objeto “Call” referente à chamada que está em atendimento por meio do atributo “call_in_attendance”.

O método “start_driving”, que é o núcleo de todo o processo, é utilizado para iniciar uma corrida em direção ao chamado que é passado como argumento. Basicamente este método pode ser acionado de duas formas distintas: quando uma nova chamada chega ao sistema ou quando o veículo termina um chamado e logo após o sistema sugere uma alocação que o veículo e o chamado aceitem.

Destaca-se também o atributo “cancel_prob”, que é um valor numérico real entre 0 e 1, representando a probabilidade de um veículo cancelar uma chamada. Do ponto de vista frequentista, numa aplicação real esse atributo pode ser mensurado por meio da quantidade de chamadas canceladas em um determinado espaço de tempo dividido pelo total de chamadas atribuídas a este veículo no mesmo espaço de tempo. No ambiente simulado estes valores foram

Figura 24 – Diagrama da classe *FreeVehicleAgent* (agente responsável pelo evento "Veículo livre")



Fonte: Elaborada pelo autor.

amostrados por meio de uma distribuição beta com parâmetros $\alpha = 2$ e $\beta = 20$, conforme Figura 19.

Sendo assim, observa-se que a maior parte dos veículos tende a cancelar poucos chamados, e à medida que a probabilidade de cancelamento cresce, diminui-se a quantidade de veículos. Além disso nenhum veículo possui uma probabilidade maior que 0,15 de cancelamento, o que pode ser interpretado como um limite de frequência de cancelamento por parte dos veículos.

4.2.2 Agentes

Como já citado anteriormente, na modelagem apresentada neste trabalho foram desenvolvidos dois agentes, sendo um responsável pelos eventos "Veículo livre" e outro responsável pelos evento "Novo chamado". A classes desenvolvidas para modelar os agentes são especialmente importantes devido a serem as responsáveis pela tomada de decisão, o que as torna determinantes para o atingimento de bons resultados.

A estrutura da classe "FreeVehicleAgent", que é responsável pelo evento "Veículo livre", está apresentada na Figura 24. Percebe-se por meio do diagrama alguns atributos que são utilizados apenas para controle entre os momentos de treinamento e os momentos em que o

agente será utilizado para prover decisões de fato. O atributo "*deterministic*" é o responsável por definir se o agente utilizará um política *"ϵ-greedy"* ou se utilizará sempre a política que foi treinada de maneira gulosa. O atributo "*is_training*" é utilizado apenas para definir se o agente deve realizar passos de treinamento, isto é, atualizar os parâmetros das redes neurais e incluir experiências na "memória" do agente, ou não.

Os atributos "*dqn*" e "*target_dqn*" são as redes neurais responsáveis por aproximar a função de valor de ação por meio da função Q. A utilização das duas "Deep Q-Networks" se deu devido à utilização do algoritmo "Double-DQN", que, conforme explicado na seção 2.6.1, é mais estável que utilizar apenas uma DQN. Nessa configuração a DQN alvo $\hat{Q}$ é mantida fixa e atualizada periodicamente. O período de atualização é definido por meio do atributo "*update_steps*". O atributo "*memory*" é o responsável por manter o histórico das experiências obtidas pelo agente, isto é, é uma implementação do "Experience Replay Buffer". O atributo "*tau*" é utilizado para definir a suavização da atualização dos parâmetros da DQN alvo $\hat{Q}$, sendo a atualização definida por

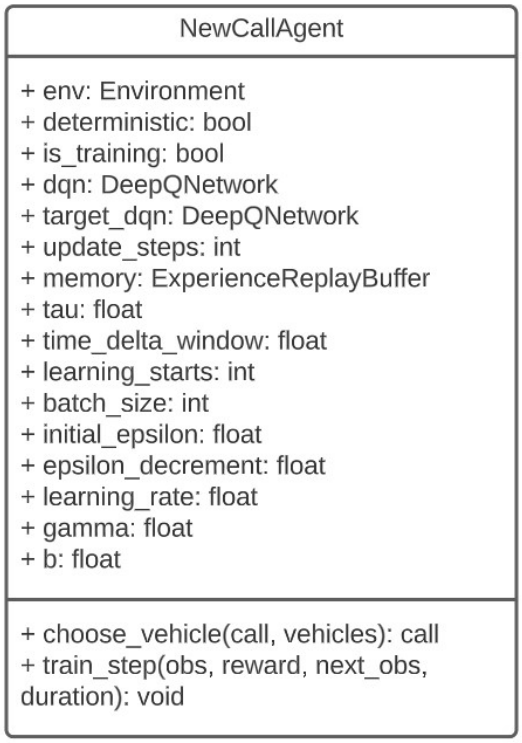
$$tau \times dqn\_weights + (1 - tau) \times target\_dqn\_weights,$$

em que $dqn\_weights$ são os pesos dos neurônios da DQN responsável pelo comportamento do agente e $target\_dqn\_weights$ os pesos dos neurônios da DQN alvo. O atributo "gamma" é o fator de desconto. O atributo "b" se refere à parcela fixa que será considerada na função de recompensa não descontada, conforme apresentada na Equação 4.1. Esta equação é utilizada na função de recompensa treinada pelos agentes, que é a Equação 4.2. Os demais atributos são referentes aos parâmetros das redes neurais utilizadas na DQN Q e na DQN alvo $\hat{Q}$. Ressalta-se que as duas redes neurais possuem a mesma arquitetura e mesmos hiperparâmetros.

O método "*choose_call*" é responsável por retornar uma chamada escolhida entre as chamadas recebidas como parâmetro. Quando o agente está em treinamento, há a possibilidade dessa escolha ser aleatória com o objetivo de explorar o espaço de decisões possíveis. Já o método "*train_step*" é o responsável pela lógica de treinamento das redes neurais e pelo registro da tupla referente à experiência do agente.

Quanto à classe "*NewCallAgent*", tem-se uma estrutura similar, diferenciando-se apenas pelo método "choose_call" que nesse caso se torna "choose_vehicle", conforme pode ser visto no diagrama apresentado na Figura 25.

Figura 25 – Diagrama da classe *NewCallAgent* (agente responsável pelo evento "Novo chamado")

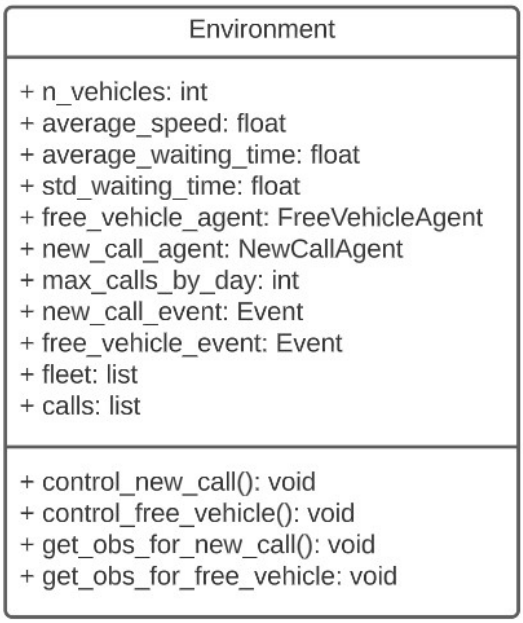


Fonte: Elaborada pelo autor.

4.2.3 Ambiente

A classe responsável por modelar o ambiente em que o processo ocorre foi chamada de "Environment". A parte principal de sua estrutura está apresentada na Figura 26. A partir da classe "Environment" é possível ter total controle sobre a forma como o experimento simulado será realizado, o que impacta diretamente na forma como os agentes serão treinados, pois as políticas obtidas por eles podem divergir quando treinadas em ambientes distintos. O atributo "n_vehicles" é a responsável por definir qual o tamanho da frota de veículos no início do processo. O atributo "average_speed" define a velocidade média que os veículos se locomovem no território. Os atributos "average_waiting_time" e "std_waiting_time" são os parâmetros da distribuição que será responsável por gerar amostras do tempo máximo de espera para cada novo chamado que chega ao sistema. Os atributos "free_vehicle_agente" e "new_call_agent" são os responsáveis por definir quais os agentes que atuaram no ambiente. O atributo "max_calls_by_day" é utilizado apenas no momento do treinamento dos agentes para limitar a quantidade máxima de chamados por dia. Os atributos "new_call_event" e "free_vehicle_event" são responsáveis por manter os eventos que podem disparar as épocas de decisão no sistema. Os atributo "fleet" e "calls" são,

Figura 26 – Diagrama da classe *Environment* (Ambiente)



Fonte: Elaborada pelo autor.

respectivamente, a lista de veículos da frota e a lista de chamados recebidos pelo sistema.

Quanto aos principais métodos da classe “Environment”, tem-se os dois métodos “control_new_call” e “control_free_vehicle” que são utilizados para execução de instruções sempre que for detectado o disparo de algum dos eventos “new_call_event” ou “free_vehicle_event”. Ambos os métodos executam em um *loop* infinito, visto que devem estar ativos durante toda a simulação. Esses métodos são responsáveis por iniciar e direcionar os fluxos apresentados nas Figuras 17 e 18. Já os métodos “get_obs_for_new_call” e “get_obs_for_free_vehicle” são os responsáveis por prover os estados que serão observados pelos agentes. Como explicado anteriormente, apesar da representação de estados ser padronizada para ambos os agentes, a decisão que eles devem tomar é diferente, pois enquanto o agente responsável pelo evento “Veículo livre” precisa escolher qual chamado irá atender, o agente responsável pelo evento “Novo chamado” precisar escolhe qual veículo irá atender o chamado em questão. As localizações dos veículos e chamados foram representadas por meio de um espaço de duas dimensões em que os valores de cada um dos eixos é representado em metros.

5 EXPERIMENTOS E RESULTADOS

Neste capitulo serão apresentados os detalhes acerca do experimento realizado, bem como os resultados obtidos nos cenários construídos.

5.1 Configuração do experimento

Tendo em vista o objetivo de utilizar dados reais para a realização dos experimentos, foram utilizados os dados abertos disponibilizados no site oficial do governo de Nova York[1]. Os dados de corridas realizadas por plataformas de mobilidade urbana com alto volume de chamados estão disponíveis para os bairros do Bronx, Brooklyn, Manhattan, Queens e Staten Island. Neste trabalho foram escolhidos os chamados localizados no Brooklyn. Essa decisão se deu por este bairro se tratar do mais populoso da cidade de Nova York. Além disso foram considerados apenas os chamados realizados por meio das empresas Uber e Lyft. Após esses dois filtros, a base de dados utilizada para treinamento, que abrange as corridas realizadas no mês de janeiro de 2022, contou com 3.129.261 de registros, em que cada registro representa um chamado. Quanto à base de dados utilizada para os testes, que contém os chamados ocorridos no mês de fevereiro de 2022, totalizou 3.199.846 registros de chamados. As informações da base de dados que foram utilizadas para o experimento foram o horário do chamado, a localização de origem e a localização de destino.

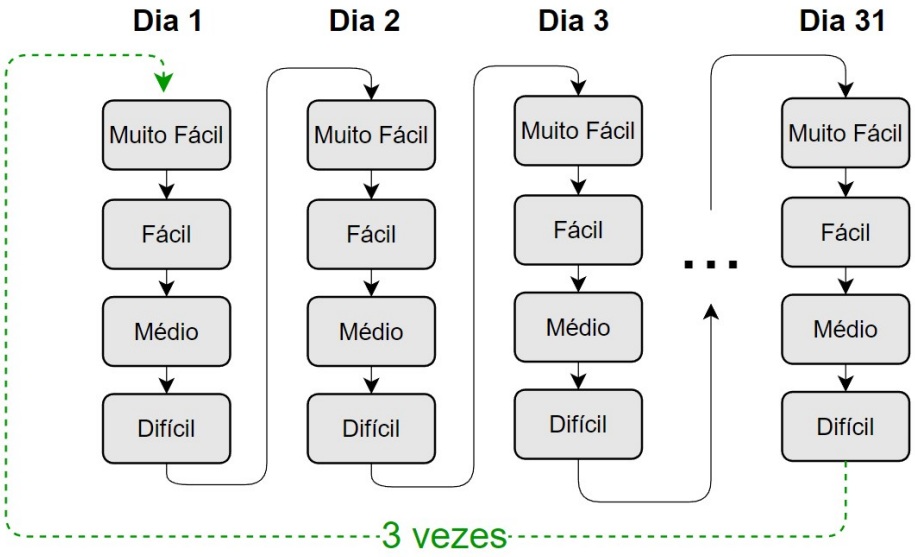
Na etapa de treinamento dos agentes decidiu-se treinar ambos os agentes simultaneamente, isto é, no início do processo de treinamento tanto o agente responsável pelo evento "Veículo livre" como o agente responsável pelo evento "Novo chamado" estarão tomando decisões de maneira aleatória e à medida que os episódios passam suas ações vão de ajustando ao ambiente utilizando uma política ϵ-*greedy*.

Como o treinamento dos agentes é muito influenciado pelos parâmetros do ambiente, foram consideradas quatro situações para cada dia do mês de treinamento: uma situação com quantidade baixa de veículos, uma com quantidade média, uma com quantidade alta e uma com quantidade muito alta. Nos experimentos essas quantidades foram definidas como percentuais do total de chamados esperados no dia e os valores escolhidos foram 0.5%, 1.0%, 2.0% e 3.0% .

O treinamento dos agentes foi realizado considerando a quantidade fixa de 1.000 chamados diários e a ordem na qual foram simulados os cenários está detalhada na Figura 27.

[1] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

Figura 27 – Fluxo do treinamento realizado



Fonte: Elaborada pelo autor.

Essa configuração de treinamento foi realizada visando obter uma política genérica adequada para diferentes cenários, evitando que os agentes se especializem em cenários específicos. Considerando todo o processo de treinamento os agentes foram treinados por 372 dias.

Todas as redes neurais utilizadas no experimento foram *Multilayer Perceptrons* com duas camadas ocultas, em que a quantidade de neurônios de cada camada é definida nos hiperparâmetros das instâncias dos agentes. Quanto às funções de ativação utilizadas, foi escolhida a função de ativação *Leaky Relu*. A principal justificativa para essa escolha é que a *Leaky Relu* tende a apresentar melhores resultados em situações em que a *ReLU* pode apresentar problemas, como por exemplo, quando há um grande número de neurônios "mortos" na camada oculta. Isso ocorre porque, ao contrário da *ReLU*, a *Leaky Relu* tem uma inclinação ligeiramente positiva para valores negativos, o que permite que esses neurônios "mortos"possam contribuir com informações úteis para a rede neural. Além disso, a *Leaky Relu* tende a ter uma convergência mais rápida em algumas situações, o que pode ser benéfico para o treinamento da rede neural. Quanto à inicialização dos pesos das redes neurais, foi utilizado o método *Kaiming* utilizando a distribuição uniforme. Uma das principais vantagens dessa inicialização de pesos é que ela leva em consideração a variância dos pesos de entrada e saída de cada neurônio, tornando a inicialização mais adaptativa e eficiente. Isso resulta em uma distribuição mais adequada de pesos, o que evita tanto a saturação da ativação dos neurônios quanto o problema de *Vanishing Gradient*.

Os hiperparâmetros utilizados para a instância do objeto “FreeVehicleAgent” estão apresentados na Tabela 5.

Quadro 5 – Hiperparâmetros do agente “*FreeVehicleAgent*”

Hiperparâmetro	Valor
learning_starts	10000
epsilon	1,0
epsilon_min	0,05
epsilon_decrement	0,99995
tau	1
gamma	0,9
update_steps	10.000
memory_size	20.000
n_neurons_hl_1	64
n_neurons_hl_2	32
batch_size	32
learning_rate	0,001
b	5

Fonte: Elaborada pelo autor.

Importante ressaltar que os hiperparâmetros referentes às redes neurais associadas ao agente são utilizados nas duas redes do algoritmo *Double DQN*, as quais eventualmente se diferenciam apenas pelos seus pesos.

No Quadro 6 estão apresentados os hiperparâmetros utilizados para a instanciação do agente “NewCallAgent”.

O hiperparâmetro learning_starts define qual a quantidade mínima de experiências que um agente deve ter para começar o processo de aprendizado. O epsilon é a probabilidade inicial de uma ação ser escolhida de maneira aleatória considerando uma distribuição uniforme, o epsilon_decrement é a taxa de redução do epsilon à medida que o agente realiza uma atualização nos pesos das redes neurais e o epsilon_min é o valor mínimo definido para o epsilon. O update_steps define a frequência em que os pesos das redes neurais serão sincronizados e o tau é a taxa de suavização dessa sincronização. O valor 1 deste hiperparâmetro significa que os pesos da rede neural responsável pelo comportamento substitui completamente os pesos da rede neural alvo no momento da sincronização. O gamma é o fator de desconto.

Em relação ao ambiente, representado pela classe “*Environment*”, ficou definida com o valor de 20km/h a velocidade média que os veículos se locomovem pelo território. O tempo considerado no ambiente é contado em minutos e cada episódio equivale a 1440 unidades, isto

Quadro 6 – Hiperparâmetros do agente "*NewCallAgent*"

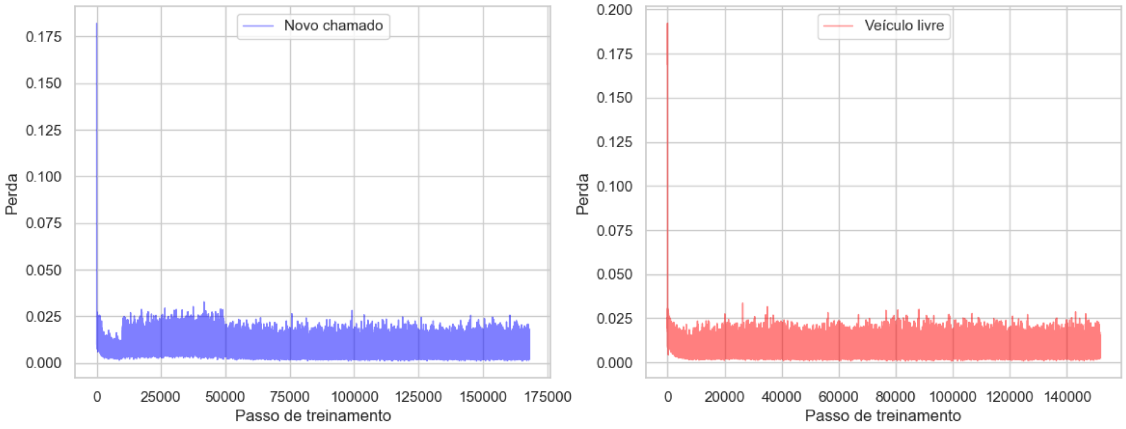
Hiperparâmetro	Valor
learning_starts	10000
epsilon	1,0
epsilon_min	0,05
epsilon_decrement	0,99995
tau	1
gamma	0,9
update_steps	10.000
memory_size	20.000
n_neurons_hl_1	64
n_neurons_hl_2	32
batch_size	32
learning_rate	0,001
b	5

Fonte: Elaborada pelo autor.

é, 24 horas. Para o cálculo das distâncias foi utilizada a *City Block*, também conhecida como distância de Manhattan.

O Gráfico 2 apresenta a função de perda das duas redes neurais dos agentes ao longo do processo de treinamento. Observa-se que ambas convergiram da forma esperada.

Gráfico 2 – Função de perda das redes neurais atreladas aos agentes decisores

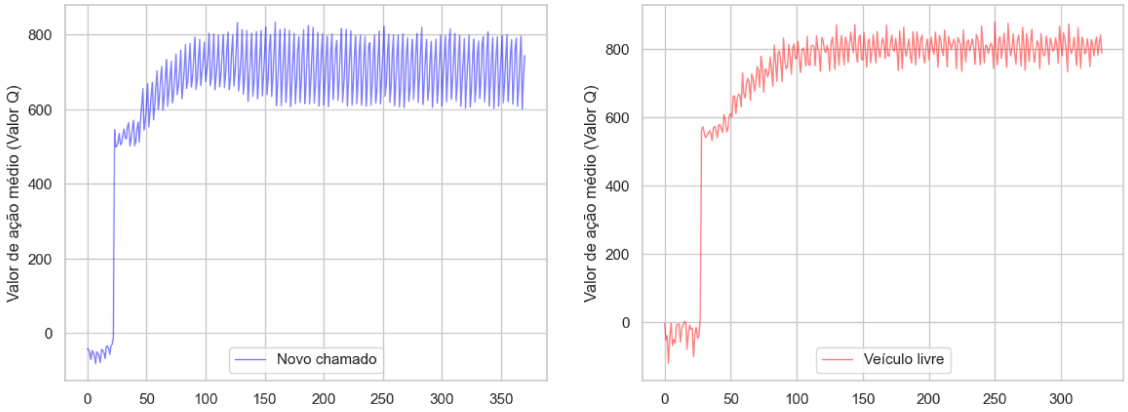


Fonte: Elaborada pelo autor.

O Gráfico 3 apresenta os valores-Q ao longo do treinamento das redes neurais. Conforme esperado, os valores aumentam ao longo do treinamento e se estabilizam após um período.

O Gráfico 4 apresenta as recompensas obtidas ao longo do treinamento das redes neurais. Assim como os valores-Q, espera-se que os valores cresçam e se estabilizem após um

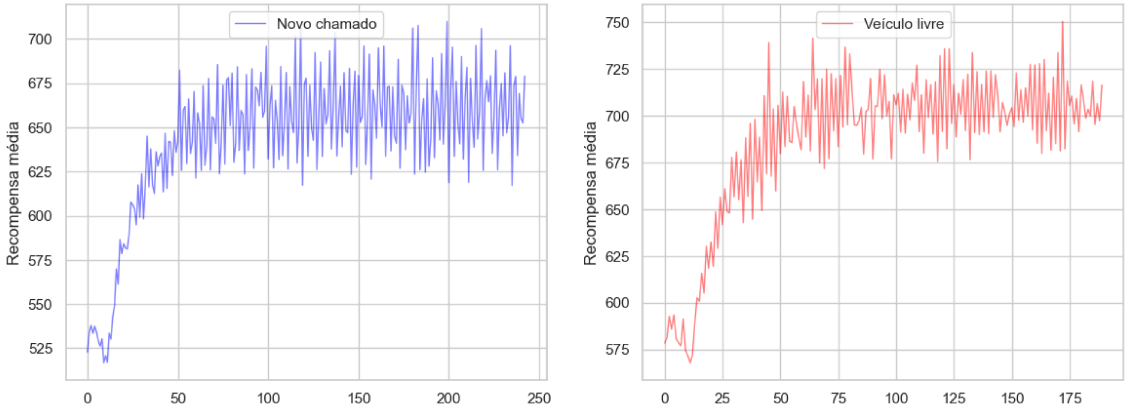
Gráfico 3 – Valores-Q ao longo do processo de treinamento



Fonte: Elaborada pelo autor.

determinado período.

Gráfico 4 – Recompensas ao longo do processo de treinamento



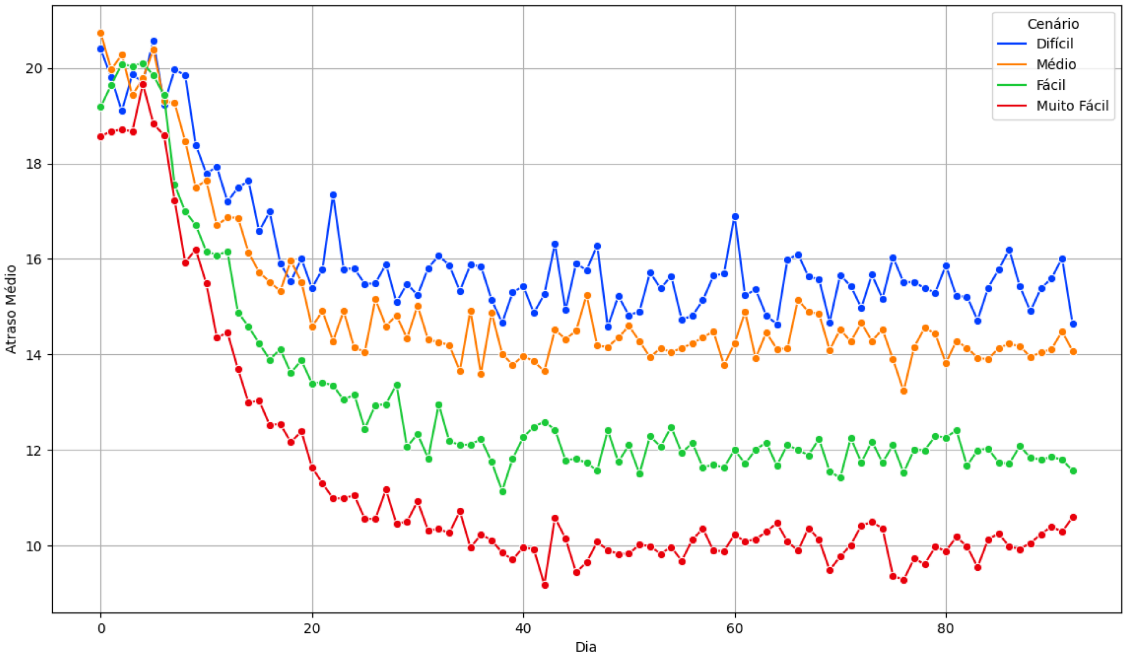
Fonte: Elaborada pelo autor.

Ressalta-se que os resultados apresentados nos Gráficos 2, 3 e 4 foram computados considerando a média dos valores a cada 1.000 decisões realizadas pelos agentes.

No Gráfico 5 é apresentado o atraso médio diário para cada dia do período de treinamento. A sua redução ao longo do período de treinamento demonstra que o treinamento foi efetivo em relação ao objetivo final dos agentes tomadores de decisão, que é reduzir o atraso médio dos clientes.

O processo de treinamento foi realizado 10 vezes com sementes distintas e a duração teve uma média de 34,63 minutes com desvio padrão de 1,05 minutos. Todo o processo de treinamento foi realizado em um computador equipado com um processado Intel Core i9 13900k, 64 GB de memória RAM a uma frequência de 5600 MHz e uma placa de video NVidia RTX 4090.

Gráfico 5 – Atraso médio ao longo do período de treinamento



Fonte: Elaborada pelo autor.

5.2 Resultados obtidos

Os resultados apresentados adiante foram obtidos considerando os chamados realizados no bairro do Brooklyn que ocorreram durante o mês de fevereiro de 2022. Para fins de análise dos resultados obtidos a abordagem desenvolvida neste trabalho será comparada com quatro heurística clássicas aplicáveis a este problema. O objetivo desta comparação é verificar se o custo associado à implementação de uma solução mais sofisticada, neste caso utilizando aprendizado por reforço profundo, pode valer a pena para as plataformas de mobilidade urbana em comparação a heurísticas mais simples.

As heurísticas utilizadas para realizar a comparação com os resultados obtidos neste trabalho foram:

a) **Vizinho mais próximo (NN; *nearest neighbor*)** - Nessa heurística considera-se a distância entre o chamado e o veículo. A alocação que tiver a menor distância é a que será sugerida pelo sistema;
b) **Primeiro a entrar, primeiro a sair (FIFO; *first in, first out*)** - Nessa heurística a ordem de entrada no *pool* de chamados em espera é considerada. O chamado que estiver há mais tempo aguardando, ou seja, tiver chegado primeiro ao sistema, é o que será sugerido para ser alocado ao veículo;

c) **Último a entrar, primeiro a sair (LIFO; *last in, first out*)** - Nessa heurística, assim como na FIFO, a ordem de entrada no *pool* de chamados em espera é considerada. Nesse caso o último chamado que tiver chegado, isto é, o que está aguardando há menos tempo, é o chamado sugerido pelo sistema;
d) **Aleatório (RANDOM)** - No caso dessa política, qualquer chamado e qualquer veículo podem ser escolhidos a qualquer momento, pois a decisão é completamente aleatória.

A políticas foram comparadas por meio de três métricas a seguir:

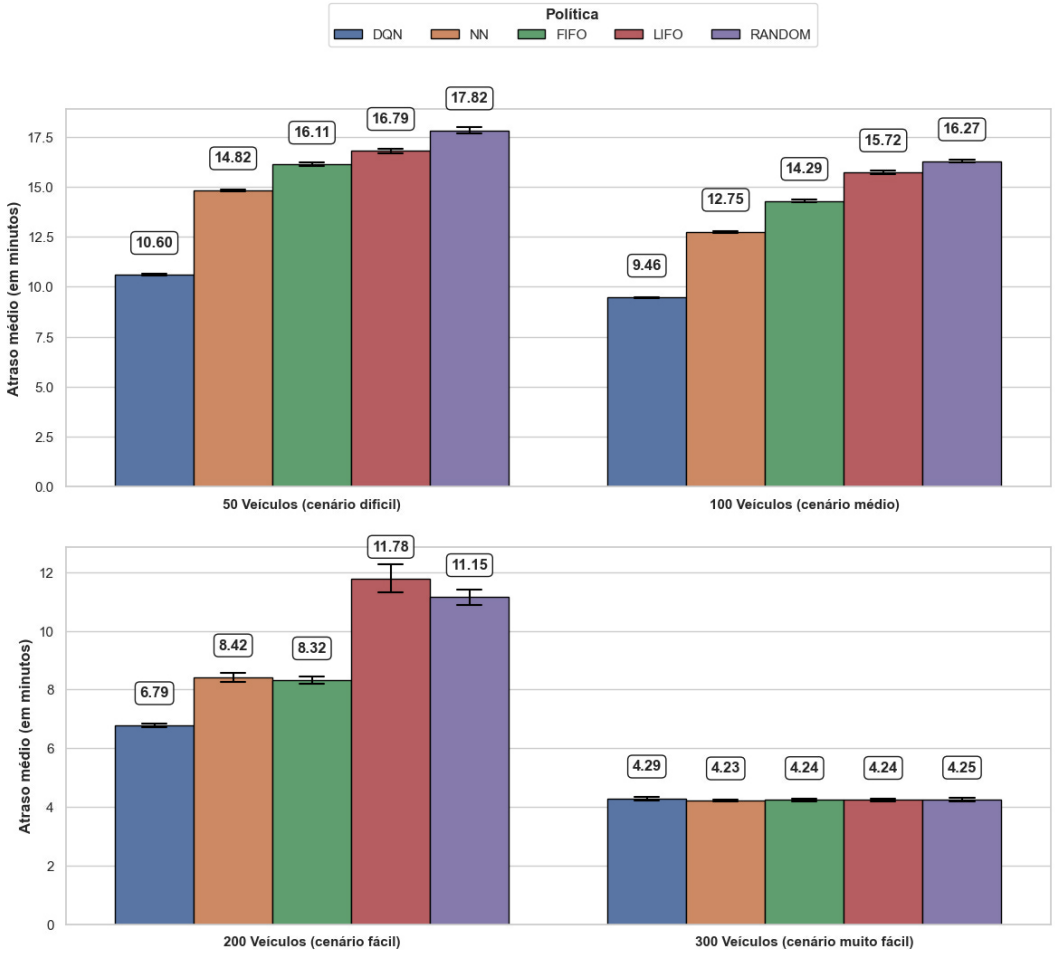
a) **Atraso médio** - Média de tempo que os veículos levam para chegar até a origem do chamado. Esse tempo é mensurado do momento que o chamado é realizado até o veículo chegar na sua origem.
b) **Taxa de cancelamento** - Razão entre chamadas canceladas e o total de chamados recebidos no dia.
c) **Tempo total de serviço** - Soma total dos períodos percorridos considerando o percurso da origem até o destino dos chamados.

Os resultados obtidos serão apresentados nas subseções a seguir, considerando dois cenários distintos: no primeiro, foi imposto um limite de 10.000 chamados diários, enquanto no segundo cenário o limite foi estabelecido em 100.000 chamadas diários. Cabe ressaltar que, durante o período de tempo correspondente aos dados utilizados no estudo, a demanda diária da plataforma Uber permaneceu abaixo do limite de 100.000 chamados, tornando o segundo cenário mais representativo da realidade. Ressalta-se também que para todos os gráficos de barras adiante foi utilizado o desvio padrão para avaliar a dispersão dos dados.

5.2.1 Cenário com 10.000 chamados diários

No primeiro cenário de teste, considerando uma limitação de 10.000 chamados diários, a abordagem proposta se mostrou eficiente em todos os cenários, tendo empatado com as demais apenas no cenário muito fácil, em que todas empataram. Isso se dá devido ao excesso de veículos disponíveis para atender a demanda, minimizando assim a necessidade de uma estratégia inteligente para manter baixos níveis de atraso. No Gráfico 6 é possível notar uma

Gráfico 6 – Média de tempo de atraso (10.000 chamados por dia)



Fonte: Elaborada pelo autor.

redução média de 4,23 minutos no cenário difícil, 3,29 minutos no cenário médio e 1,53 minutos no cenário fácil em relação à segunda melhor política de cada cenário. No cenário muito fácil houve um incremento de um valor irrisório de 0,06 segundos de atraso.

Tabela 1 – Resumo do atraso médio por quantidade de veículos e política (10.000 chamados por dia)

Quantidade de Veículos	DQN	NN	FIFO	LIFO	RANDOM	DQN - MIN *	DQN - MAX **
50 Veículos (dificil)	10.60	14.82	16.11	16.79	17.82	-4.23	-7.22
100 Veículos (médio)	9.46	12.75	14.29	15.72	16.27	-3.29	-6.81
200 Veículos (fácil)	6.79	8.42	8.32	11.78	11.15	-1.53	-4.99
300 Veículos (muito fácil)	4.29	4.23	4.24	4.24	4.25	0.06	0.04

* Atraso da DQN subtraído pelo menor atraso entre as demais políticas

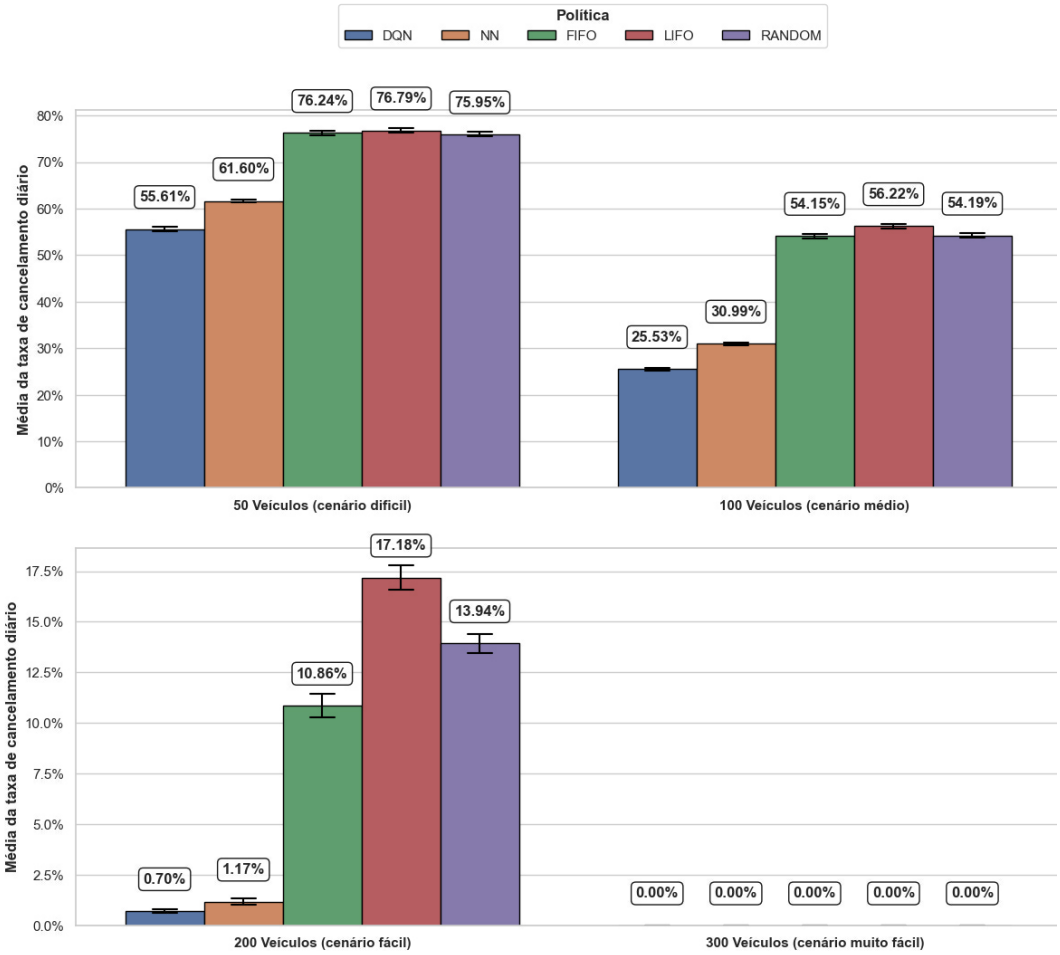
** Atraso da DQN subtraído pelo maior atraso entre as demais políticas

Fonte: Elaborada pelo autor.

Em relação à média diária da taxa de cancelamentos, a DQN também obteve os

melhores resultados nos cenários “Difícil” e “Médio”, reduzindo, respectivamente, aproximadamente 6% e 5% dos cancelamentos. No cenário “Fácil” a DQN também superou todas as demais políticas. Contudo, a taxa de cancelamentos foi muito próxima da NN, acarretando em um empate técnico. No cenário “Muito fácil” houve empate técnico por parte de todas as políticas.

Gráfico 7 – Média da taxa de cancelamento diária (10.000 chamados por dia)



Fonte: Elaborada pelo autor.

Tabela 2 – Resumo da média de taxa de cancelamento diária por quantidade de veículos e política (10.000 chamados por dia)

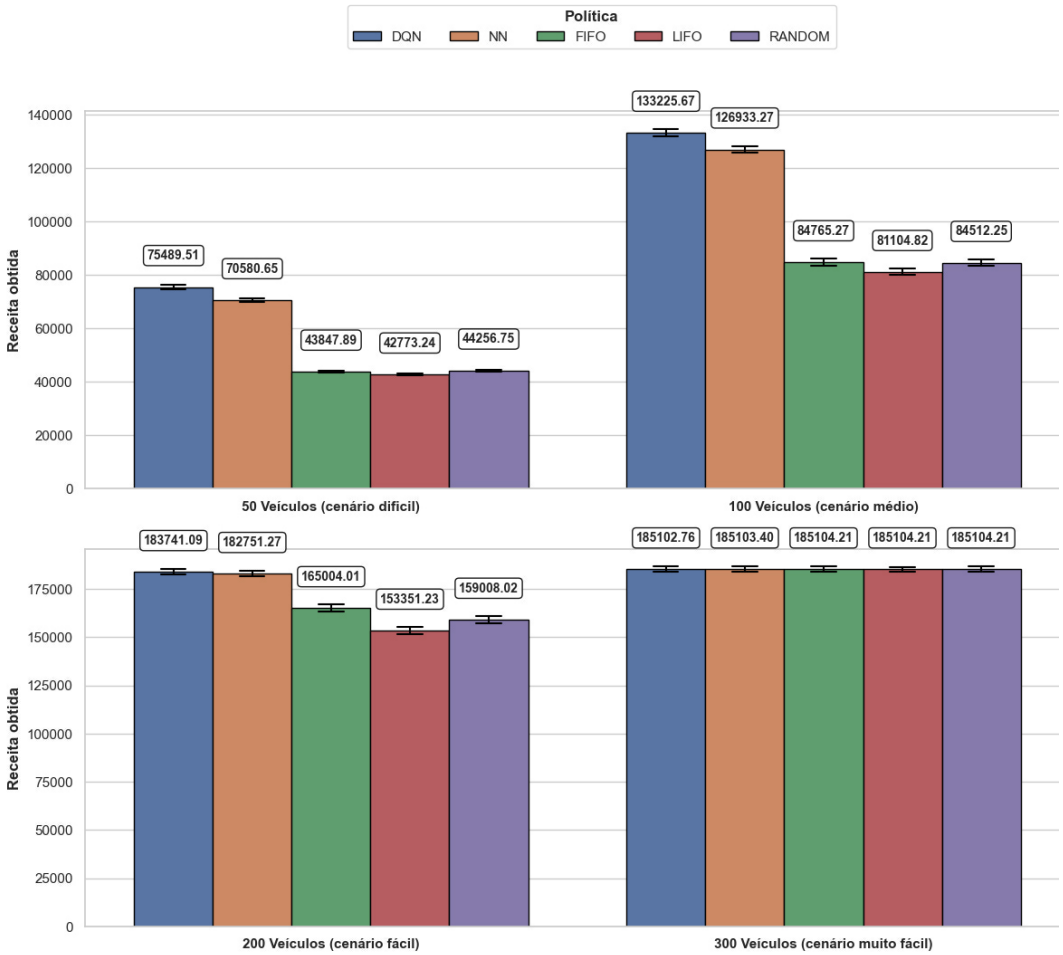
Quantidade de Veículos	DQN	NN	FIFO	LIFO	RANDOM	DQN - MIN *	DQN - MAX **
50 Veículos (difícil)	0.56	0.62	0.76	0.77	0.76	-0.06	-0.21
100 Veículos (médio)	0.26	0.31	0.54	0.56	0.54	-0.05	-0.31
200 Veículos (fácil)	0.01	0.01	0.11	0.17	0.14	-0.00	-0.16
300 Veículos (muito fácil)	0.00	0.00	0.00	0.00	0.00	0.00	0.00

* Taxa de cancelamento da DQN subtraída pela menor taxa de cancelamento entre as demais políticas

** Taxa de cancelamento da DQN subtraído pelo maior taxa de cancelamento entre as demais políticas

Fonte: Elaborada pelo autor.

Gráfico 8 – Média da receita diária (10.000 chamados por dia)



Fonte: Elaborada pelo autor.

Com 10.000 chamados diários a DQN superou todas as políticas quanto à receita obtida nos cenários “Difícil” e “Fácil”. No cenário “Fácil” houve empate técnico entre a DQN e a NN e no cenário “Muito fácil” houve empate técnico entre todas as políticas.

Tabela 3 – Resumo da média da receita diária por quantidade de veículos e política (10.000 chamados por dia)

Quantidade de Veículos	DQN	NN	FIFO	LIFO	RANDOM	DQN - MIN *	DQN - MAX **
50 Veículos (dificil)	75.49	70.58	43.85	42.77	44.26	32.72	4.91
100 Veículos (médio)	133.23	126.93	84.77	81.10	84.51	52.12	6.29
200 Veículos (fácil)	183.74	182.75	165.00	153.35	159.01	30.39	0.99
300 Veículos (muito fácil)	185.10	185.10	185.10	185.10	185.10	-0.00	-0.00

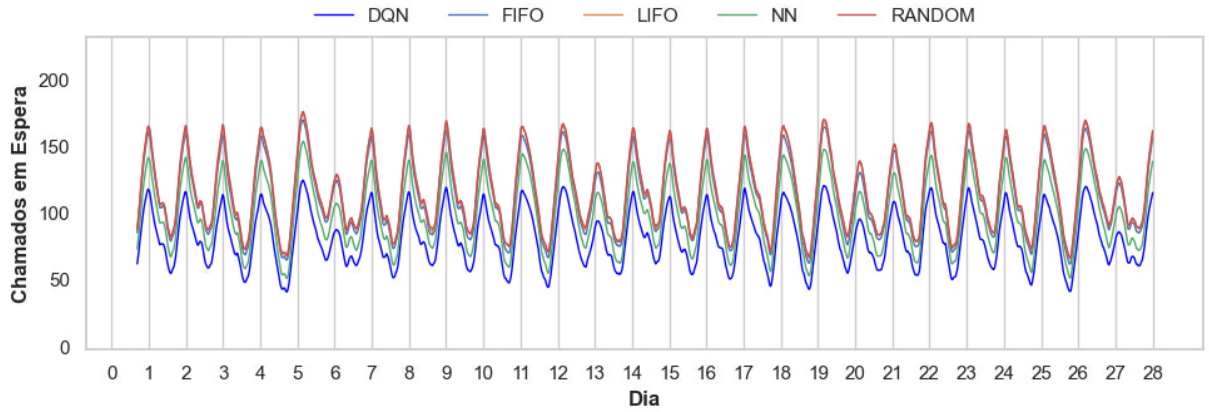
* Receita da DQN subtraída pela menor receita entre as demais políticas

** Receita da DQN subtraída pela maior receita entre as demais políticas

*** Valores em milhares

Fonte: Elaborada pelo autor.

Gráfico 9 – Monitoramento da quantidade de chamados em espera por minuto (10.000 chamados por dia no cenário médio)



* Valores suavizados pela média dos últimos 1000 minutos
Fonte: Elaborada pelo autor.

No Gráfico 9 é mostrado o monitoramento da quantidade de clientes aguardando a cada minuto do período estudado. Em todos os momentos do período estudado a DQN manteve a menor quantidade clientes aguardando uma alocação. Os resultados desse monitoramento para os outros cenários estão disponíveis no Apêndice A.

Na Figura 28 é possível verificar a distribuição dos chamados e veículos às 19 horas do dia primeiro de fevereiro de 2022, utilizando a política assumida pela DQN.

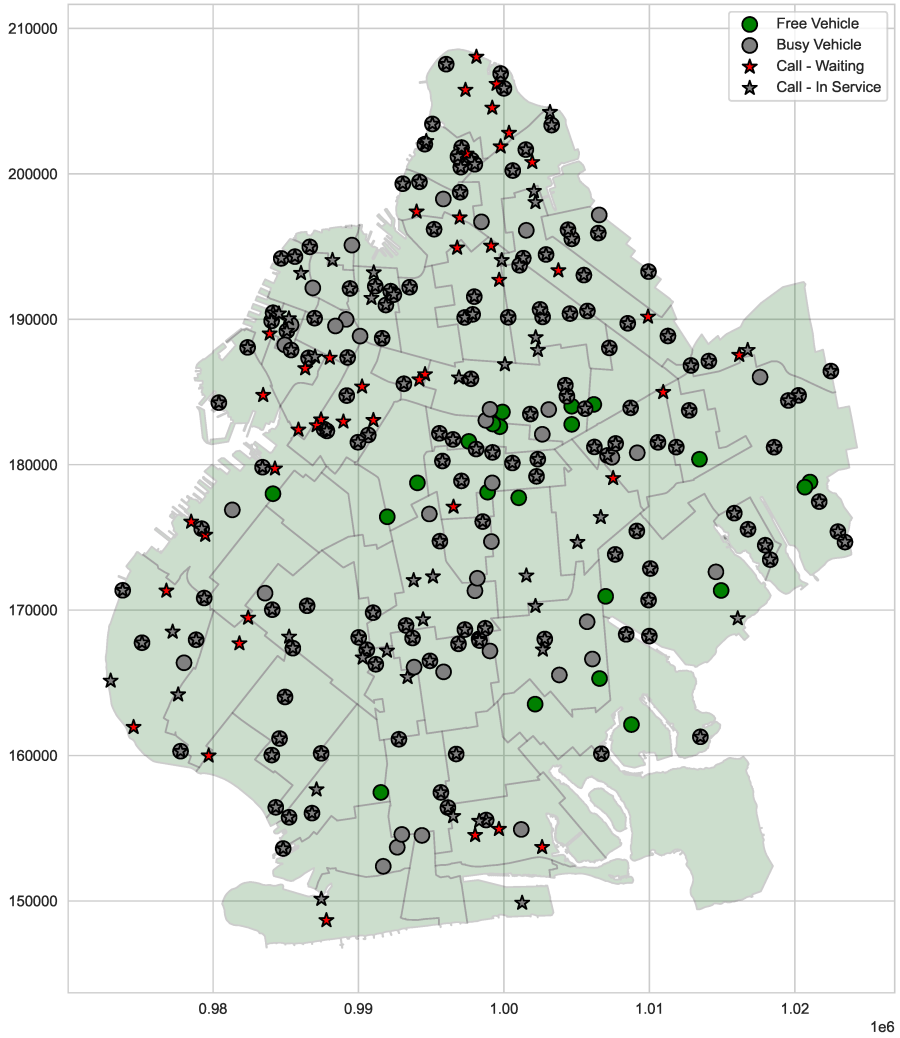
Como a abordagem proposta envolve um maior processamento de dados devido à necessidade de computar as funções de valor de ação, foi realizado o monitoramento do tempo total diário em que cada uma das políticas testadas leva para tomar as decisões. Os resultados obtidos estão apresentados na Tabela 4. A DQN, por ser uma técnica mais sofisticada, aumentou o tempo de processamento conforme o esperado, mas todos os tempos obtidos foram baixos, tendo em vista que representam a média de tempo para processar os chamados de um dia inteiro.

Tabela 4 – Média de tempo de processamento (10.000 chamados por dia)

Quantidade de Veículos	DQN	FIFO	LIFO	NN	RANDOM
50 Veículos (dificil)	0.38	0.20	0.20	0.22	0.21
100 Veículos (médio)	0.42	0.23	0.23	0.25	0.23
200 Veículos (fácil)	0.44	0.28	0.27	0.27	0.28
300 Veículos (muito fácil)	0.39	0.31	0.31	0.31	0.31

* Valores em minutos
Fonte: Elaborada pelo autor.

Figura 28 – Distribuição dos chamados e veículos no Brooklyn às 19 horas utilizando a política assumida pela DQN (10.000 chamados por dia)



Fonte: Elaborada pelo autor.

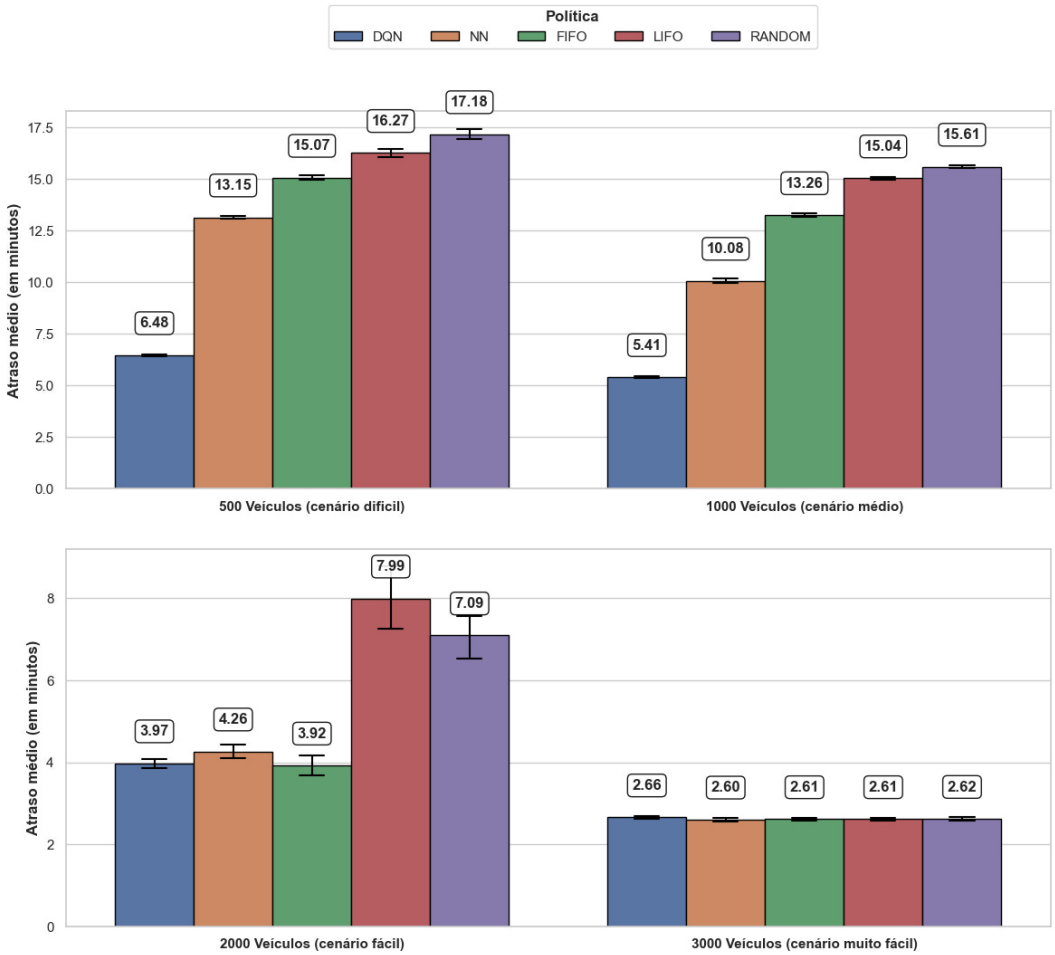
5.2.2 Cenário com 100.000 chamados diários

Neste cenário, se tem uma dimensão bastante realista do resultados esperados, pois a demanda diária de chamados nos dados utilizados, correspondente ao bairro Brooklyn no período estudado, ultrapassa a demanda de uma plataforma de mobilidade de larga escala como a Uber.

Por meio do Gráfico 10 é fácil ver que a DQN é superior às demais políticas nos cenários “Difícil” e “Médio”. Nestes cenários a DQN reduziu para aproximadamente a metade do atraso da segunda melhor política. No cenário “Fácil” houve empate técnico com as políticas FIFO e NN. Quanto ao cenário “Muito fácil” considera-se um empate técnico entre todas as políticas.

Referente à taxa de cancelamentos destacam-se as políticas DQN e NN, tendo sido

Gráfico 10 – Média de tempo de atraso (100.000 chamados por dia)



Fonte: Elaborada pelo autor.

Tabela 5 – Resumo do atraso médio por quantidade de veículos e política (100.000 chamados por dia)

Quantidade de Veículos	DQN	NN	FIFO	LIFO	RANDOM	DQN - MIN *	DQN - MAX **
500 Veículos (difícil)	6.48	13.15	15.07	16.27	17.18	-6.67	-10.70
1000 Veículos (médio)	5.41	10.08	13.26	15.04	15.61	-4.67	-10.20
2000 Veículos (fácil)	3.97	4.26	3.92	7.99	7.09	0.05	-4.02
3000 Veículos (muito fácil)	2.66	2.60	2.61	2.61	2.62	0.05	0.04

* Atraso da DQN subtraído pelo menor atraso entre as demais políticas

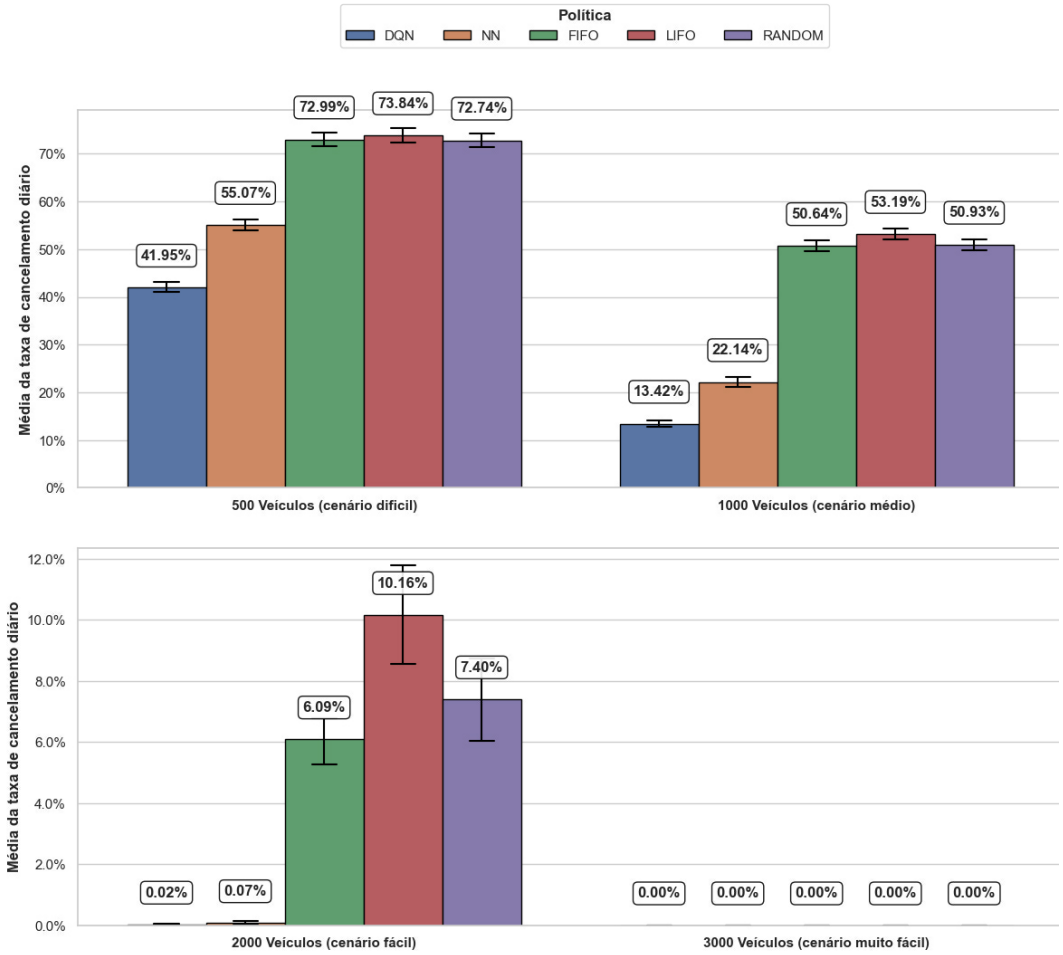
** Atraso da DQN subtraído pelo maior atraso entre as demais políticas

Fonte: Elaborada pelo autor.

a primeira melhor nos cenários “Difícil” e “Médio” e a havendo um empate entre as duas no cenário “Fácil”. Apesar da DQN e NN se destacarem das demais políticas, a DQN ainda obteve uma redução de aproximadamente 13% e 9% nos cenários “Difícil” e “Médio”, quando comparada com a NN. No cenário “Muito fácil” notou-se que qualquer que seja a política, a frequência de cancelamentos é praticamente nula, isso porque há um excesso de veículos em

relação à demanda.

Gráfico 11 – Média da taxa de cancelamento diária (100.000 chamados por dia)



Fonte: Elaborada pelo autor.

Tabela 6 – Resumo da média de taxa de cancelamento diária por quantidade de veículos e política (100.000 chamados por dia)

Quantidade de Veículos	DQN	NN	FIFO	LIFO	RANDOM	DQN - MIN *	DQN - MAX **
500 Veículos (difícil)	0.42	0.55	0.73	0.74	0.73	-0.13	-0.32
1000 Veículos (médio)	0.13	0.22	0.51	0.53	0.51	-0.09	-0.40
2000 Veículos (fácil)	0.00	0.00	0.06	0.10	0.07	-0.00	-0.10
3000 Veículos (muito fácil)	0.00	0.00	0.00	0.00	0.00	0.00	0.00

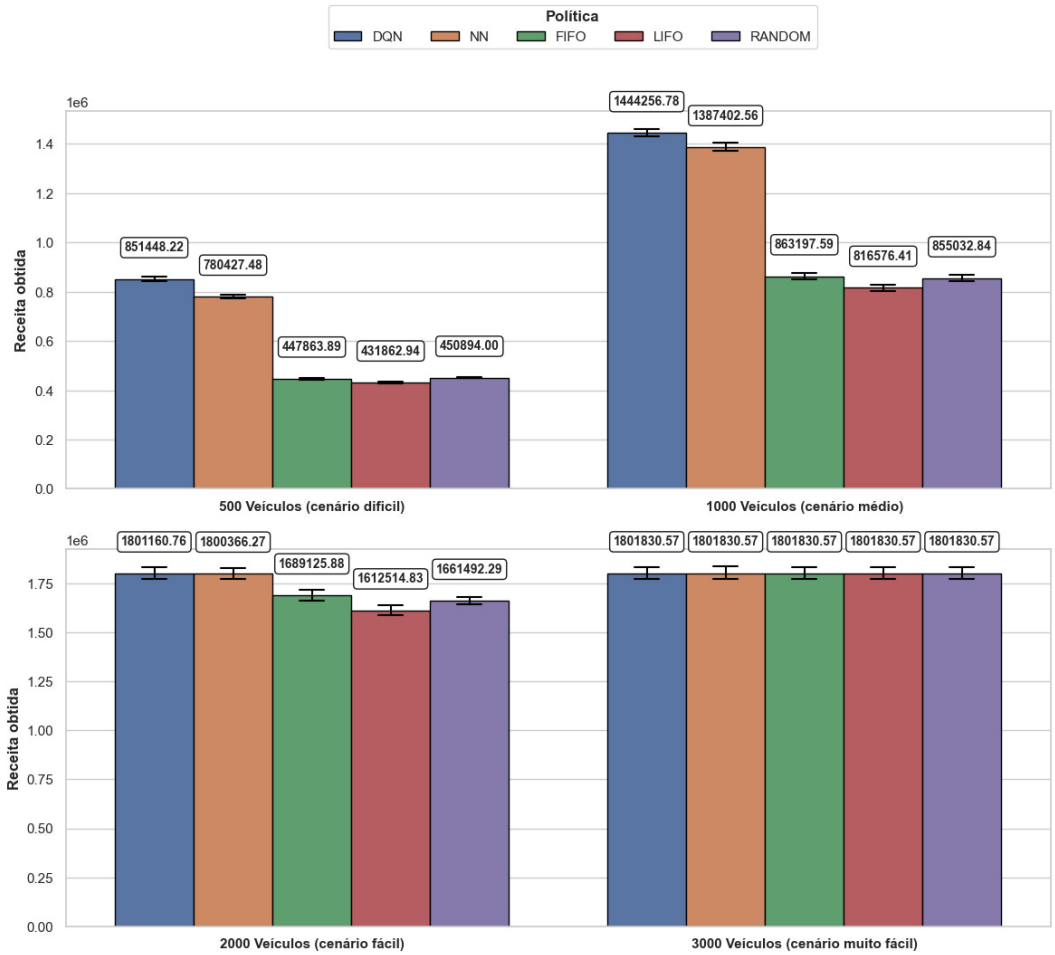
* Taxa de cancelamento da DQN subtraída pela menor taxa de cancelamento entre as demais políticas

** Taxa de cancelamento da DQN subtraída pela maior taxa de cancelamento entre as demais políticas

Fonte: Elaborada pelo autor.

Quanto à receita obtida, assim como para as demais métricas, observa-se uma superioridade da DQN nos cenários “Difícil” e “Médio”, tendo tido um acréscimo de aproximadamente 71.02 mil e 56.85 mil respectivamente. No cenário “Fácil” considera-se um empate técnico

Gráfico 12 – Média da receita diária (100.000 chamados por dia)



Fonte: Elaborada pelo autor.

Tabela 7 – Resumo da média da receita diária por quantidade de veículos e política (100.000 chamados por dia)

Quantidade de Veículos	DQN	NN	FIFO	LIFO	RANDOM	DQN - MIN *	DQN - MAX **
500 Veículos (difícil)	0.85	0.78	0.45	0.43	0.45	0.42	0.07
1000 Veículos (médio)	1.44	1.39	0.86	0.82	0.86	0.63	0.06
2000 Veículos (fácil)	1.80	1.80	1.69	1.61	1.66	0.19	0.00
3000 Veículos (muito fácil)	1.80	1.80	1.80	1.80	1.80	0.00	0.00

* Receita da DQN subtraída pela menor receita entre as demais políticas

** Receita da DQN subtraída pela maior receita entre as demais políticas

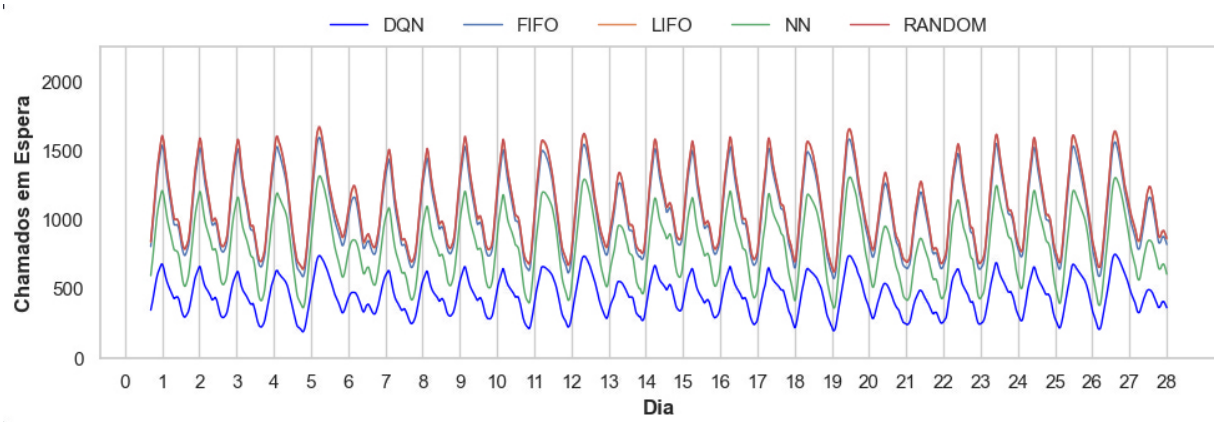
*** Valores em milhões

Fonte: Elaborada pelo autor.

entre a DQN e a NN e no cenário “Muito fácil” considera-se um empate técnico entre todas as políticas testadas.

No Gráfico 15, que apresenta a quantidade de chamados em espera em cada minuto dos dias do mês de fevereiro considerando o cenário “Médio”, pode-se perceber que a DQN

Gráfico 13 – Monitoramento da quantidade de chamados em espera por minuto (100.000 chamados por dia)



Fonte: Elaborada pelo autor.

Tabela 8 – Média de tempo de processamento (100.000 chamados por dia)

Quantidade de Veículos	DQN	FIFO	LIFO	NN	RANDOM
500 Veículos (dificil)	6.68	4.03	3.97	5.37	4.05
1000 Veículos (médio)	8.72	6.16	6.02	7.43	6.18
2000 Veículos (fácil)	12.33	10.05	9.57	9.28	9.58
3000 Veículos (muito fácil)	13.75	15.29	15.38	14.87	15.23

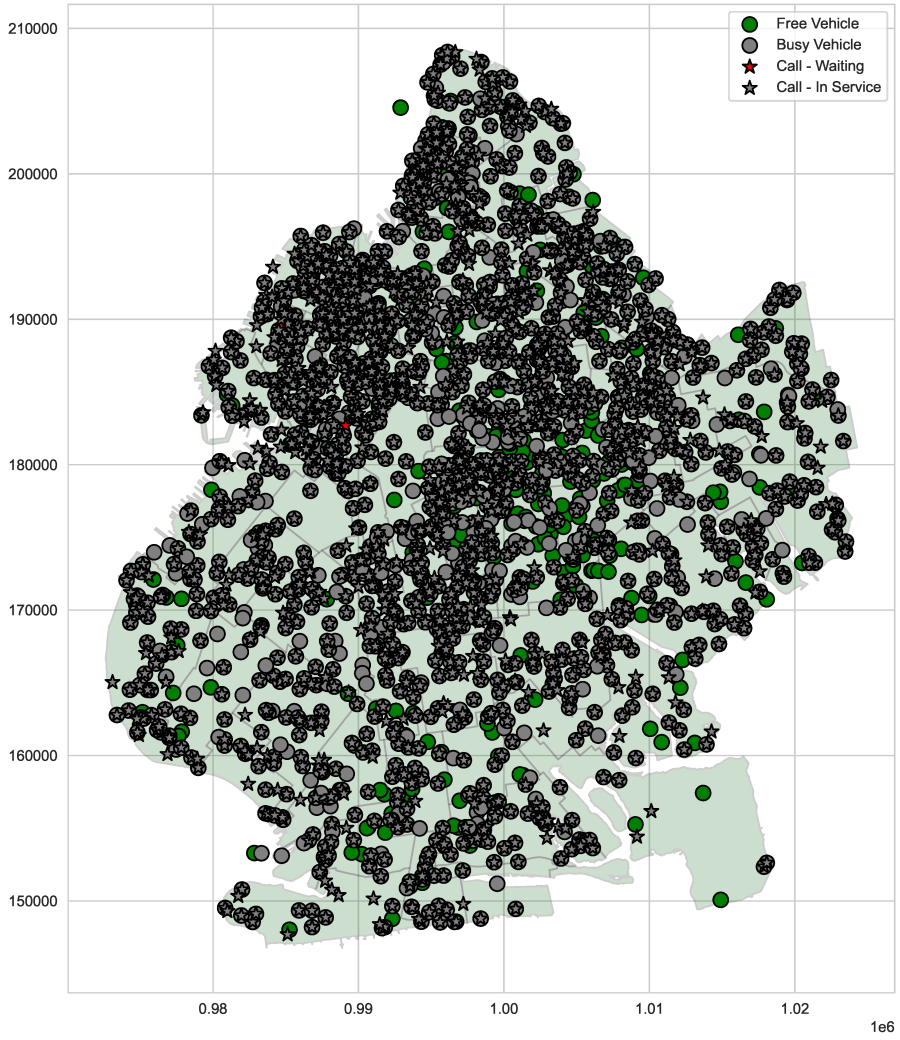
Fonte: Elaborada pelo autor.

permanece com menores quantidade de clientes aguardando durante todo o período, tendo inclusive uma dispersão menor. O resultados dos demais cenários estão disponíveis no Apêndice A.

Com a quantidade de veículos considerada neste cenário de teste, tem-se um cenário muito próximo do real, sendo portanto o cenário mais relevante para avaliação dos resultados. Na Figura 29 é apresentada a distribuição dos veículos e chamados em um horário de pico de solicitações no dia primeiro de fevereiro de 2022 utilizando a política assumida pela DQN. Nesta figura é possível perceber que a distribuição espacial dos chamados se dá com maior densidade na região norte do Brooklyn.

A Tabela 8 mostra os tempos médios de processamento para a realização dos testes. Como esperado, a DQN é a política que possui o maior tempo de processamento médio por dia. Isso é natural devido à complexidade associada à técnica, que envolve mais cálculos para a tomada de decisão. Apesar disso, os tempos não inviabilizam seu uso em ambiente de produção, uma vez que o tempo que a abordagem proposta demora para tomar todas as decisões necessárias de um dia inteiro com 100.000 chamados teve uma média menor que uma hora.

Figura 29 – Distribuição dos chamados e veículos no Brooklyn às 19 horas utilizando a política assumida pela DQN (100.000 chamados por dia)



Fonte: Elaborada pelo autor.

6 CONCLUSÃO

O trabalho desenvolvido nesta dissertação buscou realizar uma modelagem para o problema de alocação de veículos dinâmico e estocástico por uma via ainda não explorada na literatura. A abordagem baseada em eventos, utilizando como arcabouço teórico uma modelagem fundamentada em processos de decisão semimarkovianos, demonstrou-se adequada para o problema, suportando representar especificidades que são frequentemente ignoradas em outras abordagens, além de ter proporcionado uma redução da complexidade dos problemas de alocação envolvidos.

O ambiente de simulação, que fez a utilização de dados reais de corridas de táxi ocorridas em Nova York, foi implementado considerando atributos da realidade que raramente são encontrados em conjunto em trabalhos dessa natureza, como a possibilidade de rejeição da alocação por parte do motorista e a tolerância máxima de espera dos clientes. A configuração utilizada para a representação dos dados que foram apresentados para os agentes decisores possibilitou que a quantidade de ações possíveis a cada época de decisão seja variável. Além disso, os agentes decisores associados ao ambiente foram implementados de forma satisfatória, interpretando as representações de estado de maneira correta e provendo ações visando a minimização do tempo médio de espera dos clientes não só de forma imediata, mas considerando também o contexto futuro.

Os resultados obtidos nos experimentos evidenciaram que a modelagem proposta é superior às heurísticas testadas, sobretudo em cenários de alta demanda, que são aqueles em que a definição de estratégias inteligentes podem impactar os resultados. Em termos de espera dos clientes, a proposta do trabalho foi superior em todos os cenários apresentados, exceto naqueles em que há excesso de recursos. Contudo, nesses casos foi percebido que é dispensável a definição de uma estratégia inteligente, umas vez que ocorreu empate técnico entre todas, inclusive a aleatória.

Quanto às taxas de cancelamento de chamados e às receitas obtidas, a proposta do trabalho também se mostrou satisfatória em todos os cenários, apresentando resultados melhores que as demais políticas nos cenários difíceis e médios. Já nos demais cenários, no pior dos casos houve empate técnico com alguma das demais políticas, o que demonstra que a estratégia adotada para a redução do atraso médio é adequada para prover uma melhor satisfação ao usuário do serviço e manter ou aumentar a receita da operadora do serviço. Além disso, também foi evidenciado que no período estudado a proposta do trabalho manteve as quantidades mais baixas

de clientes em espera.

Por fim, considera-se que todos os objetivos inicialmente propostos foram atingidos, bem como os resultados obtidos pela abordagem desenvolvida foram satisfatórios nos cenários testados.

Como trabalhos futuros e extensões, propõem-se:

a) Realizar a integração com serviços de informações de tráfego em tempo real para uma melhor mensuração dos tempos envolvidos nas corridas;
b) Aplicar a abordagem apresentada neste trabalho em outras localidades;
c) Estudar a possibilidade de transferência de aprendizado dos agentes para utilização em outras localidades;
d) Comparar, em um ambiente de simulação único, a abordagem apresentada neste trabalho com outras abordagens recentemente propostas na literatura (as quais normalmente são baseadas em *matching*);
e) Verificar se utilizar uma abordagem multi-agente pode melhorar os resultados em termos de atraso, taxa de cancelamento, receita ou performance de processamento.

REFERÊNCIAS

ALAGOZ, O.; HSU, H.; SCHAEFER, A. J.; ROBERTS, M. S. Markov decision processes: a tool for sequential decision making under uncertainty. **Medical Decision Making**, v. 30, n. 4, p. 474–483, jul. 2010. ISSN 0272-989X, 1552-681X.

ALONSO-MORA, J.; SAMARANAYAKE, S.; WALLAR, A.; FRAZZOLI, E.; RUS, D. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. **Proceedings of the National Academy of Sciences**, United States, v. 114, n. 3, p. 462–467, jan. 2017. ISSN 0027-8424, 1091-6490.

ARULKUMARAN, K.; DEISENROTH, M. P.; BRUNDAGE, M.; BHARATH, A. A. Deep reinforcement learning: a brief survey. **IEEE Signal Processing Magazine**, v. 34, n. 6, p. 26–38, nov. 2017. ISSN 1053-5888.

BARRETO, A. M. S.; AUGUSTO, D. A.; BARBOSA, H. J. C. On the characteristics of sequential decision problems and their impact on evolutionary computation and reinforcement learning. *In*: ARTIFICIAL EVOLUTION, 9., 2009, Strasbourg, France. **Proceedings [...]**. Switzerland: Springer. v. 5975, p. 194–205.

BELLMAN, R. **Dynamic Programming**. Princeton, NJ: Princeton Univ. Pr, 1957. ISBN 978-0-691-07951-6.

BERTSIMAS, D.; JAILLET, P.; MARTIN, S. Online vehicle routing: the edge of optimization in large-scale applications. **Operations Research**, United States, v. 67, n. 1, p. 143–162, 2019.

BRADTKE, S. J.; DUFF, M. O. Reinforcement learning methods for continuous-time markov decision problems. MIT Press, Cambridge, MA, USA, 1994.

BURKARD, R. E.; DELL'AMICO, M.; MARTELLO, S. **Assignment problems**. Philadelphia: SIAM, Society for Industrial and Applied Mathematics, 2009. ISBN 978-0-89871-663-4.

CATTRYSSE, D. G.; WASSENHOVE, L. N. V. A survey of algorithms for the generalized assignment problem. **European Journal of Operational Research**, Netherlands, v. 60, n. 3, p. 260–272, ago. 1992. ISSN 03772217.

CHENG, S.-F.; QU, X. A service choice model for optimizing taxi service delivery. *In*: INTERNATIONAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEM, 12., 2009, United States. **Proceedings [...]**. United States: IEEE, 2009. p. 1–6.

CLIFTON, J.; LABER, E. Q-learning: theory and applications. **Annual Review of Statistics and Its Application**, United States, v. 7, n. 1, p. 279–301, mar. 2020. ISSN 2326-8298.

CORDEAU, J. F.; DESAUNIERS, G.; DESROSIERS, J.; SOLOMON, M. M.; SOUMIS, F. The VRP with time windows. *In*: TOTH, P.; VIGO, D. (ed.). **The vehicle routing problem**. United States: Society for Industrial and Applied Mathematics, 2002. p. 157–193.

CORDEAU, J.-F.; LAPORTE, G. The dial-a-ride problem: models and algorithms. **Annals of Operations Research**, Netherlands, v. 153, n. 1, p. 29–46, jun. 2007. ISSN 0254-5330, 1572-9338.

CRITES, R. H.; BARTO, A. G. Elevator group control using multiple reinforcement learning agents. **Machine Learning**, Kluwer Academic Publishers, Netherlands, 1998.

DANTZIG, G. B.; RAMSER, J. H. The truck dispatching problem. **Management Science**, United States, v. 6, n. 1, p. 80–91, 1959.

DESAUNIERS, G.; DESROSIERS, J.; ERDMANN, A.; SOLOMON, M. M.; SOUMIS, F. VRP with pickup and delivery. *In*: TOTH, P.; VIGO, D. (ed.). **The vehicle routing problem**. United States: Society for Industrial and Applied Mathematics, 2002. p. 225–242.

DU, J.; FUTOMA, J.; DOSHI-VELEZ, F. Model-based reinforcement learning for semi-Markov decision processes with neural odes. *In*: INTERNATIONAL CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS, 34., 2020, United States. **Proceedings [...]**. United States: Curran Associates Inc., 2020.

FISHER, M. L.; JAIKUMAR, R.; WASSENHOVE, L. N. V. A multiplier adjustment method for the generalized assignment problem. **Management Science**, United States, v. 32, n. 9, p. 1095–1103, set. 1986. ISSN 0025-1909, 1526-5501.

GALLAGER, R. G. **Stochastic Processes**: theory for applications. United Kingdom: Cambridge University Press, 2013. ISBN 978-1-107-03975-9 978-1-139-62651-4.

GAO, G.; XIAO, M.; ZHAO, Z. Optimal multi-taxi dispatch for mobile taxi-hailing systems. *In*: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 45., 2016, United States. **Proceedings [...]**. United States: IEEE, 2016. p. 294–303.

GARCIA, F.; RACHELSON, E. Markov decision processes. *In*: SIGAUD, O.; BUFFET, O. (ed.). **Markov decision processes in artificial intelligence**. Hoboken, NJ USA: John Wiley & Sons, Inc., 2013. p. 1–38. ISBN 978-1-118-55742-6 978-1-84821-167-4.

GOETSCHALCKX, M.; JACOBS-BLECHA, C. The vehicle routing problem with backhauls. **European Journal of Operational Research**, Netherlands, v. 42, n. 1, p. 39–51, set. 1989. ISSN 03772217.

GOSAVI, A. **Simulation-based optimization**. United States: Springer, 2015.

HADLEY, G. **Non-linear and dynamic programming**. United States: Addison-Wesley Publishing Company, 1964.

HAUSMAN, W. H. Sequential decision problems: A model to exploit existing forecasters. **Management Science**, United States, v. 16, n. 2, p. B–93–B–111, out. 1969. ISSN 0025-1909, 1526-5501.

KUHN, H. W. The hungarian method for the assignment problem. **Naval Research Logistics Quarterly**, United States, v. 2, n. 1-2, p. 83–97, mar. 1955. ISSN 00281441, 19319193.

KULLMAN, N. D.; COUSINEAU, M.; GOODSON, J. C.; MENDOZA, J. E. Dynamic ride-hailing with electric vehicles. **Transportation Science**, United States, 2021.

KUMAR, S. N.; PANNEERSELVAM, R. A survey on the vehicle routing problem and its variants. **Intelligent Information Management**, United States, v. 04, n. 03, p. 66–74, 2012. ISSN 2160-5912, 2160-5920.

LEW, A.; MAUCH, H. **Dynamic programming**: a computational tool. Berlin: Springer, 2007. ISBN 978-3-540-37013-0.

LI, Y. **Deep reinforcement learning**. [*S. l.*]: arXiv, 2018.

LIANG, E.; WEN, K.; LAM, W. H. K.; SUMALEE, A.; ZHONG, R. An integrated reinforcement learning and centralized programming approach for online taxi dispatching. **IEEE Transactions on Neural Networks and Learning Systems**, United States, v. 33, n. 9, p. 4742–4756, set. 2022. ISSN 2162-237X, 2162-2388.

LIAO, Z. Taxi dispatching via global positioning systems. **IEEE Transactions on Engineering Management**, United States, v. 48, n. 3, p. 342–347, 2001. ISSN 00189391.

MAHOOR, M.; SALMASI, F. R.; NAJAFABADI, T. A. A hierarchical smart street lighting system with brute-force energy optimization. **IEEE Sensors Journal**, United States, v. 17, n. 9, p. 2871–2879, maio 2017. ISSN 1530-437X, 1558-1748, 2379-9153.

MENDONÇA, V. I. T.; MENESES, R. F.; Pitombeira-Neto, A. R. Formulação e solução do problema de alocação de veículos estocástico por meio de programação dinâmica aproximada. **Anais do Congresso Brasileiro de Inteligência Computacional**, Brasil, v. 15, p. 1–7, 2021.

MIAO, F.; HAN, S.; LIN, S.; STANKOVIC, J. A.; ZHANG, D.; MUNIR, S.; HUANG, H.; HE, T.; PAPPAS, G. J. Taxi dispatch with real-time sensing data in metropolitan areas:: a receding horizon control approach. **IEEE Transactions on Automation Science and Engineering**, United States, v. 13, n. 2, p. 463–478, abr. 2016. ISSN 1545-5955, 1558-3783.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. **Playing atari with deep reinforcement learning**. [*S. l.*]: arXiv, 2013.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLOU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. Human-level control through deep reinforcement learning. **Nature**, Nature, United Kingdom, v. 518, n. 7540, p. 529–533, fev. 2015. ISSN 0028-0836, 1476-4687.

MOLENBRUCH, Y.; BRAEKERS, K.; CARIS, A. Typology and literature review for dial-a-ride problems. **Annals of Operations Research**, Springer, Netherlands, v. 259, n. 1-2, p. 295–325, dez. 2017. ISSN 0254-5330, 1572-9338.

ÖNCAN, T. A survey of the generalized assignment problem and its applications. **Information Systems and Operational Research**, Taylor & Francis, Canada, v. 45, n. 3, p. 123–141, ago. 2007. ISSN 0315-5986, 1916-0615.

PENTICO, D. W. Assignment problems: a golden anniversary survey. **European Journal of Operational Research**, Elsevier, Netherlands, v. 176, n. 2, p. 774–793, jan. 2007. ISSN 03772217.

POWELL, W. B. **Reinforcement learning and stochastic optimization**: a unified framework for sequential decisions. Hoboken, New Jersey: John Wiley & Sons, Inc, 2022. ISBN 978-1-119-81503-7.

PUTERMAN, M. L. **Markov decision processes**: discrete stochastic dynamic programming. Hoboken, NJ: Wiley-Interscience, 2005. ISBN 978-0-471-72782-8.

QIN, Z. T.; TANG, X.; JIAO, Y.; ZHANG, F.; XU, Z.; ZHU, H.; YE, J. Ride-hailing order dispatching at didi via reinforcement learning. **INFORMS Journal on Applied Analytics**, INFORMS, United States, v. 50, n. 5, p. 272–286, set. 2020. ISSN 2644-0865, 2644-0873.

RABBANI, Q.; KHAN, A.; QUDDOOS, A. Modified hungarian method for unbalanced assignment problem with multiple jobs. **Applied Mathematics and Computation**, Elsevier, United States, v. 361, p. 493–498, nov. 2019. ISSN 00963003.

SEOW, K. T.; DANG, N. H.; LEE, D.-H. A collaborative multiagent taxi-dispatch system. **IEEE Transactions on Automation Science and Engineering**, IEEE, United States, v. 7, n. 3, p. 607–616, jul. 2010. ISSN 1545-5955.

SILVER, D.; HUANG, A.; MADDISON, C. J.; GUEZ, A.; SIFRE, L.; van den Driessche, G.; SCHRITTWIESER, J.; ANTONOGLOU, I.; PANNEERSHELVAM, V.; LANCTOT, M.; DIELEMAN, S.; GREWE, D.; NHAM, J.; KALCHBRENNER, N.; SUTSKEVER, I.; LILLICRAP, T.; LEACH, M.; KAVUKCUOGLU, K.; GRAEPEL, T.; HASSABIS, D. Mastering the game of Go with deep neural networks and tree search. **Nature**, Nature, United Kingdom, v. 529, n. 7587, p. 484–489, jan. 2016. ISSN 0028-0836, 1476-4687.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning**: an introduction. Second edition. Cambridge, Massachusetts: The MIT Press, 2018. ISBN 978-0-262-03924-6.

SUTTON, R. S.; PRECUP, D.; SINGH, S. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. **Artificial Intelligence**, Elsevier, Netherlands, v. 112, n. 1-2, p. 181–211, ago. 1999. ISSN 00043702.

TANG, X.; QIN, Z. T.; ZHANG, F.; WANG, Z.; XU, Z.; MA, Y.; ZHU, H.; YE, J. A deep value-network based approach for multi-driver order dispatching. *In*: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY & DATA MINING, 25., 2019, United States. **Proceedings [...]**. United States: ACM, 2019. p. 1780–1790. ISBN 978-1-4503-6201-6.

TOTH, P.; VIGO, D. An overview of vehicle routing problems. *In*: TOTH, P.; VIGO, D. (ed.). **The vehicle routing problem**. United States: Society for Industrial and Applied Mathematics, 2002. p. 1–26. ISBN 978-0-89871-498-2 978-0-89871-851-5.

TOTH, P.; VIGO, D. VRP with backhauls. *In*: TOTH, P.; VIGO, D. (ed.). **The vehicle routing problem**. United States: Society for Industrial and Applied Mathematics, 2002. p. 195–224. ISBN 978-0-89871-498-2 978-0-89871-851-5.

VAN-HASSELT, H.; GUEZ, A.; SILVER, D. **Deep reinforcement learning with double Q-Learning**. [*S. l.*]: arXiv, 2015.

VOTAW, D. F.; ORDEN, A. Personnel assignment problem. *In*: SYMPOSIUM ON LINEAR INEQUALITIES AND PROGRAMMING, 1951, United States. **Proceedings [...]**. United States: Planning Research Division, 1952.

WANG, H.; CHEU, R.; LEE, D.-H. Intelligent taxi dispatch system for advance reservations. **Journal of Public Transportation**, Elsevier, United States, v. 17, n. 3, p. 115–128, set. 2014. ISSN 1077-291X, 2375-0901.

WATKINS, C. J. C. H. **Learning from delayed rewards**. Tese (Doutorado) – King's College, United Kingdom, 1989.

WEN, J.; ZHAO, J.; JAILLET, P. Rebalancing shared mobility-on-demand systems: a reinforcement learning approach. *In*: IEEE INTERNATIONAL CONFERENCE ON INTELLIGENT TRANSPORTATION SYSTEMS (ITSC), 20., 2017, Japan. **Proceedings [...]**. Japan: IEEE, 2017. p. 220–225. ISBN 978-1-5386-1526-3.

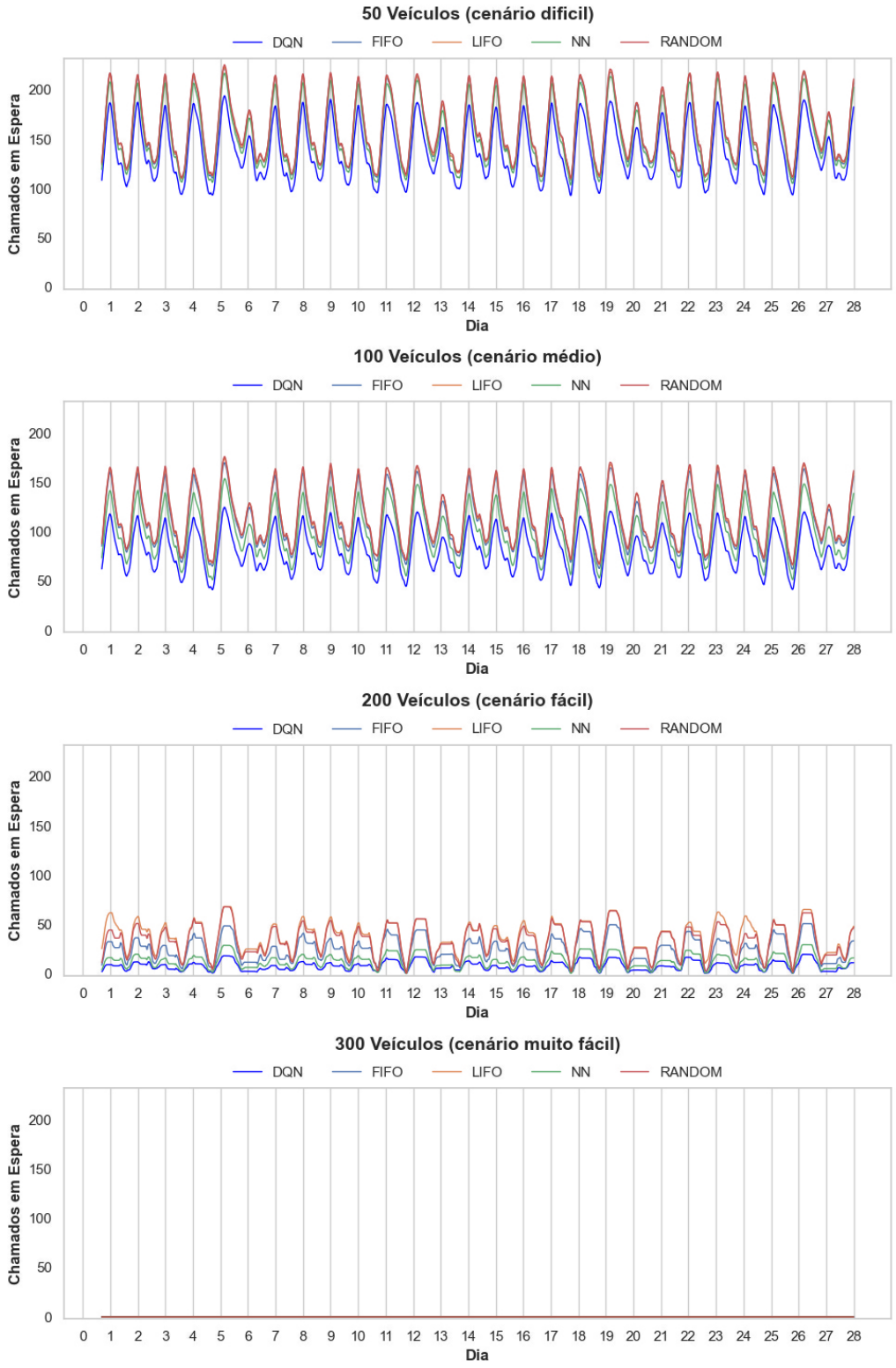
XIAO, Y.; ZHAO, Q.; KAKU, I.; XU, Y. Development of a fuel consumption optimization model for the capacitated vehicle routing problem. **Computers & Operations Research**, Elsevier, United Kingdom, v. 39, n. 7, p. 1419–1431, jul. 2012. ISSN 03050548.

XU, Z.; LI, Z.; GUAN, Q.; ZHANG, D.; LI, Q.; NAN, J.; LIU, C.; BIAN, W.; YE, J. Large-scale order dispatch in on-demand ride-hailing platforms: a learning and planning approach. *In*: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 24., 2018, United Kingdom. **Proceedings [...]**. United Kingdom: ACM, 2018. p. 905–913.

ZHANG, L.; HU, T.; MIN, Y.; WU, G.; ZHANG, J.; FENG, P.; GONG, P.; YE, J. A taxi order dispatch model based on combinatorial optimization. *In*: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 23., 2017, Canada. **Proceedings [...]**. Canada: ACM, 2017. p. 2151–2159.

APÊNDICE A – CHAMADOS EM ESPERA POR MINUTO

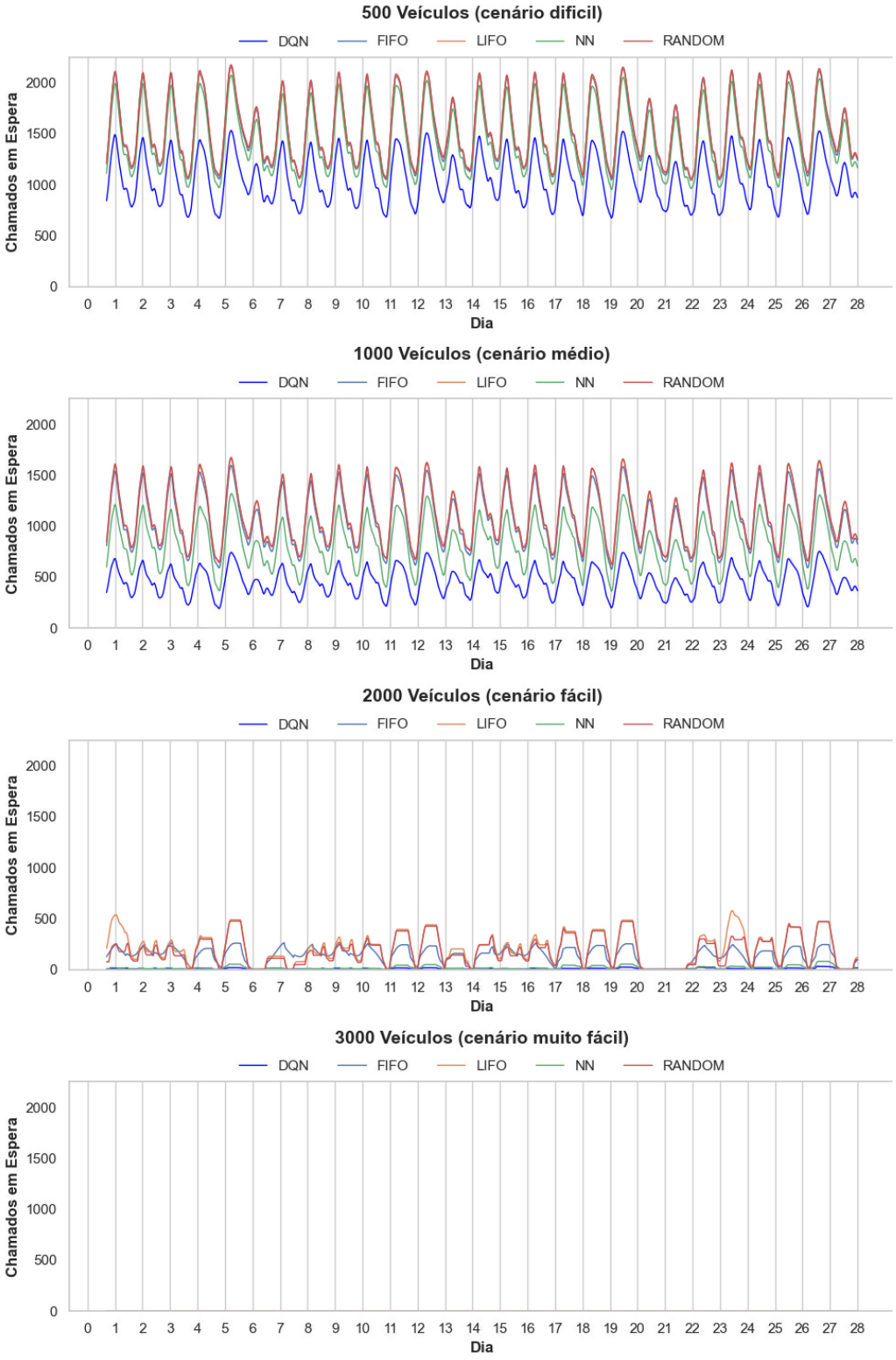
Gráfico 14 – Quantidade de chamados em espera por minuto (10.000 chamados por dia)



* Valores suavizados pela média dos últimos 1000 minutos

Fonte: Elaborada pelo autor.

Gráfico 15 – Média da quantidade de chamados em espera por minuto (100.000 chamados por dia)



* Valores suavizados pela média dos últimos 1000 minutos

Fonte: Elaborada pelo autor.