Avaliação de Modelos de Aprendizado por Reforço no Problema da Mochila Multidimensional: Uma Comparação entre Q-learning e SARSA

Luciano José Menezes de Oliveira
Postgraduate Program in Computer Science
Federal University of Ouro Preto
Ouro Preto, Minas Gerais
luciano.menezes@aluno.ufop.edu.br

Jefferson Vinícius Lourenço de Assis
Postgraduate Program in Computer Science
Federal University of Ouro Preto
Ouro Preto, Minas Gerais
jefferson.assis@aluno.ufop.edu.br

André Luiz Carvalho Ottoni
Department of Computing
Federal University of Ouro Preto
Ouro Preto, Minas Gerais
andre.ottoni@ufop.edu.br

***Abstract*—O Problema da Mochila Multidimensional (MKP) é um desafio clássico de otimização combinatória, presente em diversas aplicações, como alocação de recursos e logística. O aumento da complexidade computacional à medida que o número de itens cresce torna essencial a busca por abordagens eficientes para sua resolução. Neste trabalho, é investigado a aplicação de algoritmos de Aprendizado por Reforço, comparando o desempenho do Q-learning e do SARSA na resolução do problema da mochila. O objetivo do estudo é avaliar o impacto de diferentes funções de recompensa no processo de aprendizado e na qualidade das soluções encontradas. Foram testadas três funções de recompensa com variações no balanceamento entre maximização de valor e penalizações por restrições. Os algoritmos foram avaliados em doze instâncias da OR-Library, com análise de sensibilidade dos parâmetros α, γ e ϵ. Os resultados mostraram que a segunda função recompensa, combinada com o algoritmo SARSA, obteve desempenho mais robusto, atingindo valores ótimos ou próximos na maioria das instâncias. Além disso, o SARSA superou o Q-learning em duas instâncias complexas.**

***Index Terms*—Aprendizado por Reforço. Q-learning. SARSA. Otimização Combinatória.**

I. Introdução

O Problema da Mochila Multidimensional (PMM), ou *Multidimensional Knapsack Problem (MKP)*, é um desafio clássico de otimização combinatória [1]. Esse problema, resolvido de forma binária, possui diversas aplicações práticas, como na indústria siderúrgica [2], logística [3], [4] e até em frameworks para simulações [5], [6]. O objetivo do problema da mochila é selecionar os itens que maximizam o valor total, respeitando múltiplas restrições de capacidade. Entretanto, à medida que o número de itens cresce, o espaço de busca aumenta exponencialmente, tornando a resolução do problema um grande desafio computacional [7].

Diversas abordagens já foram propostas para resolver o MKP. Algoritmos genéticos [8], [9], algoritmos populacionais [10], [11] e busca tabu [12], [13] têm sido amplamente utilizados, demonstrando bons resultados. Além disso, métodos híbridos [14], técnicas baseadas em programação dinâmica [15] e modelos com processos gaussianos [16] foram aplicados ao problema. Por outro lado, algumas abordagens combinam diferentes estratégias para obter soluções mais eficientes [17], [18]. Também foi abordada modelagem matemática para resolver o problema [19].

Nesse contexto, o Aprendizado por Reforço (AR), ou Reinforcement Learning (RL), surge como uma abordagem promissora para otimização combinatória [20]. Esse método permite que um agente aprenda estratégias de decisão por meio da interação com o ambiente, ajustando suas políticas com base nas recompensas recebidas. Trabalhos anteriores já demonstraram a viabilidade do AR para resolver o MKP [21], explorando diferentes formas de implementação [22]. Além disso, estudos recentes aplicaram técnicas de aprendizado por reforço profundo ao MKP [23] e a outros problemas de otimização combinatória [24], [25].

Entretanto, a maioria dos estudos que aplicam aprendizado por reforço a problemas de otimização combinatória utilizam uma única função de recompensa ou não comparam o impacto dos parâmetros do algoritmo. Além disso, poucos trabalhos comparam diferentes algoritmos, como *Q-learning* e SARSA, no contexto do problema da mochila multidimensional. Assim, este artigo busca preencher essa lacuna ao analisar o desempenho dessas duas abordagens em diversas instâncias do problema.

As principais contribuições deste trabalho são:

- Comparação detalhada entre os algoritmos *Q-learning* e SARSA na resolução do MKP.
- Avaliação de três funções de recompensa e seu impacto na busca por soluções ótimas.
- Análise da sensibilidade dos parâmetros α, γ e ϵ para otimizar a convergência dos algoritmos.

Este artigo está organizado em cinco seções. A Seção II apresenta os conceitos fundamentais do Aprendizado por Reforço, incluindo a definição formal do problema, bem como a descrição detalhada dos algoritmos *Q-learning* e SARSA. A Seção III aborda a metodologia adotada, descrevendo as funções de recompensa utilizadas, os parâmetros dos algoritmos e as configurações experimentais. Na Seção IV, são apresentados e discutidos os resultados obtidos, destacando a influência das diferentes configurações no desempenho dos

algoritmos. Por fim, a Seção V sintetiza as conclusões do estudo e sugere direções para trabalhos futuros.

II. Formulação Matemática do Problema da Mochila Multidimensional

O Problema da Mochila Multidimensional (PMM) é um problema no qual se busca selecionar um subconjunto de itens de forma a maximizar o valor total, respeitando múltiplas restrições de capacidade. Diferentemente do problema da mochila tradicional, onde há uma única restrição de peso, nele cada item consome recursos em múltiplas dimensões, tornando a resolução do problema mais complexa [26].

A. Definição Formal

Seja um conjunto de n itens e m restrições de capacidade. Cada item $i \in \{1, 2, \ldots, n\}$ possui um valor associado v_i e consome uma quantidade p_{ji} do recurso j, onde $j \in \{1, 2, \ldots, m\}$. O objetivo é determinar quais itens devem ser incluídos na mochila, respeitando as restrições de capacidade [27].

1) Variáveis de Decisão: Definimos a variável de decisão binária:

$$x_i = \begin{cases} 1, & \text{se o item } i \text{ for incluído na solução,} \\ 0, & \text{caso contrário.} \end{cases} \tag{1}$$

2) Função Objetivo: A função objetivo do problema busca maximizar o valor total dos itens incluídos na mochila:

$$\max \sum_{i=1}^{n} v_i x_i. \tag{2}$$

3) Restrições: Cada restrição impõe um limite na soma dos pesos dos itens incluídos, garantindo que o consumo de cada recurso não exceda a capacidade disponível:

$$\sum_{i=1}^{n} p_{ji} x_i \leq c_j, \quad \forall j \in \{1, 2, \ldots, m\}. \tag{3}$$

Além disso, as variáveis de decisão são restritas ao domínio binário:

$$x_i \in \{0, 1\}, \quad \forall i \in \{1, 2, \ldots, n\}. \tag{4}$$

B. Complexidade Computacional

Este problema pertence à classe de problemas NP-difíceis, pois a quantidade de combinações possíveis cresce exponencialmente com o número de itens e restrições [27]. Dessa forma, abordagens exatas, como programação dinâmica e algoritmos de branch-and-bound, são inviáveis para instâncias grandes. Alternativas heurísticas e metaheurísticas incluindo aprendizado por reforço, citadas na introdução deste trabalho, têm sido amplamente exploradas para resolver esse problema de forma eficiente.

III. Metodologia

Neste trabalho, três diferentes funções recompensas são analisadas para avaliar a eficiência dos algoritmos *Q-learning* e SARSA na resolução do Problema da Mochila Multidimensional. A metodologia proposta segue um fluxo que pode ser visualizado na Figura 1.

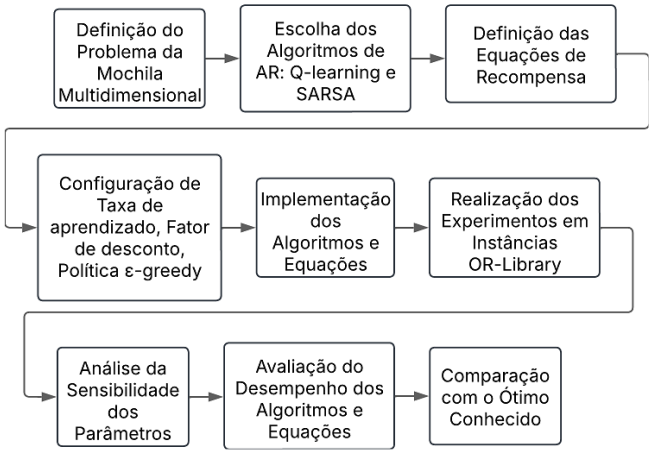

Fig. 1. Fluxo da metodologia proposta neste estudo.

A. Aprendizado por Reforço

O Aprendizado por Reforço (AR) baseia-se nos Processos de Decisão de Markov (MDP), que fornecem a estrutura matemática para modelar o comportamento de um agente em um ambiente dinâmico [28]. O modelo adotado segue a formulação clássica, onde um agente interage com um ambiente através de um conjunto de estados, ações e recompensas.

- Estado (s): Representa a configuração do ambiente no instante de tempo t.
- Ação (a): Define a decisão tomada pelo agente no estado atual.
- Recompensa (r): Retorno imediato obtido ao executar uma ação em um estado.

O objetivo do agente é aprender uma política que maximiza a soma das recompensas ao longo do tempo, chamada de recompensa acumulativa. Essa abordagem permite que o agente explore diferentes possibilidades no ambiente, ajustando suas escolhas para melhorar os resultados futuros.

B. Q-learning

O *Q-learning* é um dos algoritmos mais populares de Aprendizado por Reforço, conhecido por sua simplicidade e eficácia [20]. Ele utiliza uma tabela (Q-table) para armazenar a qualidade Q de cada par estado-ação, que é atualizada iterativamente usando a seguinte função:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right], \tag{5}$$

em que,

- $Q(s, a)$: Valor estimado da recompensa esperada ao executar a ação a no estado s.
- r: Recompensa imediata recebida após executar a ação a no estado s.
- α: Taxa de aprendizado, que controla a velocidade de atualização dos valores.
- γ: Fator de desconto, que pondera a importância das recompensas futuras.
- s': Próximo estado alcançado após executar a ação a.
- a': Ação que maximiza $Q(s', a')$ no próximo estado.

A partir das interações com o ambiente, o agente converge para uma política ótima, maximizando a recompensa acumulativa.

C. SARSA

O SARSA ("*State-Action-Reward-State-Action*") é uma alternativa ao *Q-learning*, mas diferentemente deste, utiliza a ação futura escolhida pelo agente para atualizar os valores da tabela Q [29]. A função de atualização é dada por:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma Q(s', a') - Q(s, a) \right], \quad (6)$$

em que,

- $Q(s, a)$: Valor estimado da recompensa esperada ao executar a ação a no estado s.
- r: Recompensa imediata recebida após executar a ação a no estado s.
- α: Taxa de aprendizado, que controla a velocidade de atualização dos valores.
- γ: Fator de desconto, que pondera a importância das recompensas futuras.
- s': Próximo estado alcançado após executar a ação a.
- a': Próxima ação escolhida no estado s' de acordo com a política atual.

Essa abordagem tem um comportamento mais conservador, pois considera explicitamente a política seguida pelo agente ao calcular os valores de Q.

A comparação desses dois principais algoritmos de Aprendizado por Reforço para a resolução de problemas de otimização combinatória motivou o início deste estudo.

D. Funções de Recompensa

Neste estudo, três funções de recompensa foram analisadas:

1) Recompensa 1: A recompensa simples por item utiliza uma abordagem que considera o valor dos itens subtraído por possíveis penalidades devido a violações das restrições de capacidade [20].

$$r(s, a) = v_i - p_{ji}, \quad j = 1, \ldots, m \quad (7)$$

Onde:

- $r(s, a)$: Recompensa recebida ao executar a ação a no estado s.
- v_i: Valor associado ao item i selecionado.
- p_{ji}: Quantidade do recurso j consumida pelo item i.
- m: Número total de restrições.

2) Recompensa 2: A recompensa multiobjetivo com controle do valor ótimo busca equilibrar a maximização do valor total com o respeito às restrições do problema.

$$R(s, a) = \sum_{j=1}^{m} \frac{w_j}{c_j} - \sum_{j=1}^{m} \max(0, w_j - c_j) + V(s) \quad (8)$$

Onde:

- $R(s, a)$: Recompensa total recebida após executar a ação a no estado s.
- w_j: Quantidade acumulada do recurso j utilizada após a ação.
- c_j: Capacidade total disponível do recurso j.
- m: Número total de restrições.
- $V(s)$: Valor total acumulado dos itens incluídos até o estado s.

Essa formulação é inspirada em métodos de normalização e penalização por excesso, comuns em técnicas de otimização multiobjetivo, onde as penalizações são adaptativas e visam guiar o aprendizado para regiões viáveis sem descaracterizar completamente o espaço de busca..

3) Recompensa 3: A função de recompensa híbrida baseada na eficiência e penalidades combina eficiência relativa, penalidades por excesso de capacidade e uma penalização proporcional ao uso das mochilas.

$$R(s, a) = \begin{cases} \frac{v_a}{\sum_{j=1}^{m} p_{j,a}} - \beta \frac{\sum_{j=1}^{m} w_j}{\sum_{j=1}^{m} c_j}, & \text{se } w_j \leq c_j \; \forall j; \\ -\gamma, & \text{se } \exists j \text{ tal que } w_j > c_j. \end{cases} \quad (9)$$

Onde:

- $R(s, a)$: Recompensa calculada após executar a ação a no estado s.
- v_a: Valor associado ao item escolhido pela ação a.
- $p_{j,a}$: Quantidade do recurso j consumida pelo item a.
- w_j: Quantidade acumulada do recurso j utilizada após a ação.
- c_j: Capacidade total disponível do recurso j.
- β: Fator de penalização proporcional ao uso acumulado das mochilas.
- γ: Penalidade fixa aplicada se alguma restrição for violada.
- m: Número total de restrições.

Essa função é inspirada em modelos de eficiência relativa, comumente utilizados em análise envoltória de dados, e em funções de custo benefício em economia matemática. Além disso, a penalização direta por violações das restrições faz com que o modelo rejeite fortemente soluções inviáveis, tornando essa abordagem adequada para problemas onde a busca por soluções factíveis é mais desafiadora.

A Tabela I apresenta a comparação entre as três funções de recompensa utilizadas na resolução do Problema da Mochila Multidimensional. São destacadas as diferenças em relação à abordagem adotada, se realizam o controle do ótimo, o tipo de penalidade aplicada e a complexidade de cada método.

Essas características influenciam diretamente o desempenho dos algoritmos *Q-learning* e SARSA na busca por soluções viáveis e otimizadas.

TABLE I
COMPARAÇÃO DAS TRÊS FUNÇÕES DE RECOMPENSA

Característica	**R1**	**R2**	**R3**
Abordagem	Valor-Peso	Eficiência	Eficiência
Controle Ótimo	Não	Sim	Parcial
Penalidade	Implícita	Severa	Fixa
Complexidade	Baixa	Média	Alta

E. Planejamento dos Experimentos

Os experimentos foram realizados utilizando uma abordagem de *grid search* sobre diferentes combinações de parâmetros (α, γ, ϵ) para avaliar o impacto das escolhas na busca por soluções ótimas. Cada configuração foi executada por 1000 episódios, repetidos cinco vezes para garantir a robustez dos resultados.

As configurações utilizadas nos experimentos foram ajustadas para explorar as diferentes dinâmicas dos algoritmos. Os parâmetros configurados incluem:

- Taxa de aprendizado (α): Define a velocidade de atualização dos valores da matriz Q. Os valores testados foram $0,01; 0,15; 0,3; 0,45; 0,6; 0,75; 0,9; 0,99$.
- Fator de desconto (γ): Controla o peso atribuído às recompensas futuras. Os valores testados foram $0,01; 0,15; 0,3; 0,45; 0,6; 0,75; 0,9; 0,99$.
- Política ϵ-greedy (ϵ): Equilibra exploração e exploração de ações no espaço de busca. Os valores testados foram $0; 0,05; 0,1$.
- Ambiente Computacional: Os experimentos foram realizados em um computador com processador Intel Core i7, sistema operacional Windows, 8GB de memória RAM, e utilizando Google Colab como ambiente de execução.
- Código e Instâncias: O código foi implementado em Python e executado no Google Colab. As instâncias do problema da mochila foram armazenadas em arquivos .m dentro do Google Drive, de onde foram carregadas.

Foi obtido o melhor resultado de cada algoritmo, além dos parametros que geraram aquele resultado e a média das vezes. [1]

F. Dataset e Instâncias

Foram utilizadas 12 instâncias da OR-Library para avaliar a eficiência dos algoritmos *Q-learning* e SARSA. As instâncias selecionadas incluem conjuntos clássicos como sento1, pb1, weish30, hp2, pb2, hp1, weing1, weing2, weing3, weish1, weish2 e weing7 permitindo uma análise comparativa dos métodos. Os arquivos estão estruturados com as informações a seguir:

- otimo = melhor valor do problema.
- m = número de mochilas.
- n = número de itens.
- v = valor de cada objeto.
- c = capacidade de cada mochila (restrição).
- p = peso de cada objeto.

[1]O código-fonte utilizado neste estudo está disponível em: https://github.com/luucianomenezes/mkp-reinforcement-learning.git

IV. RESULTADOS E DISCUSSÕES

Nos experimentos realizados, foram avaliadas três diferentes funções de recompensa (Eq. 7, Eq. 8 e Eq. 9) em combinação com os algoritmos *Q-learning* e SARSA. O objetivo foi analisar como influenciam a busca por soluções ótimas no Problema da Mochila e avaliar a sensibilidade dos parâmetros α, γ e ϵ. As melhores configurações e os valores obtidos são apresentados e discutidos abaixo.

A. Sensibilidade dos Hiperparâmetros

A sensibilidade dos parâmetros foi avaliada com base nos resultados obtidos em diferentes instâncias. As observações principais são:

1) Taxa de aprendizado (α):

- Valores baixos ($\alpha = 0,01$) garantiram maior estabilidade em instâncias complexas, como *weing2* e *weing3*.
- Valores intermediários ($\alpha = 0,3$) apresentaram bom equilíbrio entre estabilidade e rapidez de aprendizado, sendo eficazes em instâncias menores, como *sento1* e *pb1*.
- Valores altos ($\alpha = 0,9$) resultaram em oscilações em instâncias complexas, dificultando a convergência para soluções ótimas.

2) Fator de desconto (γ):

- Valores altos ($\gamma = 0,99$) foram essenciais para capturar recompensas futuras em problemas com muitas restrições, como *weish30*.
- Valores moderados ($\gamma = 0,6$) apresentaram resultados satisfatórios em problemas menores, como *pb1* e *hp2*.

3) Política ϵ-greedy (ϵ): Os resultados obtidos para a política ϵ-greedy indicam comportamentos consistentes com o trabalho [20]. As observações principais incluem:

- A configuração $\epsilon = 0,1$ foi a mais eficiente na maioria das instâncias, alcançando a maior média em 10 problemas (*weing1*, *weing2*, *weing3*, *weish1*, *weish2*, *weish30*, *hp1*, *hp2*, *pb1* e *pb2*).
- Por outro lado, a configuração $\epsilon = 0$ resultou nas piores médias em 11 problemas (*sento1*, *weing1*, *weing2*, *weing3*, *weish1*, *weish2*, *weish30*, *hp1*, *hp2*, *pb1* e *pb2*), evidenciando a importância de ações exploratórias.

B. Análise dos Resultados

Os resultados experimentais demonstraram que as diferentes funções de recompensa possuem características específicas que influenciam o desempenho de cada:

- **Eq. 1**: Obteve melhores resultados em problemas simples, como *sento1*, mas apresentou dificuldades em instâncias complexas.
- **Eq. 2**: Mostrou maior robustez em instâncias complexas, alcançando ou se aproximando do ótimo conhecido.
- **Eq. 3**: Teve desempenho mais equilibrado em problemas menores, mas apresentou limitações em instâncias com

muitas restrições assim como em trabalhos anteriores [14].

A Tabela II abaixo resume os resultados obtidos para as instâncias analisadas.

TABLE II
RESULTADOS OBTIDOS PARA AS INSTÂNCIAS DA OR-LIBRARY COM OS MELHORES VALORES POR FUNÇÃO

Problema	**Ótimo**	**R1**	**R2**	**R3**
sento1	7772	7461	7109	4838
pb1	3090	3073	3073	3060
weish30	11191	11109	11191	10181
hp2	3186	3148	3186	3110
pb2	3186	3168	3186	3110
hp1	3418	3418	3418	3404
weing1	141278	141278	141278	141278
weing2	130883	130883	130883	130883
weing3	95677	95677	95677	95677
weish1	4554	4554	4554	4554
weing7	1095445	1095445	1095445	1095445
weish2	4536	4536	4536	4536

A Tabela III resume os melhores resultados obtidos para cada tipo de algoritmo.

TABLE III
MELHORES ALGORITMOS POR FUNÇÃO DE RECOMPENSA NAS INSTÂNCIAS DA OR-LIBRARY

Problema	**Ótimo**	**R1**	**R2**	**R3**
sento1	7772	SARSA	Q-Learning	SARSA
pb1	3090	Q-Learning	SARSA	Q-Learning
weish30	11191	SARSA	SARSA	Q-Learning
hp2	3186	Q-Learning	SARSA	SARSA
pb2	3186	SARSA	SARSA	Q-Learning
hp1	3418	SARSA	SARSA	SARSA
weing1	141278	SARSA	Q-Learning	SARSA
weing2	130883	Q-Learning	Q-Learning	SARSA
weing3	95677	SARSA	Q-Learning	SARSA
weish1	4554	SARSA	SARSA	SARSA
weing7	1095445	SARSA	Q-Learning	Q-Learning
weish2	4536	SARSA	Q-Learning	Q-Learning

A Tabela IV resume os parametros obtidos para a função 1 e os resultados.

TABLE IV
RESULTADOS E PARÂMETROS OBTIDOS PARA A FUNÇÃO 1 COM Q-LEARNING E SARSA

Problema	**Q-Learning**	**SARSA**	α	γ	ϵ
sento1	6682	7461	0,15	0,3	0
pb1	3073	3049	0,01	0,01	0,1
weish30	10940	11109	0,01	0,99	0
hp2	3148	3066	0,15	0,01	0,1
pb2	3143	3168	0,3	0,6	0,1
hp1	3385	3418	0,15	0,01	0,05
weing1	141278	141278	0,01	0,01	0,1
weing2	130883	130883	0,01	0,01	0,1
weing3	95677	95677	0,01	0,01	0,1
weish1	4554	4554	0,01	0,01	0
weing7	1095206	1095445	0,9	0,6	0
weish2	4531	4536	0,01	0,01	0

A Tabela V resume os parâmetros obtidos para a função 2 e os resultados.

TABLE V
RESULTADOS E PARÂMETROS OBTIDOS PARA A FUNÇÃO 2 COM *Q-learning* E SARSA

Problema	**Q-Learning**	**SARSA**	α	γ	ϵ
sento1	7109	6033	0,3	0,6	0,1
pb1	3060	3073	0,01	0,01	0
weish30	11187	11191	0,01	0,01	0
hp2	3180	3186	0,01	0,01	0
pb2	3186	3186	0,01	0,01	0,05
hp1	3404	3418	0,01	0,01	0
weing1	141278	140848	0,01	0,3	0,1
weing2	130883	130879	0,01	0,9	0,1
weing3	95677	95562	0,01	0,01	0
weish1	4554	4554	0,01	0,01	0,05
weing7	1095445	1095442	0,01	0,01	0,1
weish2	4536	4535	0,01	0,01	0,05

A Tabela VI resume os parâmetros obtidos para a função 3 e os resultados.

TABLE VI
RESULTADOS E PARÂMETROS OBTIDOS PARA A FUNÇÃO 3 COM *Q-learning* E SARSA

Problema	**Q-Learning**	**SARSA**	α	γ	ϵ
sento1	4280	4838	0,01	0,01	0
pb1	3060	3028	0,9	0,6	0,1
weish30	10181	9762	0,3	0,01	0,1
hp2	3038	3110	0,6	0,3	0,1
pb2	3110	3049	0,3	0,01	0,1
hp1	3313	3404	0,99	0,9	0,1
weing1	117930	141278	0,99	0,6	0
weing2	101812	130883	0,15	0,45	0
weing3	95506	95677	0,15	0,01	0,05
weish1	4324	4554	0,01	0,01	0
weing7	1095445	1094440	0,15	0,3	0,05
weish2	4536	4436	0,01	0,15	0,1

A Tabela VII mostra as médias encontradas para cada algoritmo após finalizar a exploração e encontrar os melhores valores exibidos anteriormente. Nas colunas o Q representa a média obtida com o *Q-learning* para cada função, já o S representa a média obtida com o SARSA, para cada uma das 3 funções. As colunas Q1, Q2 e Q3 correspondem aos resultados obtidos com o algoritmo *Q-learning* para três diferentes combinações de parâmetros. Da mesma forma, S1, S2 e S3 representam os resultados obtidos com o algoritmo SARSA. Esses resultados mostram a importância de calibrar os parâmetros com base nas características específicas de cada problema, maximizando o desempenho dos algoritmos e também evidencia a importância de ações exploratórias.

TABLE VII
MÉDIA DOS RESULTADOS OBTIDOS PARA OS ALGORITMOS *Q-learning* E SARSA.

Problema	Q1	S1	Q2	S2	Q3	S3
sento1	6423	6689	6666	5792	4134	4394
pb1	3048	3043	3060	3071	3012	2994
weish30	10920	10973	11177	11190	9747	9660
hp2	3082	3048	3180	3185	2880	3080
pb2	3091	3087	3186	3186	3062	3006
hp1	3328	3348	3404	3414	3296	3314
weing1	140733	141075	141271	140047	112036	140094
weing2	130322	130322	130876	130837	99450	128428
weing3	92695	95471	95271	95585	89769	90684
weish1	4554	4554	4553	4553	4062	4434
weing7	1094637	1094103	1095439	1095423	1094216	1092293
weish2	4531	4532	4536	4533	4419	4184

V. CONCLUSÃO

Os experimentos realizados mostram que as escolhas adequadas de função de recompensa e algoritmo de aprendizado têm um impacto significativo na qualidade das soluções obtidas para o Problema da Mochila. O objetivo deste trabalho foi avaliar o desempenho dos algoritmos *Q-learning* e SARSA em combinação com três diferentes funções de recompensa. A principal contribuição foi demonstrar como essas escolhas influenciam a busca por soluções ótimas, fornecendo diretrizes para a seleção de abordagens mais eficazes.

De maneira geral, a função 2 se destacou como a mais robusta, alcançando o maior número de soluções ótimas ou muito próximas do valor ótimo conhecido, especialmente em instâncias mais complexas. O algoritmo SARSA foi levemente predominante nesse cenário, demonstrando convergir para soluções de alta qualidade. A função 1 teve um desempenho consistente em problemas de menor complexidade, enquanto a função 3 mostrou um desempenho menor nessas instâncias. Com base nesses resultados, a combinação do algoritmo SARSA com a função 2 apresentou o melhor desempenho geral, tanto em termos de estabilidade quanto na obtenção de soluções próximas ao ótimo.

Como extensão deste trabalho, será investigado o desenvolvimento de novas funções de recompensa que combinem os pontos fortes das funções avaliadas, bem como a inserção de estratégias adaptativas para ajustar dinamicamente os parâmetros α, γ e ϵ. Além disso, sugere-se a integração com algoritmos meta-heurísticos ou aprendizado profundo para aprimorar ainda mais os resultados obtidos. A predominância do algoritmo SARSA neste estudo destaca seu potencial para problemas combinatórios complexos, mas em algumas instâncias o *Q-learning* apresentou melhor desempenho, o que abre caminhos para técnicas de AutoML na escolha do algoritmo mais adequado. Por fim, explorar a transferência de aprendizado pode ser uma abordagem promissora para a resolução do problema da mochila e em cenários mais amplos [30].

VI. AGRADECIMENTOS

Os autores agradecem à Universidade Federal de Ouro Preto (UFOP) pelo suporte acadêmico e institucional, à Pró-Reitoria de Pesquisa e Pós-Graduação (PROPPI/UFOP) pelo incentivo à pesquisa, e à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) pelo apoio, fundamental para a realização deste estudo.

REFERENCES

[1] A. Fréville, "The multidimensional 0–1 knapsack problem: An overview," *European Journal of Operational Research*, vol. 155, no. 1, pp. 1–21, 2004.

[2] F. d. P. Marques and M. N. Arenales, "O problema da mochila compartimentada e aplicações," *Pesquisa Operacional*, vol. 22, pp. 285–304, 2002.

[3] A. L. C. Ottoni, E. G. Nepomuceno, M. S. Oliveira, and D. C. R. de Oliveira, "Reinforcement learning for the traveling salesman problem with refueling," *Complex Intelligent Systems*, vol. 8, no. 4, pp. 2001–2015, Jun. 2021.

[4] G. K. B. Souza and A. L. C. Ottoni, "AutoRL-TSP-RSM: automated reinforcement learning system with response surface methodology for the traveling salesman problem," *Revista Brasileira de Computação Aplicada*, vol. 13, no. 3, pp. 86–100, Nov. 2021.

[5] G. K. B. Souza and A. L. C. Ottoni, "AutoRL-Sim: Automated reinforcement learning simulator for combinatorial optimization problems," *Modelling*, vol. 5, pp. 1056–1083, 2024.

[6] A. Ottoni, E. G. Nepomuceno, and M. S. Oliveira, "Development of a pedagogical graphical interface for the reinforcement learning," *IEEE Latin America Transactions*, vol. 18, no. 1, pp. 92–94, 2020.

[7] J. Krause, J. A. Cordeiro, and H. S. Lopes, "Comparação de métodos de computação evolucionária para o problema da mochila multidimensional," in *Meta-Heurísticas em Pesquisa Operacional*, Curitiba, PR: Omnipax, 2013, pp. 87–98.

[8] P. Chu and J. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[9] W. Liu and G. Liu, "Genetic algorithm with directional mutation based on greedy strategy for large-scale 0–1 knapsack problems," *International Journal of Advancements in Computing Technology*, vol. 4, no. 3, pp. 66–74, 2012.

[10] L. Ke et al., "An ant colony optimization approach for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 16, no. 1, pp. 65–83, 2010.

[11] T.-C. Lu and G.-R. Yu, "An adaptive population multi-objective quantum-inspired evolutionary algorithm for multi-objective 0/1 knapsack problems," *Information Sciences*, vol. 243, pp. 39–56, 2013.

[12] S. Hanafi and A. Fréville, "An efficient tabu search approach for the 0–1 multidimensional knapsack problem," *European Journal of Operational Research*, vol. 106, no. 2, pp. 659–675, 1998.

[13] Z. Yang, G. Wang, and F. Chu, "An effective grasp and tabu search for the 0–1 quadratic knapsack problem," *Computers and Operations Research*, vol. 40, no. 5, pp. 1176–1185, 2013.

[14] J. A. L. S. Junior, D. J. A. Silva, and R. C. L. Oliveira, "Parametrização de um algoritmo cultural híbrido para o problema da mochila multidimensional," in *XII SBAI - Simpósio Brasileiro de Automação Inteligente*, 2015, pp. 678–688.

[15] A. Rong, J. Figueira, and K. Klamroth, "Dynamic programming based algorithms for the discounted 0–1 knapsack problem," *Applied Mathematics and Computation*, vol. 218, no. 12, pp. 6921–6933, 2012.

[16] S. Glimsdal and O.-C. Granmo, "Gaussian process based optimistic knapsack sampling with applications to stochastic resource allocation," in *Proceedings of the 24th Midwest Artificial Intelligence and Cognitive Science Conference*, 2013, pp. 43–50.

[17] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, and A. A. Cire, "Combining reinforcement learning and constraint programming for combinatorial optimization," *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, pp. 3677–3685, 2021.

[18] T. D. Barrett, W. R. Clements, J. N. Foerster, and A. I. Lvovsky, "Exploratory combinatorial optimization with reinforcement learning," in *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20)*, 2020.

[19] M. Chih et al., "Particle swarm optimization with time-varying acceleration coefficients for the multidimensional knapsack problem," *Applied Mathematical Modelling*, vol. 38, no. 4, pp. 1338–1350, 2014.

[20] A. L. C. Ottoni, E. G. Nepomuceno, and M. S. Oliveira, "Análise do desempenho do aprendizado por reforço na solução do problema da mochila multidimensional," *Revista Brasileira de Computação Aplicada*, vol. 9, no. 3, pp. 56–70, Oct. 2017.

[21] S. Bushaj and İ. E. Büyüktahtakın, "A k-means supported reinforcement learning framework to multi-dimensional knapsack," *Journal of Global Optimization*, vol. 89, pp. 655–685, 2024.
[22] A. Arin and G. Rabadi, "Integrating estimation of distribution algorithms versus q-learning into meta-raps for solving the 0-1 multidimensional knapsack problem," *Computers Industrial Engineering*, 2016.
[23] G. Sur, S. Y. Ryu, J. Kim, and H. Lim, "A Deep Reinforcement Learning-Based Scheme for Solving Multiple Knapsack Problems," *Applied Sciences*, vol. 12, no. 6, p. 3068, 2022. DOI: 10.3390/app12063068.
[24] J. Kallestad, R. Hasibi, A. Hemmati, and K. Sörensen, "A general deep reinforcement learning hyperheuristic framework for solving combinatorial optimization problems," *European Journal of Operational Research*, vol. 309, pp. 446–468, 2023.
[25] K. Li, T. Zhang, R. Wang, Y. Wang, Y. Han, and L. Wang, "Deep reinforcement learning for combinatorial optimization: Covering salesman problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13142–13154, Dec. 2022.
[26] S. Martello and P. Toth, "Knapsack Problems: Algorithms and Computer Implementations," Wiley-Interscience, 1990.
[27] H. Kellerer, U. Pferschy, and D. Pisinger, "Knapsack Problems," Springer, 2004.
[28] A. L. C. Ottoni, E. G. Nepomuceno, M. S. Oliveira, and L. T. Cordeiro, "Análise da influência da taxa de aprendizado e do fator de desconto sobre o desempenho dos algoritmos Q-learning e SARSA: aplicação do aprendizado por reforço na navegação autônoma," *Revista Brasileira de Computação Aplicada*, vol. 8, no. 2, pp. 44–59, Jul. 2016.
[29] R. S. Sutton and A. G. Barto, "Reinforcement learning: An introduction," *MIT Press*, 2nd ed., 2018.
[30] G. K. B. Souza, S. O. S. Santos, A. L. C. Ottoni, M. S. Oliveira, D. C. R. Oliveira, and E. G. Nepomuceno, "Transfer Reinforcement Learning for Combinatorial Optimization Problems," *Algorithms*, vol. 17, no. 2, p. 87, 2024. DOI: 10.3390/a17020087.