

ESTUDO E IMPLEMENTAÇÃO DE ALGORITMOS ONLINE BASEADOS EM APRENDIZADO POR REFORÇO PARA O PROBLEMA DOS K-SERVOS

George Felipe Fernandes Vieira

Universidade Federal Rural do Semi-Árido - UFERSA
Rua Francisco Mota, 572 - Pres. Costa e Silva, Mossoró - RN
georgeffv1@gmail.com

Amanda Gondim de Oliveira

Universidade Federal Rural do Semi-Árido - UFERSA
Rua Francisco Mota, 572 - Pres. Costa e Silva, Mossoró - RN
amandagondim@ufersa.edu.br

Fábio Francisco da Costa Fontes

Universidade Federal Rural do Semi-Árido - UFERSA
Rua Francisco Mota, 572 - Pres. Costa e Silva, Mossoró - RN
fabio_fontes@ufersa.edu.br

RESUMO

O problema dos K-servos consiste em minimizar a alocação servos para atender requisições em um espaço métrico. Este trabalho compara o desempenho dos algoritmos Work Function (WF) e Harmonic com o Q-Learning. Foram realizadas simulações com diferentes configurações de instâncias, variando número de servos, espaço de estados e distribuição de requisições. Os resultados indicam que o Q-Learning, quando bem parametrizado, atinge custos semelhantes ao WF em cenários simples e supera-o em ambientes com padrões estáveis, devido à sua capacidade de adaptação online. Já o WF apresenta desempenho inferior em situações adversas e complexas. Testes estatísticos confirmam a significância das diferenças observadas em instâncias com até 5 servos. Conclui-se que o Q-Learning é uma alternativa promissora ao WF em contextos práticos, embora enfrente limitações com o aumento do espaço de estados, sendo sugerido o uso de técnicas como deep Q-learning.

PALAVRAS CHAVE. K-servos, Aprendizado por Reforço, Q-Learning.

ABSTRACT

The K-server problem consists of minimizing the allocation of servers to meet requests in a metric space. This work compares the performance of the Work Function (WF) and Harmonic algorithms with Q-Learning. Simulations were carried out with different instance configurations, varying the number of servers, state space, and request distribution. The results indicate that Q-Learning, when properly parameterized, achieves costs similar to WF in simple scenarios and outperforms it in environments with stable patterns, due to its online adaptation capability. On the other hand, WF shows inferior performance in adverse and complex situations. Statistical tests confirm the significance of the observed differences in instances with up to 5 servers. It is concluded that

Q-Learning is a promising alternative to WF in practical contexts, although it faces limitations with the increase in state space, suggesting the use of techniques such as deep Q-learning.

KEYWORDS. K-Servers, Reinforcement Learning, Q-Learning

1. Introdução

O problema dos K-servos é um problema da área de algoritmos online, tendo sido inicialmente proposto por Manasse et al. [1988]. Esse problema busca minimizar o custo de movimentação de k servos para atender a uma sequência de requisições em um espaço métrico. A cada nova requisição, o sistema deve decidir, de forma imediata e sem conhecimento das futuras solicitações, qual servo se moverá para atendê-la, com o objetivo de minimizar o custo total acumulado.

O problema dos K-servos consiste em um cenário onde k servos estão posicionados em um espaço métrico e devem atender a uma sequência de requisições que ocorrem em pontos desse espaço. O custo de atender a uma requisição está relacionado à distância percorrida pelo servo que se move até o ponto requisitado.

Rudec et al. [2009] em seu trabalho, aplicou o caso de uso em uma instância concreta do problema. consideremos um conjunto de localizações dentro da área montanhosa e costeira da Croácia, nesse exemplo, busca-se tratar de incêndios florestais que são muito comuns durante o verão. Suponha que $k = 3$ helicópteros extintores de incêndio estejam inicialmente localizados em Rijeka (RI), Zadar (ZD) e Dubrovnik (DU), respectivamente. Se o primeiro incêndio começar, por exemplo, em Split (ST), o algoritmo online precisa decidir qual dos helicópteros disponíveis deve ser movido para esse local. Como representado na figura 1.

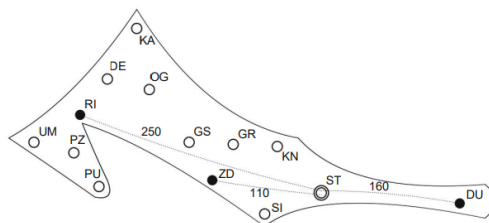


Figura 1: Representação de um problema dos K-servos por Rudec et al. [2009]

Ao longo das últimas décadas, o problema dos K-servos tem sido estudado sob diversas abordagens, destacando-se os algoritmos clássicos como o *Work Function Algorithm* (WFA), que apresenta desempenho competitivo comprovado Koutsoupias e Papadimitriou [1995]. No entanto, esses métodos tradicionais enfrentam desafios significativos quando aplicados a cenários de alta dimensionalidade e dinamicidade, como os presentes em redes de sensores móveis, logística urbana e gerenciamento de recursos em computação em nuvem Borodin e El-Yaniv [1998].

Com os avanços em aprendizado por reforço (*Reinforcement Learning* — RL), novas possibilidades surgem para enfrentar os desafios impostos pela natureza online e incerta do problema. Técnicas como o Q-learning têm demonstrado grande potencial em contextos onde a adaptação e a capacidade de generalização são cruciais Sutton e Barto [2018]. Neste cenário, torna-se relevante investigar a aplicação de RL juntamente com heurísticas ao problema dos K-servos, especialmente em ambientes de grande escala e sujeitos a variações dinâmicas.

1.1. Motivação

O problema dos K -servos persiste como um desafio central em algoritmos online com aplicações críticas em computação em nuvem e redes de sensores móveis Borodin e El-Yaniv [1998]. Apesar do marco teórico estabelecido pelo algoritmo *Work Function*, lacunas práticas em cenários de alta dimensionalidade motivam a integração com técnicas de aprendizado por reforço (RL) Sutton e Barto [2018].

Este trabalho justifica-se por três contribuições principais:

- Aplicação de técnicas de aprendizado por reforço a fim de atenuar o problema da maldição da dimensionalidade
- Implementação de Q-learning com utilização de estados quantificados a partir de espaços métricos
- Estudo comparativo entre algoritmos online clássicos e o Q-Learning

Apesar dos avanços teóricos, trabalhos recentes como o de Coester e Koutsoupas [2019] evidenciam a complexidade inerente a generalizações do problema clássico, onde abordagens puramente competitivas apresentam limitações fundamentais em espaços métricos não uniformes. A metodologia deste trabalho busca avançar o estado da arte ao integrar técnicas de aprendizado adaptativo com modelos clássicos online.

2. Revisão Bibliográfica

2.1. O Problema dos K -Servos

O problema dos K -servos é um problema clássico no contexto de problemas de otimização, destacam-se os algoritmos online como as soluções eficientes para problemas desse tipo. Seu apelo reside na capacidade de modelar sistemas que exigem decisões em tempo real, com aplicações que vão desde transporte autônomo até alocação de recursos em data centers.

2.1.1. Definição Formal

Formalmente, o problema é definido em um espaço métrico (X, d) , onde X representa o conjunto de pontos e $d: X \times X \rightarrow \mathbb{R}$ é uma função métrica. Os K servos iniciam em uma configuração inicial $S = \{s_{i0}, s_{20}, s_{30}, \dots, s_{K0}\}$, e uma sequência de demandas $D = \{D_1, D_2, \dots, D_n\}$ surge ao longo do tempo. A cada demanda D_i , um dos servos deve ser designado para atendê-la, com o objetivo de minimizar o custo total de deslocamento, dado por:

$$\text{Custo Total} = \sum_{i=1}^n d(s_{j_i}, D_i), \quad [1]$$

em que s_{j_i} representa o servo responsável por atender à demanda D_i .

O principal desafio reside no caráter online do problema: as decisões devem ser tomadas sem conhecimento da sequência completa de demandas futuras, o que exige estratégias eficientes em contextos de incerteza.

2.2. Problemas de Otimização Online

Os problemas de otimização online caracterizam-se por exigir decisões sequenciais com informações parciais, à medida que os dados são revelados de forma incremental e irreversível ao longo do tempo. Essa estrutura é típica em aplicações como roteamento de veículos e sistemas distribuídos Borodin e El-Yaniv [1998]. A avaliação da eficiência de algoritmos online é feita por meio da razão competitiva (*competitive ratio*), sendo desejável que esse valor seja próximo de 1 Koutsoupas e Papadimitriou [1995]. Definida como:

$$\text{Razão Competitiva} = \frac{\text{Custo do Algoritmo Online}}{\text{Custo da Solução Ótima Offline}}, \quad [2]$$

2.3. Espaço Métrico

O espaço métrico (X, d) constitui a base formal do problema dos K -servos. Nele, a função de distância d deve satisfazer as propriedades de não negatividade, identidade dos indiscerníveis, simetria e desigualdade triangular Rudin [1976]. Esses espaços são fundamentais para modelagem de problemas em grafos, aprendizado de máquina e sistemas de roteamento.

2.4. Maldição da Dimensionalidade

O termo “maldição da dimensionalidade”, cunhado por Bellman Bellman [1961], descreve os desafios enfrentados por algoritmos em espaços de alta dimensão. À medida que o número de dimensões cresce, a complexidade computacional aumenta exponencialmente, e a noção de proximidade entre pontos torna-se menos informativa. Esse fenômeno impacta diretamente áreas como aprendizado de máquina, exigindo técnicas como redução de dimensionalidade e métodos não paramétricos Bishop [2006].

2.5. Aprendizado por Reforço

O aprendizado por reforço (RL) é uma subárea do aprendizado de máquina que se concentra na interação de agentes com um ambiente dinâmico para alcançar objetivos definidos. Diferentemente de abordagens supervisionadas, nas quais o modelo é treinado com base em exemplos rotulados, no RL o agente aprende a tomar decisões através da interação contínua com o ambiente, utilizando um mecanismo de recompensa e penalidade como guia Sutton e Barto [2018].

O principal objetivo do RL é treinar um agente para maximizar uma função de recompensa cumulativa ao longo do tempo. Essa abordagem é particularmente útil em cenários onde o feedback não é imediato, mas está relacionado a uma sequência de ações que levam a um resultado final. Exemplos incluem controle de robôs, navegação autônoma, jogos e otimização de processos industriais Kaelbling et al. [1996].

O problema de RL pode ser formalizado como um Processo de Decisão de Markov (Markov Decision Process, MDP), que consiste nos seguintes componentes principais [Sutton e Barto, 2018].

- **Estado (s):** Representa a condição atual do ambiente.
- **Ação (a):** Escolha feita pelo agente para interagir com o ambiente.
- **Recompensa (r):** Feedback recebido após executar uma ação.
- **Política (π):** Estratégia que o agente utiliza para decidir suas ações com base no estado.
- **Função de Valor ($V(s)$):** Representa o valor esperado de recompensa que pode ser acumulado a partir de um estado s .

Esses elementos permitem que o agente avalie o impacto de suas decisões e melhore continuamente sua política por meio de métodos iterativos de aprendizado Puterman [1994].

O RL ganhou destaque devido à sua aplicabilidade em uma ampla gama de problemas complexos que não podem ser resolvidos apenas com aprendizado supervisionado. Exemplos notáveis incluem o desenvolvimento de sistemas de recomendação, como em plataformas de *streaming*, controle de tráfego em redes de comunicação e até mesmo o avanço em jogos complexos, como o famoso AlphaGo, como comprovado por Silver et al. [2016].

Além disso, o RL tem demonstrado grande potencial em cenários envolvendo sistemas dinâmicos, como a gestão de energia em veículos elétricos autônomos e a otimização de processos logísticos. Esses avanços consolidam sua relevância como uma das abordagens mais promissoras dentro do campo da inteligência artificial Mnih et al. [2015].

2.5.1. Q-Learning

O Q-learning é um algoritmo de aprendizado por reforço projetado para encontrar a política ótima de tomada de decisão em um determinado ambiente. Sua principal característica é ser um método *off-policy*, o que significa que ele aprende a política ótima de forma independente das ações que são de fato executadas durante o processo de aprendizagem. Em outras palavras,

enquanto o agente pode explorar o ambiente tomando ações não-ótimas, o Q-learning utiliza essas experiências para aprender o valor da melhor ação possível em cada estado. Ele se baseia no conceito de função $Q(s, a)$ (estado-ação), que representa o valor esperado da recompensa ao executar uma ação a em um estado s , seguido de comportamentos ótimos subsequentes Sutton e Barto [2018].

O Q-learning começa com uma inicialização arbitrária da função $Q(s, a)$ para todos os estados e ações possíveis. A cada iteração, o agente interage com o ambiente, observa um estado atual s_t , escolhe uma ação a_t , e transita para um novo estado s_{t+1} , recebendo uma recompensa r_t . A função $Q(s, a)$ é então atualizada com base na seguinte fórmula de atualização:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad [3]$$

Temos que, onde $\alpha \in (0, 1]$ é a taxa de aprendizado, e $\gamma \in [0, 1]$ é o fator de desconto que controla a importância de recompensas futuras em relação às recompensas imediatas Watkins e Dayan [1992].

Uma característica essencial do Q-learning é o balanceamento entre exploração e exploração. Durante a aprendizagem, o agente precisa explorar o ambiente (escolher ações aleatórias) para descobrir recompensas desconhecidas, processo esse denominado com exploração, mas também deve explorar as ações que já são conhecidas por fornecer altos valores $Q(s, a)$. Uma estratégia comum para isso é o método ϵ -greedy, onde o agente escolhe uma ação aleatória com probabilidade ϵ , e a ação com maior valor $Q(s, a)$ com probabilidade $1 - \epsilon$, este por sua vez, é chamado de exploração.

Sob condições adequadas, como a exploração contínua de todos os estados e ações, o algoritmo Q-learning é garantido a convergir para uma função $Q^*(s, a)$ ótima, que define a política ótima para o problema Watkins e Dayan [1992]. Este algoritmo tem ampla aplicação em problemas de controle, navegação autônoma, jogos e até mesmo em cenários complexos de otimização, como o roteamento de veículos elétricos autônomos Mnih et al. [2015].

2.6. Algoritmos Online: Work Function e Harmonic

Dentre os algoritmos mais importantes para o problema dos k-servos, destacam-se o algoritmo Work Function e o algoritmo Harmonic, cada um com abordagens distintas na tomada de decisão, ambos são considerados algoritmos clássicos da literatura quando trata-se de algoritmos online.

2.6.1. Work Function

O algoritmo Work Function baseia-se no conceito de função trabalho, que representa o custo mínimo acumulado para levar os servos de uma configuração inicial até uma determinada configuração futura, atendendo a todas as requisições intermediárias. Ele opera escolhendo a configuração de servos que minimiza a soma do custo do movimento mais o valor da função trabalho. Matematicamente, a work function é definida como:

$$w_t(X) = \min \left\{ \sum_{i=1}^{t+1} d(C_{i-1}, C_i) : C_i \in M(k), r_i \in C_i, C_{t+1} = X \right\} \quad [4]$$

onde $d(\cdot, \cdot)$ denota a distância no espaço métrico.

O algoritmo Work Function é determinístico e apresenta uma razão competitiva de $2k-1$, o que o torna um dos melhores algoritmos conhecidos para o problema dos k-servos em métricas

gerais Koutsoupias e Papadimitriou [1995]. No entanto, sua complexidade computacional é elevada, pois exige o cálculo e a manutenção da função trabalho para todas as possíveis configurações dos servos.

2.6.2. Harmonic

O algoritmo Harmonic é uma abordagem aleatorizada que se baseia na escolha probabilística do servo que atenderá uma requisição. A probabilidade de um servo ser escolhido é inversamente proporcional à sua distância até a requisição. Em um espaço métrico com $k+1$ pontos, pode-se demonstrar que a razão competitiva do algoritmo Harmonic é exatamente $(k(k+1))/2$. Para espaços métricos gerais, resultados posteriores estabeleceram uma cota superior da ordem de $O(2k \log k)$ Bartal e Grove [2000].

Apesar de sua maior simplicidade e menor custo computacional, o algoritmo Harmonic tende a ter um desempenho inferior ao algoritmo Work Function em algumas classes de métricas. Contudo, sua natureza aleatorizada pode ser vantajosa em casos onde a previsibilidade dos algoritmos determinísticos é explorada adversarialmente Raghavan e Snir [1994].

3. Metodologia

A presente metodologia visa comparar o desempenho do algoritmo de aprendizado por reforço Q-Learning com os algoritmos clássicos Work Function Algorithm (WFA) e o Harmonic no contexto do problema dos K-servos. Para isso, foi desenvolvido um ambiente de simulação customizado que modela instâncias do problema com diferentes configurações de servos, pontos e sequências de requisições. O objetivo é avaliar a capacidade do agente baseado em Q-Learning em aprender políticas de decisão eficientes e compará-las com as abordagens online representadas pelo WFA e Harmonic.

3.1. Ambiente de Simulação

Foi desenvolvido um ambiente interativo em Python, o ambiente de simulação foi executado em um notebook Acer Nitro V15, modelo ANV15-51, equipado com processador Intel Core i5-13420H (13ª geração, 8 núcleos e 12 threads), 8GB de memória RAM DDR5, SSD NVMe de 512GB e placa gráfica dedicada NVIDIA GeForce RTX 3050 com 6GB de VRAM GDDR6.

3.2. Agente Q-Learning

A modelagem do algoritmo Q-Learning para o problema dos K-Servos exige uma representação adequada dos estados, ações e recompensas, de forma a capturar eficientemente a dinâmica de atendimento das demandas por múltiplos servidores móveis (servos). O problema dos K-Servos consiste em posicionar k servidores em um espaço de estados, de modo que, a cada nova demanda, apenas um deles se mova para atender à solicitação, minimizando o custo acumulado de movimentações ao longo do tempo.

No contexto do Q-Learning, o **estado** é representado pela configuração atual dos k servidores, isto é, a posição de cada um deles no espaço. Para evitar redundâncias e garantir a consistência, assume-se que dois servidores nunca ocupam o mesmo estado simultaneamente. Além disso, o estado é estendido para incluir a posição da demanda atual, resultando em uma tupla que descreve a configuração completa do sistema. Um exemplo de estado seria: $(s_1, s_2, \dots, s_k, d)$, onde s_k representa a posição do k -ésimo servo e d a posição da demanda.

As ações correspondem à escolha de qual dos k servidores será responsável por atender à demanda presente. Assim, em cada etapa do processo de aprendizado, o agente escolhe um dos servos para se mover até a posição da demanda. A ação escolhida modifica a configuração dos servidores, gerando um novo estado, e essa transição é registrada na Q-table, que armazena os valores $Q(s, a)$ estimando a qualidade de se escolher a ação a no estado s , como indicado na Figura 2.

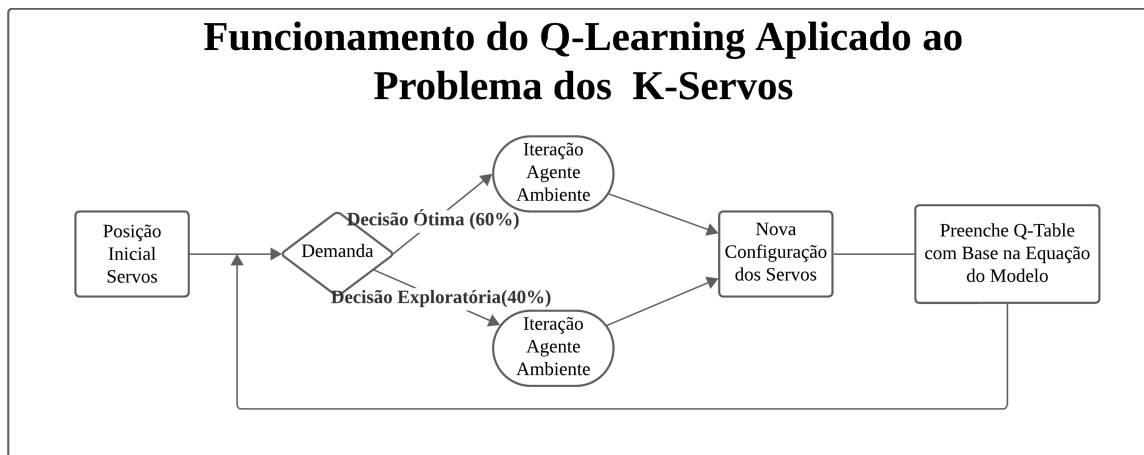


Figura 2: Diagrama de funcionamento do *Q-Learning* Aplicado ao Problema dos K-Servos. Fonte: Autoria própria.

A recompensa r_t é determinada conforme a lógica do problema modelado. No caso do presente trabalho, ela foi projetada para refletir o custo ou benefício da ação tomada, podendo, por exemplo, ser o valor negativo da distância percorrida por um agente (em problemas de roteamento), a penalidade por violar uma restrição do sistema, ou a recompensa positiva por atender corretamente a uma demanda. Assim, a função de recompensa foi construída de modo a guiar o aprendizado em direção a comportamentos desejáveis, ou seja, aqueles que minimizam o custo total ou maximizam a eficiência da solução.

Ao longo do treinamento, cada iteração contribui com uma nova atualização nos valores de Q , baseando-se na diferença entre o valor atual e o valor esperado (estimado pela recompensa obtida somada ao valor máximo das ações futuras).

Para garantir validade das ações, o agente considera apenas aquelas que não causam colisões entre servos. A escolha de ações segue uma política ϵ -greedy, balanceando exploração e exploração. Essa fase é fundamental no início do aprendizado, pois permite que o agente conheça diferentes caminhos possíveis e evite se prender prematuramente a soluções subótimas. Uma estratégia comum é a ϵ -greedy, em que o agente escolhe aleatoriamente uma ação com probabilidade ϵ , promovendo a exploração. Por outro lado, a **exploração** consiste em utilizar o conhecimento já adquirido para escolher a ação com maior valor esperado, ou seja, aquela que parece oferecer a maior recompensa com base nas experiências anteriores. Essa abordagem é mais frequente nas fases finais do treinamento, quando o agente já tem confiança nas estimativas da Q-table.

3.3. Treinamento e Avaliação

Para cada instância do problema (definida em arquivos .json), o agente Q-Learning é treinado por 10.000 episódios. Após o treinamento, a política aprendida é avaliada sobre a mesma sequência de requisições, sendo comparada com a política gerada pelos outros algoritmos analisados (WFA e Harmonic). O custo total de atendimento das requisições é computado para cada abordagem e utilizado como métrica principal de comparação.

3.4. Problemas-teste

Para a avaliação dos algoritmos *Q-Learning*, *Harmonic* e *Work Function* no contexto do problema dos k -servos, foram utilizadas diferentes instâncias sintéticas, variando o número de servos, pontos e solicitações. Essa variação permite testar o comportamento dos algoritmos sob diferentes configurações de complexidade e distribuição espacial.

Cada instância é representada por uma matriz de distâncias, um vetor de posições iniciais dos servos e uma sequência de requisições. A Tabela 1 apresenta um resumo dessas instâncias.

Instância	Nós	servos	Requisições
Instância I	10	4	12
Instância II	100	5	50
Instância III	100	8	200

Tabela 1: Parâmetros das instâncias do problema do k -servo

A seguir, descrevemos e analisamos individualmente cada instância.

Três instâncias distintas do problema do k -servo foram geradas variando o número de nós, quantidade de servos e total de requisições:

- **Instância 1:** Nesta configuração, são utilizados 4 servidores e possui apenas 12 requisições. A matriz de distâncias entre os 10 pontos é simétrica e apresenta uma variedade de custos de deslocamento, permitindo simulações em um espaço relativamente pequeno, mas com variabilidade suficiente para testar a resposta do algoritmo em cenários compactos e dinâmicos.
- **Instância 2:** Esta instância é composta por 5 servidores que atuam em um ambiente com 100 pontos e 50 demandas. A matriz de distâncias entre os pontos segue um padrão crescente, onde os custos de deslocamento aumentam linearmente à medida que os pontos se afastam. Essa estrutura é útil para simulações controladas, nas quais a influência da distância nas decisões do algoritmo pode ser avaliada de forma mais isolada.
- **Instância 3:** Esta instância envolve um total de 8 servidores. Cada servidor começa em uma posição inicial distinta. O sistema deve atender a uma longa sequência de demandas, totalizando 200 requisições que se distribuem ao longo de diversos pontos do espaço. A matriz de distâncias fornece os custos de deslocamento entre 100 pontos distintos, representando um cenário de alta complexidade e grande escala, ideal para avaliar o desempenho do algoritmo em um ambiente denso e realista.

Cada uma das instâncias foi executada em ambos os algoritmos, possibilitando comparações diretas em termos de custo total e eficiência de movimentação. A diversidade dos cenários reforça a robustez da análise, evidenciando o desempenho do algoritmo de aprendizado por reforço frente aos algoritmos clássicos *Work Function* e *Harmonic*.

4. Resultados e Discussões

Os resultados obtidos a partir da avaliação do agente Q-Learning e dos algoritmos clássicos Work Function Algorithm (WFA) e Harmonic são apresentados na Tabela 2. Em geral, observa-se que o WFA apresenta melhor desempenho em instâncias de baixa complexidade, enquanto o Q-Learning se destaca em instâncias de maior escala ou com padrões dinâmicos de requisições.

Com base nos resultados apresentados na Tabela 2, é possível realizar uma análise comparativa entre os algoritmos Q-Learning, Work Function e Harmonic sob diferentes aspectos de desempenho — notadamente custo total e tempo de execução — ao serem aplicados a três instâncias distintas do problema dos K-servos.

Na Instância 1, que apresenta pequeno porte (4 servidores e poucas demandas), o algoritmo Work Function obteve o menor custo total (25), indicando sua efetividade em encontrar soluções próximas do ótimo. O Q-Learning apresentou custo ligeiramente superior (28), demonstrando boa capacidade de aprendizado mesmo em cenários com poucas interações. Já o Harmonic,

Tabela 2: Comparação de desempenho entre os algoritmos nas diferentes instâncias (média de 10x Execuções)

Instância	Algoritmo	Custo Total	Tempo de Execução (s)
Instância 1	Q-Learning	28	0,49
	Work Function	25	0,01
	Harmonic	38	0,084
Instância 2	Q-Learning	1558	3,44
	Work Function	1520	7270,28
	Harmonic	1682	0,33
Instância 3	Q-Learning	1854	5,08
	Work Function	1934	8246,31
	Harmonic	2510	0,57

embora tenha sido o mais rápido (0.084 s), apresentou desempenho inferior em termos de custo (38), evidenciando sua natureza simplificada. O tempo de execução do Q-Learning permaneceu aceitável (0.49 s), apresentando assim uma métrica tempo não muito inferior em relação ao Work Function, o que o torna competitivo nesse cenário.

Na Instância 2, de porte intermediário (5 servidores, 100 demandas), o Work Function novamente obteve o menor custo (1520), porém à custa de um tempo de execução extremamente elevado (7270.28 s), o que compromete sua viabilidade em aplicações em tempo real. O Q-Learning, por sua vez, apresentou custo competitivo (1558) com tempo de execução significativamente menor (3.44 s), destacando-se como uma alternativa eficiente e escalável. O Harmonic manteve seu padrão de rápida execução (0.33 s), porém com custo mais alto (1682), confirmando sua limitação quanto à qualidade da solução obtida.

Na Instância 3, de grande porte (8 servidores, mais de 100 demandas), observou-se um comportamento distinto. O Q-Learning superou o Work Function em termos de custo total (1854 contra 1934), além de ser substancialmente mais rápido (5.08 s contra 8246.31 s). Esse resultado evidencia a capacidade do Q-Learning em lidar com instâncias maiores de forma mais eficiente, possivelmente devido à saturação computacional do Work Function, cujo espaço de estados cresce exponencialmente com o número de servidores e pontos. O Harmonic novamente teve o maior custo (2510), embora tenha mantido sua vantagem em tempo de execução (0.571 s).

Em resumo, os resultados confirmam o trade-off entre qualidade de solução e tempo de execução. O algoritmo Work Function produz os menores custos em instâncias pequenas e médias, mas seu tempo de execução cresce rapidamente. O Q-Learning mostrou-se promissor, oferecendo um bom equilíbrio entre desempenho e escalabilidade. Já o Harmonic, embora útil como referência pela sua simplicidade e rapidez, não se mostrou competitivo em termos de custo total, especialmente em instâncias maiores. Esses achados reforçam a viabilidade do Q-Learning como alternativa prática para o problema dos K-servos, principalmente em contextos onde o tempo de resposta é relevante.

4.1. Discussões

O Q-Learning apresenta vantagens significativas sobre métodos tradicionais como o WFA Algorithm, especialmente ao lidar com a Maldição da Dimensionalidade. À medida que o número de servos e pontos aumenta, o espaço de estados cresce exponencialmente, tornando algoritmos como o WFA computacionalmente inviáveis. O Q-Learning, por sua vez, utiliza decomposição de estados e técnicas de exploração, oferecendo uma abordagem escalável e mais adequada para ambientes de alta dimensão.

Além disso, o Q-Learning demonstra uma capacidade superior de adaptação a ambientes dinâmicos. Enquanto o WFA é competitivo em cenários estáticos, ele encontra limitações quando ocorrem mudanças frequentes na configuração do espaço ou no padrão de requisições. O agente Q-Learning, ao contrário, adapta sua política de decisão continuamente, melhorando o desempenho em ambientes não estacionários.

Outro diferencial é sua capacidade de generalização. Diferentemente de abordagens que exigem um conhecimento completo e prévio do ambiente, o Q-Learning aprende políticas eficientes por meio da interação direta, sem a necessidade de modelar explicitamente todas as possibilidades futuras. Ele também é eficaz na exploração de padrões recorrentes, capturando regularidades em requisições para otimizar suas decisões e superar abordagens determinísticas que não possuem aprendizado adaptativo.

Por fim, o Q-Learning se destaca pela eficiência computacional durante a fase de inferência. Após o treinamento, a aplicação da política aprendida para tomar decisões em tempo real é simples, exigindo apenas consultas à Q-Table, ao passo que o WFA necessita de cálculos mais custosos a cada nova decisão.

Os resultados obtidos demonstram que, embora o Q-Learning não supere o WFA em todas as instâncias, ele oferece vantagens significativas em termos de escalabilidade e adaptabilidade. Dessa forma, o uso de técnicas de aprendizado por reforço se mostra promissor para problemas on-line de alta complexidade, como o dos K-servos, especialmente em cenários realistas que envolvam variabilidade e incerteza.

5. Conclusão

Este trabalho propôs e avaliou uma abordagem baseada em aprendizado por reforço, especificamente o algoritmo Q-Learning, para resolver o problema dos K-servos em ambientes de alta dimensionalidade e dinamicidade. Diferentemente dos métodos tradicionais, como o Work Function Algorithm (WFA) e o Harmonic, que dependem de conhecimento completo e prévio da estrutura de requisições, o Q-Learning oferece a capacidade de adaptação contínua, permitindo o desenvolvimento de políticas de decisão robustas em cenários onde a incerteza e a variabilidade são predominantes.

Inicialmente, foi desenvolvido um ambiente de simulação customizado, permitindo a geração de diversas instâncias do problema com variação no número de servos, pontos e padrões de requisições. O agente baseado em Q-Learning foi treinado e avaliado nestas instâncias, sendo comparado diretamente com o WFA e Harmonic.

Os resultados obtidos indicam que, embora o WFA tenha desempenho superior em instâncias pequenas e estáticas, o agente Q-Learning se destaca em cenários mais complexos, com padrões de requisições variáveis e ambientes não estacionários. A curva de aprendizado do agente demonstrou tendência de melhoria ao longo dos episódios, validando a efetividade do processo de treinamento. Além disso, o agente foi capaz de identificar padrões recorrentes nas demandas, otimizando seus movimentos mesmo sem o conhecimento explícito da sequência futura.

O uso do Q-Learning mostrou-se particularmente adequado para lidar com a maldição da dimensionalidade, problema inerente ao crescimento exponencial do espaço de estados no contexto dos K-servos. Através da exploração e exploração balanceadas, o agente conseguiu explorar eficazmente o espaço de soluções, aprendendo estratégias viáveis sem necessidade de enumerar todas as configurações possíveis.

Outro fator que se destacou foi a eficiência computacional da inferência após o treinamento. Uma vez treinado, o agente pode tomar decisões em tempo real com baixo custo computacional, o que é um requisito fundamental para aplicações práticas em sistemas dinâmicos como redes de sensores, logística de veículos autônomos e gerenciamento de data centers.

Apesar dos avanços, o trabalho também revelou limitações. A dependência de uma boa definição da função de recompensa, o potencial sobreajuste a padrões específicos de requisições e a necessidade de grande capacidade de armazenamento para Q-tables em espaços muito grandes são desafios que ainda precisam ser abordados em pesquisas futuras.

Referências

- Bartal, Y. e Grove, E. (2000). The harmonic k-server algorithm is competitive. *Journal of the ACM*, 47(1):1–15.
- Bellman, R. (1961). *Adaptive control processes: a guided tour*. Princeton University Press.
- Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.
- Borodin, A. e El-Yaniv, R. (1998). *Online computation and competitive analysis*. Cambridge University Press.
- Coester, C. e Koutsoupias, E. (2019). The k-server problem with requests in arbitrary sets. *ACM Transactions on Algorithms*, 15(4):1–19.
- Kaelbling, L. P., Littman, M. L., e Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285.
- Koutsoupias, E. e Papadimitriou, C. (1995). On the k-server conjecture. *Journal of the ACM*, 42: 971–983.
- Manasse, M. S., McGeoch, L. A., e Sleator, D. D. (1988). Competitive algorithms for server problems. *Journal of Algorithms*, 11:208–230.
- Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.
- Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience.
- Raghavan, P. e Snir, M. (1994). The harmonic algorithm for the k-server problem. In *Proceedings of the Symposium on Discrete Algorithms*.
- Rudec, T., Baumgartner, A., e Manger, R. (2009). A fast work function algorithm for solving the k-server problem. *Central European Journal of Operations Research*, 21:1–19.
- Rudin, W. (1976). *Principles of mathematical analysis*. McGraw-Hill.
- Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.
- Sutton, R. S. e Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. MIT Press. ISBN 978-0262039246.
- Watkins, C. J. e Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3-4):279–292.