Development of a Pedagogical Graphical Interface for the Reinforcement Learning

A. Ottoni, E. Nepomuceno, *Member,* and M. de Oliveira

***Abstract*—In this paper the development of a pedagogical graphical interface for the Reinforcement Learning is presented. The proposed framework aims to help students from simulations of traditional domains in literature: Travelling Salesman Problem, Multidimensional Knapsack Problem and Autonomous Navigation. The modules developed in MATLAB are described and case studies are exemplified in the text.**

***Index Terms*—Reinforcement learning, Graphical interface, Travelling salesman problem, Multidimensional knapsack problem, Autonomous navigation, Artificial intelligence.**

I. Introdução

O Aprendizado por Reforço (AR) é um campo da Inteligência Artificial amplamente abordado na literatura [1], [2], [3], [4]. No AR um agente aprende a partir de reforços/penalidades a selecionar as melhores ações para as diversas situações de um ambiente (estados) [1], [5].

A ideia de se realizar o aprendizado de máquina a partir do sucesso e o fracasso em um ambiente é aplicável em diversos domínios [6], [7], [8]. Nesse sentido, a robótica é uma aplicação de grande relevância do AR [9]. Um exemplo de uso dessa abordagem é na navegação autônoma, quando um robô deve aprender a desviar obstáculos [10], [11]. Outra aplicação do AR que concentra diversos estudos é o campo da otimização combinatória [12], [13], [14], [15]. Nessa área, por exemplo, um agente pode aprender a tomar decisões, de forma a otimizar o custo entre as diversas soluções para o problema [14].

Em pesquisas recentes, os autores deste trabalho analisaram a influência da definição das condições de simulação do AR em três problemas: Problema do Caixeiro Viajante (TSP) [14], [16], Problema da Mochila Multidimensional (MKP) [17] e Navegação Autônoma [11]. Nesse sentido, esses estudos ressaltam a importância de investigar a determinação das condições de aprendizado para a obtenção de bons resultados. No entanto, estudantes iniciantes podem ter dificuldades em verificar a relevância da definição da estrutura do AR, devido ao desafio da programação e à complexidade dos problemas. Para contornar esse problema, têm sido proposto ferramentas gráficas para auxiliar no ensino do Aprendizado por Reforço, como pode ser visto em [18], [19].

O desenvolvimento de interfaces gráficas para usuários (Graphical User Interface - GUI) está presente em diversas pesquisas que envolvem métodos de aprendizagem de máquina (*Machine Learning*) [20], [21], [22]. Nesse mesmo aspecto, GUIs também já foram desenvolvidas em aplicações específicas do AR, como: otimização da operação de um sistema de reservatórios [23], robótica móvel [24] e treinamento de um cão virtual [25].

O MATLAB é um *software* frequentemente utilizado no desenvolvimento de interfaces gráficas para pesquisas em engenharias e ciências exatas [26], [27], [28]. Alguns exemplos são nas áreas de: telecomunicações [26], processamento de imagens [29], sistemas elétricos de potência [27], robótica [30] e otimização [28]. Além disso, o MATLAB também é uma ferramenta bastante utilizada no ensino de disciplinas que envolvam programação e métodos numéricos [31], [32]. Experiências bem sucedidas nessa linha para área de sistemas não-lineares e Arduino podem ser vistas em [33], [34]. Assim, baseando-se nisso, o MATLAB foi adotado para a implementação da interface gráfica proposta neste trabalho.

Dessa forma, este trabalho apresenta uma interface gráfica interativa em MATLAB para o estudo de AR, direcionada para alunos de graduação e pós-graduação. A interface proposta permite ao estudante testar algoritmos de AR, configurar parâmetros de aprendizado e verificar resultados da simulação para instâncias de problemas de otimização combinatória (TSP e MKP) e navegação simulada. Nesse sentido, espera-se facilitar e motivar estudantes no aprendizado de conceitos de AR.

Este artigo está dividido 8 seções: a Seção II apresenta conceitos teóricos sobre AR, TSP, MKP e navegação autônoma; a Seção III descreve a interface gráfica proposta; a Seção IV apresenta uma proposta de metodologia para atividades práticas; a Seção V apresenta os resultados para alguns estudos de casos propostos; a Seção VI mostra resultados de avaliação da ferramenta; a Seção VII discute aspectos comparativos da interface proposta; finalmente a Seção VIII aborda as conclusões.

II. Fundamentação Teórica

A. Aprendizado por Reforço

A teoria dos Processos de Decisão de Markov, em inglês *Markov Decision Processes* (MDP), é a base da formulação do AR [1], [2], [3], [5]. Um MDP é definido por um conjunto de estados possíveis do ambiente (S), um conjunto finito de ações que o agente pode realizar (A), uma função de transição de estados (T) e uma função R que mapeia a recompensa pela execução de uma determinada ação (a) em um estado (s) [1]. Assim, o AR consiste em aprender mapeando ações para as

André L. C. Ottoni, Universidade Federal do Recôncavo da Bahia (UFRB), Cruz das Almas (BA), Brasil, andre.ottoni@ufrb.edu.br.

Erivelton G. Nepomuceno, Universidade Federal de São João del-Rei (UFSJ), São João del-Rei (MG), Brasil, nepomuceno@ufsj.edu.br.

Marcos S. de Oliveira, Universidade Federal de São João del-Rei (UFSJ), São João del-Rei (MG), Brasil, mso@ufsj.edu.br.

Definir os parâmetros: α, γ e ϵ
Para cada s,a inicialize $Q(s,a)$=0
Observe o estado s
while *o critério de parada não é satisfeito* **do**
  Selecione a ação a usando a política $\epsilon - greedy$
  Execute a ação a
  Receba a recompensa imediata r
  Observe o novo estado s'
  Atualize Q(s,a) com a Equação 1
  $s = s$'
end

Fig. 1. Algoritmo *Q-learning*.

situações do ambiente (estados), com o objetivo de maximizar ao longo do tempo o valor da recompensa [1].

O *Q-learning* é um dos algoritmos de AR mais conhecidos e aplicados [1], [2]. O método se baseia na atualização da matriz de aprendizado Q de dimensão n_s (número de estados) $\times$ n_a (número de ações). Cada célula da matriz Q armazena um valor de utilidade estimada (aprendida), referente a uma ação a no estado s. A (1) representa a atualização da matriz Q via algoritmo *Q-learning*:

$$Q_{t+1} = Q_t(s,a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q_t(s,a)], \quad (1)$$

em que, $Q_t(s,a)$ é o valor armazenado no instante t na matriz de aprendizado Q para o par (s, a); Q_{t+1} é a atualização da matriz Q; r é a recompensa pela execução de a em s; $max_{a'} Q(s', a')$ é a utilidade do novo estado s'); α é a taxa de aprendizado e γ é o fator de desconto. A Fig. 1 apresenta o *Q-learning*.

Outro relevante algoritmo de AR é o SARSA. O SARSA é uma modificação do *Q-learning*, pois considera para a atualização da matriz Q também a nova ação (a') selecionada para o novo estado (s'). A (2) apresenta o método de atualização do SARSA:

$$Q_{t+1} = Q_t(s, a) + \alpha[r + \gamma Q_t(s', a') - Q_t(s, a)]. \quad (2)$$

A definição de parâmetros do AR é um ponto importante para um bom desempenho nos experimentos [14], [17], [11]. A taxa de aprendizado (α) regula a velocidade que as novas informações aprendidas se sobrepõem perante as antigas. Já o fator de desconto (γ) visa maximizar a soma de recompensas no futuro. O Algoritmo *Q-learning* (Fig. 1) apresenta ainda a política $\epsilon - greedy$, responsável por controlar a gula e a aleatoriedade na seleção de ações, a partir do parâmetro $\epsilon - greedy$. Todos esses parâmetros podem ser definidos entre 0 e 1.

B. Problema do Caixeiro Viajante

O Problema do Caixeiro Viajante, em inglês, *Travelling Salesman Problem* (TSP) é um problema de otimização combinatória com vasta aplicação na literatura [12], [14], [35], [36], [37]. No TSP, o objetivo é minimizar a distância percorrida em uma rota. O caixeiro viajante (agente) deve passar por todas as localidades uma única vez e retornar à cidade inicial ao final do percurso. Uma possível formulação matemática para o TSP é apresentada em seguida [38]:

$$Min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij}, \quad (3)$$

sujeito a:

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad j = 1, \ldots, N, \quad (4)$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad i = 1, \ldots, N, \quad (5)$$

$$x_{ij} \in \{0, 1\} \qquad i, j = 1, \ldots, N, \quad (6)$$

$$X = x_{ij} \in S \qquad i, j = 1, \ldots, N, \quad (7)$$

em que, N é um conjunto de nós. A (3) retrata o objetivo de minimizar a distância percorrida total do caixeiro na rota. O custo de deslocamento entre duas cidades (i e j) é dado por c_{ij}. Já a variável de decisão $x_{i,j}$, assume 1 se o arco (i,j) compor a solução e 0 caso contrário. As restrições de (4) e (5) asseguram que cada localidade será visitada uma única vez. Além disso, a (6) garante que a variável x_{ij} é binária. Por fim, na restrição de (7), o conjunto S representa qualquer grupo de restrições que eliminem a formação de sub-rotas.

O modelo de AR adotado para a resolução do TSP é baseado em definições de trabalhos anteriores: [12], [14], [36], [37]. O conjunto de estados são todas as localidades (nós) que devem ser acessadas para a formação da rota pelo agente (caixeiro viajante). As ações refletem a intenção de visitar outra localidade (estado) do problema. Já a função de reforço é dada pelo o custo de deslocamento entre os nós i e j multiplicado por -1 ($r_{ij} = -c_{ij}$), assim, quanto maior a distância entre duas cidades, maior é a penalidade.

C. Problema da Mochila Multidimensional

O objetivo do Problema da Mochila Multidimensional, em inglês, *Multidimensional Knapsack Problem* (MKP), é selecionar um conjunto de itens que maximizem o transporte desses objetos em m mochilas (8). Além disso, deve ser respeita a restrição de capacidade para cada mochila (9). Uma possível formulação matemática para o MKP é apresentada de 8 à 10 [39]:

$$Max \sum_{i=1}^{n} v_i x_i, \quad (8)$$

sujeito a:

$$\sum_{i=1}^{n} p_{ji} x_i \leq c_j \qquad j = 1, \ldots, m, \quad (9)$$

$$x_i \in \{0, 1\} \qquad i = 1, \ldots, n, \quad (10)$$

em que, uma variável binária x_i armazena a decisão se um objeto i foi selecionado (1 = 'Sim' ou 0 = 'Não'); n é o

número total de itens disponíveis; m é o número de mochilas; i é o índice referente ao item; j é o índice referente à mochila; v_i é o valor do item i; p_{ji} é o peso do item i referente à mochila j; e c_j é a capacidade máxima da mochila j.

A estrutura de AR adotada foi proposta em [17]. No modelo, um estado identifica qual item i é inserido nas m mochilas no instante t. As ações representam a intenção de inserir um objeto i em $t+1$. Já a função de reforço (11), busca maximizar os valores dos itens (v_i) e penalizar a inserção de objetos alta restrição (p_{ji}):

$$r = v_i - p_{ji}, \qquad j = 1, \ldots, m. \tag{11}$$

D. Navegação Reativa e Deliberativa

A navegação de robôs móveis pode ser dividida em duas arquiteturas básicas: deliberativa e reativa [40], [41]. Na abordagem deliberativa, o robô (agente) utiliza as informações disponíveis sobre o ambiente (ex.: mapas) para planejar uma sequência de ações necessárias para alcançar o objetivo. Resumidamente, na navegação deliberativa, o agente realiza as seguintes etapas: definir o objetivo da tarefa; analisar o modelo interno do mundo (obstáculos, paredes, caminhos livres); buscar ações para atingir o objetivo; e planejar uma sequência de ações [40].

Já as arquiteturas reativas surgiram após as deliberativas [40], [41]. Nesse tipo de navegação modelos internos globais não são utilizados. Além disso, uma arquitetura reativa depende de informações locais obtidas por sensores (sonares, infravermelho, sensor de visão). Assim, os comportamentos reativos são mapeados em percepção (sensores) e ação (atuadores) [40].

A navegação de robôs/agentes de móveis é um importante campo de aplicação do AR [9], [10], [11]. Nesse aspecto, existem pesquisas voltadas para movimentação de robôs físicos e também agentes virtuais [9]. A estrutura de AR adotada neste trabalho foi apresentada em [11]. Em [11], são propostos dois modelos de aprendizado com características reativas e deliberativas. Na primeira estrutura (reativa), o agente possui informações apenas locais e recebe os seguintes reforços: -1 (movimentar por um caminho livre); -100 (colidir com um obstáculo); e 1000 (chegar ao objetivo). Já o segundo modelo incorpora características deliberativas. A recompensa deliberativa utiliza as informações sensoriais do mapa do ambiente para o seu cálculo, e é definida como a distância entre a posição do agente até o objetivo, multiplicada por -1.

III. Descrição da Interface Gráfica

Inicialmente foram elaboradas rotinas no formato MATLAB *Code* ('.m'), responsáveis por aplicar o AR nos estudos de casos abordados [14], [17], [11]. Em seguida, iniciou-se o desenvolvimento do ambiente gráfico adotando o ambiente GUIDE do MATLAB. O GUIDE oferece um conjunto de ferramentas para criação e edição de interfaces. O LARSim é composto por 51 arquivos em MATLAB *Code* e 4 interfaces gráficas.

O simulador desenvolvido possibilita ao usuário definir condições de aprendizado, como: tipo de algoritmo (*Q-learning* ou SARSA), parâmetros (α, γ e ϵ) e número de episódios. Além disso, os resultados são apresentados a partir de gráficos, campos numéricos na interface e também podem ser salvos em arquivos de texto. O LARSim possui três módulos, cada qual responsável pela simulação de uma aplicação do AR:

- LARSim TSP: Problema do Caixeiro Viajante.
- LARSim MKP: Problema da Mochila Multidimensional.
- LARSim Nav: Navegação Reativa e Deliberativa.

Um módulo pode ser iniciado à partir da interface gráfica principal do LARSim (Fig. 2).

Em seguida, cada um dos três módulos são detalhados.

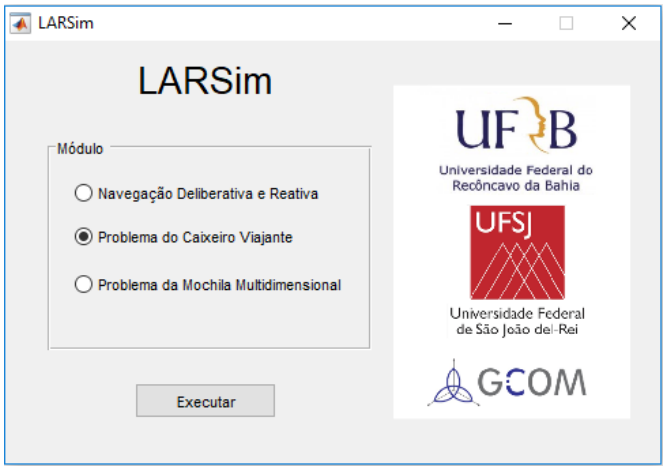

Fig. 2. Interface gráfica principal do LARSim.

A. LARSim TSP

O módulo LARSim TSP possibilita simular estudos de casos do TSP disponíveis na biblioteca TSPLIB (*Traveling Salesman Problem Library*) [30]. A TSPLIB é um repositório de dados amplamente abordado na literatura em pesquisas envolvendo otimização combinatória [12], [14], [36], [37].

O desenvolvimento do código base deste módulo foi iniciado em trabalhos anteriores [14], [16]. As pesquisas realizadas em [14], [16] verificaram que os parâmetros do AR influenciam diretamente no desempenho do cálculo da rota. Nesse sentido, a interface gráfica do LARSim TSP, proposta neste trabalho, permite ao usuário analisar como a definição de condições de aprendizado (algoritmo, parâmetros e número de episódios) podem afetar as rotas geradas para cada instância da TSPLIB selecionada.

A Fig. 3 apresenta a interface gráfica do módulo LARSim TSP. Essa interface é composta por diferentes áreas que permitem desde configurar a simulação até visualizar os resultados. Cada um dos campos do LARSim TSP será explicado na sequência.

A Fig. 4 apresenta os campos responsáveis por definir as condições de aprendizado e também os botões 'Executar' e 'Gráficos':

- Campo 'Algoritmo': definição entre os algoritmos *Q-learning* e SARSA.
- Campo 'Parâmetros': definições dos parâmetros α, γ e ϵ. Esses valores devem estar no intervalo [0, 1].

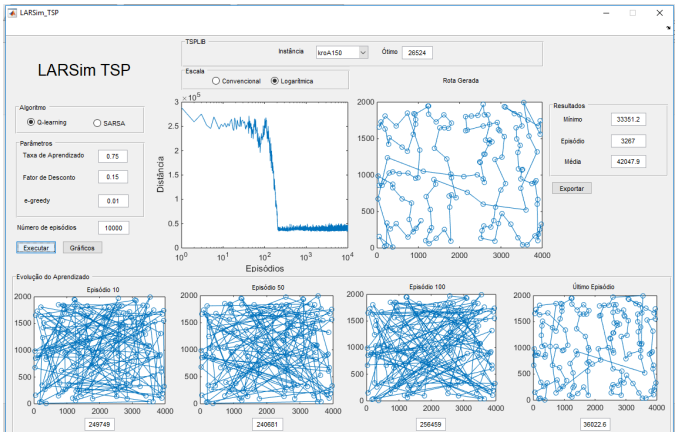

Fig. 3. Interface gráfica do módulo LARSim TSP.

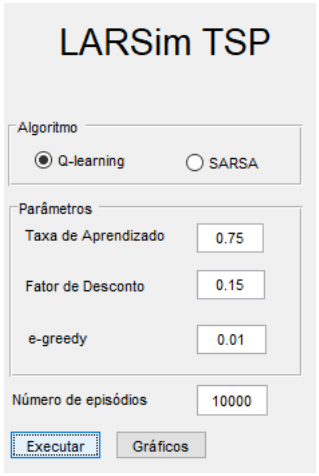

Fig. 4. Interface gráfica do módulo LARSim TSP: campos para definição das condições de aprendizado.

- Campo ‘Número de episódios’: definição do número de episódios de aprendizado. Um episódio corresponde ao agente (caixeiro viajante) completar o cálculo de rota completa.
- Botão ‘Executar’: inicia a simulação.
- Botão ‘Gráficos’: Gera os gráficos resultantes da simulação em janelas separadas.

A Fig. 5, por sua vez, apresenta o painel de identificação da instância (TSPLIB) e os gráficos resultantes de uma simulação:

- Painel ‘TSPLIB’: possui uma caixa de seleção (‘Instância’) para definição da instância da TSPLIB para a simulação. São 10 problemas disponíveis: berlin52, ch130, eil101, kroA100, kroA150, kroA200, lin105, pr124, rat195 e rd400. Além disso, o campo numérico ‘Ótimo’ é preenchido automaticamente com o melhor valor calculado na literatura para a instância e disponibilizado pela TSPLIB.
- Campo ‘Escala’: permite definir a escala do gráfico ‘Episódios x Distância’ para o eixo ‘Episódios’ em ‘Convencional’ ou ‘Logarítmica’.
- Gráfico ‘Episódios x Distância’: permite a visualização do desempenho do AR no cálculo da rota (distância

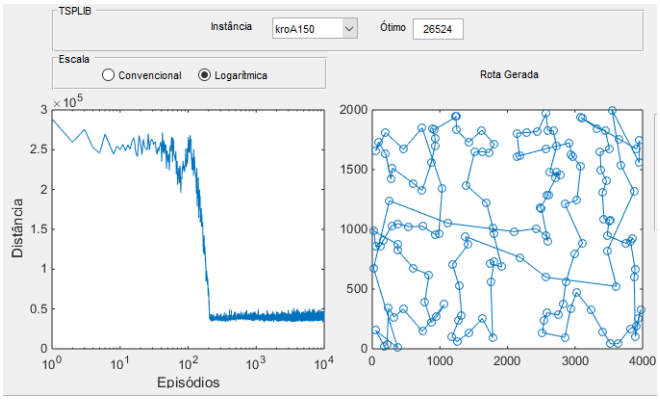

Fig. 5. Interface gráfica do módulo LARSim TSP: painel de identificação da instância (TSPLIB) e gráficos resultantes de uma simulação.

Fig. 6. Interface gráfica do módulo LARSim TSP: painel de resultados.

percorrida pelo caixeiro viajante) ao longo dos episódios de aprendizado.
- Gráfico ‘Rota Gerada’: apresenta a rota referente a menor distância calculada pelo AR ao longo de todos episódios de aprendizado.

Os resultados da simulação também são apresentados em campos numéricos (Fig. 6):

- Campo ‘Mínimo’: menor distância calculada pelo AR para formar uma rota na instância da TSPLIB selecionada.
- Campo ‘Episódio’: episódio referente ao momento durante o aprendizado que o valor ‘Mínimo’ foi calculado.
- Campo ‘Média’: média de valores calculados de distância ao longo dos episódios de aprendizado.
- Botão ‘Exportar’: permite exportar os resultados da simulação para arquivos de texto.

A interface LARSim TSP apresenta ainda (parte inferior da Fig. 3) a evolução do desenho e do cálculo da rota ao longo do aprendizado, para os episódios: 10, 50, 100 e final.

B. LARSim MKP

O módulo LARSim MKP, assim como o LARSim TSP, é voltado para simulações de um problema de otimização combinatória, neste caso, o Problema da Mochila Multidimensional. O repositório de dados abordado é a ORLIB [42].

O código inicial do LARSim MKP foi desenvolvido no trabalho [17]. Em [17], os autores verificaram como os parâmetros (α, γ e ϵ) podem afetar o desempenho do AR na solução do MKP.

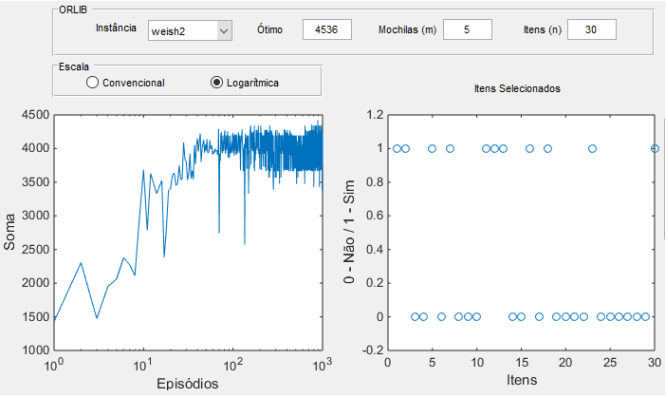

Fig. 7. Interface gráfica do módulo LARSim MKP: painel de identificação da instância (ORLIB) e gráficos resultantes de uma simulação.

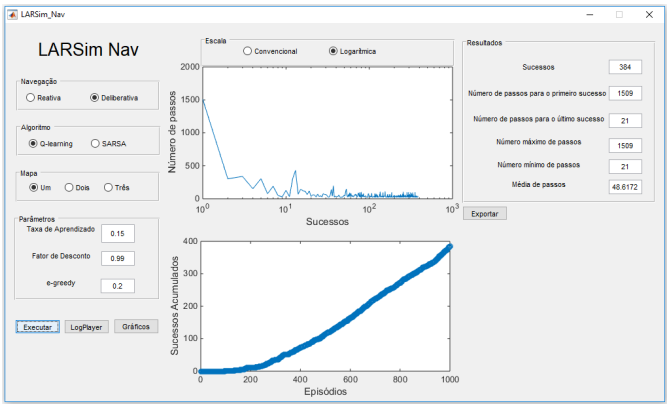

Fig. 8. Interface gráfica do módulo LARSim Nav.

A interface do LARSim MKP é composta por funcionalidades semelhantes às já explicadas para o módulo LARSim TSP, como: configurações de condições de aprendizado (‘Algoritmo’, ‘Parâmetros’, ‘Números de episódios’); botões de ‘Executar’, ‘Gráficos’ e ‘Relatórios’; definição da instância; gráficos e campos numéricos com resultados.

A Fig. 7 detalha o painel de identificação da instância (ORLIB) e os gráficos resultantes de uma simulação:

- Painel ‘ORLIB: possui uma caixa de seleção (‘Instância’) para definição da instância da ORLIB para a simulação. São 16 problemas disponíveis: hp1, hp2, pb1, pb2, sento1, weing1, weing2, weing3, weing4, weing5, weing6, weing7, weish1, weish2, weish3 e weish30. Além disso, os campos numéricos ‘Ótimo’, ‘Mochilas (m)’ e ‘Itens (n)’ são preenchidos automaticamente com os valores disponibilizados pela ORLIB.
- Gráfico ‘Episódios x Soma’: permite a visualização do desempenho do AR no cálculo da soma de valores de itens alocados nas mochilas ao longo dos episódios de aprendizado.
- Gráfico ‘Itens Selecionados’: apresenta os itens selecionados no formato binário (0 – ‘Não’ e 1 – ‘Sim’) referentes ao episódio com o maior encontrado na soma de valores dos objetos.

C. LARSim Nav

O módulo LARSim Nav tem o objetivo de simular o aprendizado da navegação de um agente em um ambiente com obstáculos. O desenvolvimento do código base deste módulo foi apresentada no trabalho [11]. Em [11], os autores investigaram o papel dos algoritmos *Q-learning* e SARSA e dos parâmetros no desempenho da navegação. Nesse sentido, a interface proposta neste trabalho permite ao usuário analisar como as condições de simulação (algoritmo e parâmetros), o tipo de navegação (reativa e deliberativa) e o mapa do ambiente podem influenciar no aprendizado da movimentação.

A Fig. 8 apresenta a interface gráfica do módulo LARSim Nav. O LARSim Nav também apresenta funções similares ao LARSim TSP, como definição de: algoritmo, parâmetros, escala gráfica e resultados gráficos/numéricos. No entanto, vale destacar outras funcionalidades:

- Campo ‘Navegação’: definição entre o tipo de navegação adotada na simulação (reativa ou deliberativa).
- Campo ‘Mapa’: definição do tipo de mapa adotado. Nesta versão, o LARSim Nav apresenta três tipos de ambientes com as disposições dos obstáculos distintas.
- Gráfico ‘Sucessos x Número de Passos’: semelhante ao gráfico de ‘Episódios x Distância’ do LARSim TSP. Permite visualizar o número de passos necessários (movimentos do agente) para alcançar o objetivo no ambiente ao longo do aprendizado.
- Gráfico ‘Episódios x Sucessos Acumulados’: apresenta o acúmulo de sucessos (chegar ao objetivo) ao longo dos episódios. Um episódio inicia quanto o agente sai da posição inicial e termina quando colide em um obstáculo ou chega ao objetivo.

Além disso, vale destacar que o botão ‘LogPlayer’ abre uma nova janela para visualização da navegação do agente ao longo do aprendizado, conforme Fig. 9. O ‘LogPlayer’ realiza a leitura dos *logs* da simulação e possibilita gerar dois gráficos:

- Gráfico ‘Simulação do Ambiente’: visualização *offline* da movimentação do agente durante o aprendizado de acordo com o tipo de mapa selecionado. As cores identificam elementos do ambiente: vermelho (obstáculos e paredes), azul (agente), verde (caminho livre para movimentação), verde claro (ponto inicial) e amarelo (objetivo - estrela).
- Gráfico ‘Iterações x Sucessos Acumulados’: apresenta o acúmulo de sucessos (chegar ao objetivo) ao longo das iterações. Uma iteração representa um passo (movimentação) do agente no ambiente.

IV. Metodologia Pedagógica

Nesta seção, é apresentada uma proposta de metodologia pedagógica para a utilização do LARSim. Para isso, sugere-se a realização de experimentos práticos intercalados com estudos teóricos sobre o AR. A interface gráfica permite a abordagem e análise de importantes aspectos do AR, como: comparação entre os algoritmos *Q-learning* e SARSA; influência da definição dos parâmetros nos resultados; influência do número de episódios na convergência dos algoritmos; análise do desempenho do AR de acordo com a complexidade do problema; estudos de casos em módulos distintos.

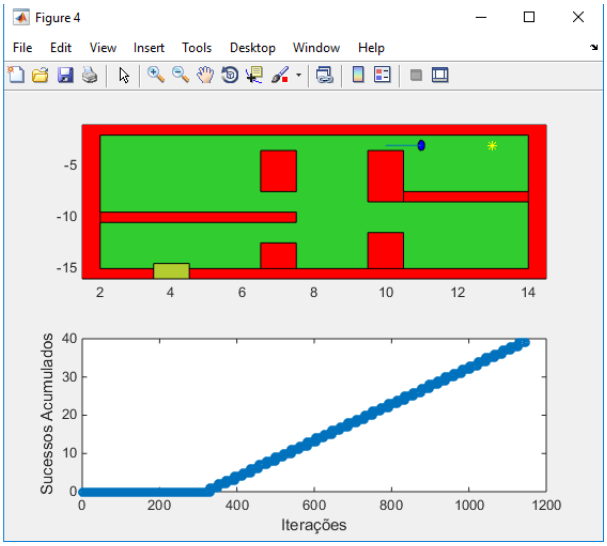

Fig. 9. Interface gráfica do módulo LARSim Nav: janela de visualização do aprendizado da navegação ('Mapa 2').

Em seguida, é apresentada uma sequência de 15 tópicos da metodologia proposta:

1) Estudo de tipos de aprendizado de máquina: supervionado, não-supervisionado, por reforço [5].
2) Experimento 1: introdução ao LARSim.
3) Estudo de conceitos básicos do AR: política, sinal de reforço e função valor [1].
4) Experimento 2: análise da política de seleção de ações no desempenho do AR. Para isso, variar os valores de $\epsilon - greedy$ de 0 (guloso) até 1 (totalmente aleatório) em simulações nos três módulos.
5) Estudo de Processos de Decisão de Markov [1].
6) Experimento 3: análise da dimensionalidade do conjunto de estados e ações no desempenho do AR. Para isso, executar simulações com as instâncias berlin52 (52 estados), kroA200 (200 estados) e rd400 (400 estados) no módulo LARSim TSP.
7) Experimento 4: repetir a análise do Exp. 3 para o módulo LARSim MKP, instâncias: weish2, weish3 e weish30.
8) Experimento 5: análise da influência da observabilidade do ambiente nos resultados. Realizar simulações com o módulo LARSim Nav selecionando o tipo de navegação: reativa (ambiente parcialmente observável) e deliberativo (ambiente completamente observável).
9) Estudo da regra de diferença temporal no AR e dos parâmetros α e γ [1].
10) Experimento 6: análise da variação da taxa de aprendizado nas aplicações dos três módulos.
11) Experimento 7: análise da variação do fator de desconto nas aplicações dos três módulos.
12) Estudo dos algoritmos de diferença temporal: *Q-learning* e SARSA [1].
13) Experimento 8: comparação de desempenho dos algoritmos adotando os mesmos parâmetros (α e γ) em ambos.
14) Experimento 9: comparação de desempenho dos algoritmos adotando parâmetros definidos na literatura: TSP [14], MKP [17] e Nav [11].
15) Experimento 10: proposta de alteração no código base. Ex.: inclusão de outro algoritmo de AR, funções de reforço ou taxa de aprendizado decaindo com o tempo.

De forma a exemplificar uma prática com o LARSim, é apresentada na sequência, a estrutura abordada no Experimento 1. As atividades descritas nos tópicos de 1 à 9 apresentam definições iniciais para realizar simulações utilizando o módulo LARSim TSP. O exercício 10, por sua vez, retrata uma possível questão de análise de evolução do aprendizado.

1) Digite "LARSim" no terminal do MATLAB.
2) Selecione o "Módulo": Problema do Caixeiro Viajante.
3) Selecione o "Algoritmo": *Q-learning*.
4) Definir o "número de episódios": 10000.
5) Definir os "Parâmetros" (conforme [14]): "Taxa de aprendizado" (α) = 0,7273, "Fator de Desconto" (γ) = 0,1539 e $\epsilon - greedy$ = 0,01.
6) Selecione a "Instância" - berlin52.
7) Clique em "Executar" para simular.
8) Clique em "Gráficos" para salvar as figuras.
9) Clique em "Exportar" para salvar os resultados em arquivos de texto.
10) Analise os resultados de "Evolução de Aprendizado" e aponte as diferenças nas rotas geradas e nas distâncias calculadas entre os episódios 10, 50 e 100.

V. Estudos de Casos

Nesta seção, são apresentados estudos de casos simulados no LARSim. Nesse sentido, são discutidos resultados gerados para o TSP e MKP. As configurações dos parâmetros adotadas nos exemplos foram definidas a partir dos valores apresentados em estudos anteriores: [14], [17]. Neste sentido, este trabalho não tem como objetivo expôr as metodologias de estimação de parâmetros, e sim ilustrar a aplicação da interface.

A. 1º Estudo de Caso: LARSim TSP

O primeiro estudo de caso utiliza o módulo LARSim TSP no aprendizado de rotas para a instância kroA200. Nesse aspecto, foram adotadas condições definidas em [14]: algoritmo SARSA, $\alpha = 0{,}7894$, $\gamma = 0{,}0894$, $\epsilon = 0{,}01$ e 10000 episódios. Em [14], os parâmetros α, γ e ϵ foram criteriosamente determinados a partir de experimentos e da modelagem matemática pela Metodologia de Superfície de Resposta.

A Fig. 10 apresenta o gráfico de 'Episódios X Distância' (escala logarítmica). Por esse gráfico, é possível observar que para os 10 primeiros episódios de aprendizado, a distância calculada pelo AR oscila em torno de 350 mil. Em seguida, próximo dos 100 episódios, o valor da rota gerado fica próximo de 300 mil. Entre 100 e 1000 episódios existe uma diminuição significativa na distância calculada. Após isso, é possível verificar que o valor calculado fica abaixo de 50 mil até o último episódio da simulação. Nesse sentido, a Fig. 10 reflete uma característica marcante do AR, a melhora do desempenho do agente aprendiz ao longo do treinamento.

As Figs. 11, 12 e 13 (Gráficos 'Rota Gerada') deixam ainda mais clara a visualização da evolução do aprendizado.

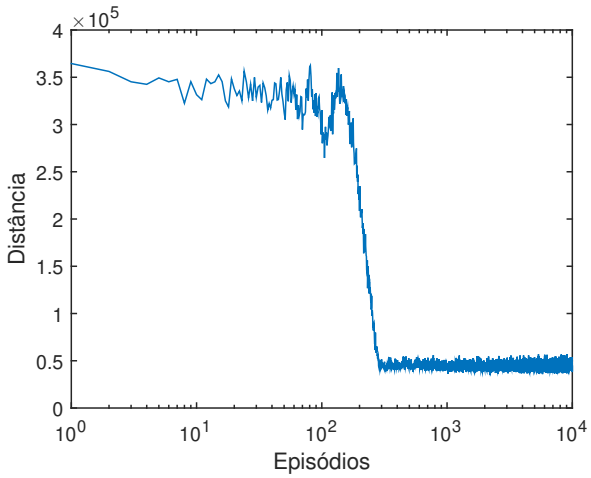

Fig. 10. Gráfico 'Episódios X Distância' para a instância kroA200 (LARSim TSP) para uma simulação com 10000 episódios com o algoritmo SARSA, $\alpha = 0{,}7894$, $\gamma = 0{,}0894$ e $\epsilon = 0{,}01$.

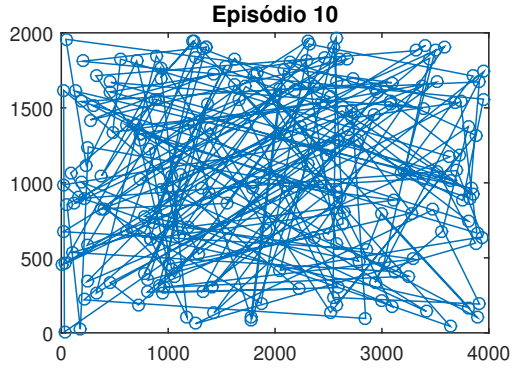

Fig. 11. Gráfico 'Rota Gerada' para a instância kroA200 (LARSim TSP) referente ao décimo episódio em uma simulação com 10000 episódios com o algoritmo SARSA, $\alpha = 0{,}7894$, $\gamma = 0{,}0894$ e $\epsilon = 0{,}01$.

A Fig. 11 apresenta a rota gerada para o início da simulação (episódio 10), onde a distância calculada foi 331527. Já a Fig. 12 representa uma rota com o valor de 315239, calculada pelo AR no centésimo episódio. A Fig. 13, por sua vez, apresenta a menor rota gerada (episódio 2206), com um valor de 34795. A interface do LARSim TSP ainda apresenta a rota gerada no episódio 50 (313239) e o último episódio de simulação (40114).

Na sequência, foram realizadas simulações com o algoritmo *Q-learning*, adotando também as condições definidas em [14] para a instância kroA200: $\alpha = 0{,}6927$, $\gamma = 0{,}2566$ e $\epsilon = 0{,}01$. A Fig. 14 apresenta a menor rota calculada (episódio 8827) com o valor de 38508, distância muito maior do que a calculada pelo SARSA (34795). Nesse sentido, o LARSim TSP oferece a oportunidade de verificar a influência no desempenho de métodos e parâmetros na resolução do TSP.

B. 2º Estudo de Caso: LARSim MKP

O segundo estudo de caso apresenta resultados para uma simulação do Problema da Mochila Multidimensional no módulo LARSim MKP. As configurações adotadas foram

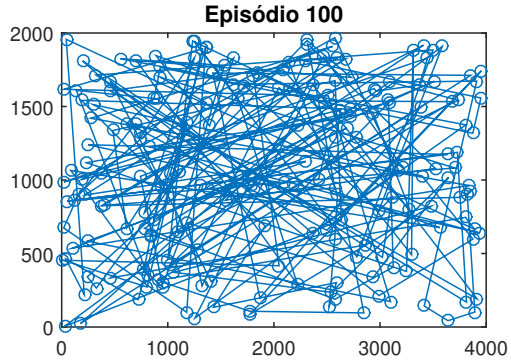

Fig. 12. Gráfico 'Rota Gerada' para a instância kroA200 (LARSim TSP) referente ao centésimo episódio em uma simulação com 10000 episódios com o algoritmo SARSA, $\alpha = 0{,}7894$, $\gamma = 0{,}0894$ e $\epsilon = 0{,}01$.

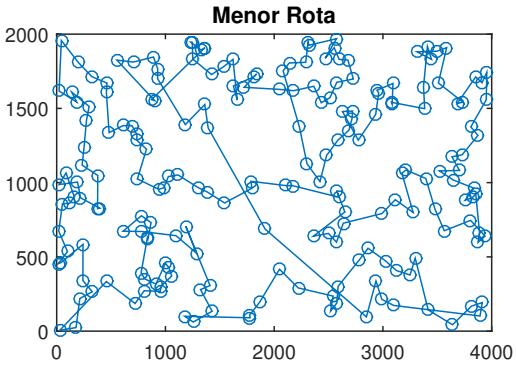

Fig. 13. Gráfico 'Rota Gerada' para a instância kroA200 (LARSim TSP) referente à menor rota calculada (34795 no episódio 2206) em uma simulação com 10000 episódios com algoritmo SARSA, $\alpha = 0{,}7894$, $\gamma = 0{,}0894$ e $\epsilon = 0{,}01$.

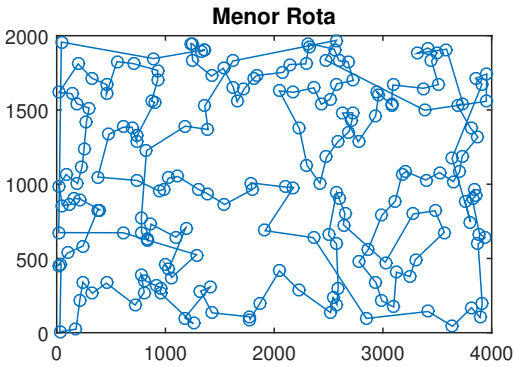

Fig. 14. Gráfico 'Rota Gerada' para a instância kroA200 (LARSim TSP) referente à menor rota calculada (38508 no episódio 8827) em uma simulação com 10000 episódios com algoritmo *Q-learning*, $\alpha = 0{,}6927$, $\gamma = 0{,}2566$ e $\epsilon = 0{,}01$.

baseadas nos experimentos realizados em [17] para a instância weish2: algoritmo *Q-learning*, $\alpha = 0{,}30$, $\gamma = 0{,}45$, $\epsilon = 0{,}10$ e 1000 episódios de aprendizado. A Fig. 15 apresenta o gráfico de 'Itens Selecionados' para o episódio 214, no qual o AR encontrou a solução de 4472. Por esse gráfico, é possível perceber que foram selecionados 12 itens ('1 – Sim'), de um total de 30 objetos disponíveis na instância weish2.

A Fig. 16, por sua vez, exemplifica um relatório gerado pela

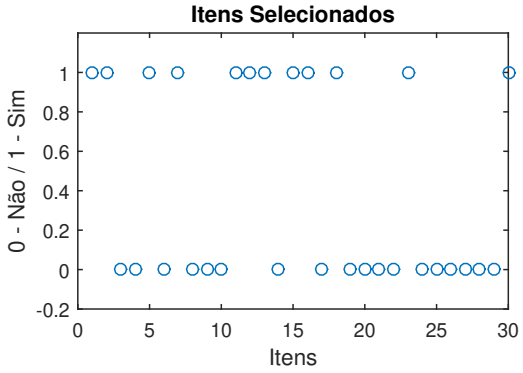

Fig. 15. Gráfico 'Itens Selecionados' para a instância weish2 (LARSim MKP) referente ao maior valor calculado (4472 no episódio 265) em uma simulação com 1000 episódios com o algoritmo *Q-learning*, $\alpha = 0{,}30$, $\gamma = 0{,}45$, $\epsilon = 0{,}10$.

```
Resultados:
Máximo: 4472.000000
Episódio: 265
Média: 3927.617000
------------------------
Itens Selecionados :
  Item (1 - Sim / 0 - Não)    Valor
     1                  1   360.000000
     2                  1    83.000000
     3                  0    59.000000
     4                  0   130.000000
     5                  1   431.000000
     6                  0    67.000000
     7                  1   230.000000
     8                  0    52.000000
     9                  0    93.000000
    10                  0   125.000000
    11                  1   670.000000
    12                  1   892.000000
    13                  1   600.000000
    14                  0    38.000000
    15                  1    48.000000
    16                  1   147.000000
    17                  0    78.000000
    18                  1   256.000000
    19                  0    63.000000
    20                  0    17.000000
    21                  0   120.000000
    22                  0   164.000000
    23                  1   432.000000
    24                  0    35.000000
    25                  0    92.000000
    26                  0   110.000000
    27                  0    22.000000
    28                  0    42.000000
    29                  0    50.000000
    30                  1   323.000000
```

Fig. 16. Relatório em arquivo de texto gerado pela função 'Exportar' (LARSim MKP) para a instância weish2 em uma simulação com 1000 episódios com o algoritmo *Q-learning*, $\alpha = 0{,}30$, $\gamma = 0{,}45$, $\epsilon = 0{,}10$.

função 'Exportar' para o formato de arquivo de texto.

TABELA I
RESULTADOS DE AVALIAÇÃO DA INTERFACE EM CONSULTA OPCIONAL

Item avaliado	**Resultado**
LARSim TSP	9,4
LARSim MKP	9,0
LARSim Nav	9,2
Conteúdo Teórico	9,2
Metodologia Pedagógica	9,3
Nota Geral do Curso	8,8

VI. AVALIAÇÃO DO LARSIM E METODOLOGIA PEDAGÓGICA

A metodologia proposta foi aplicada no curso de "Introdução ao Aprendizado por Reforço", realizado em outubro de 2019 na Universidade Federal de São João del-Rei, como atividade do projeto "Aprendizado por Reforço: Engenharia e Estatística impulsionando a sociedade". Ao final do curso, os estudantes avaliaram (caráter opcional) os módulos da interface gráfica LARSim, a metodologia pedagógica adotada, o conteúdo teórico e o curso de forma geral. Em cada questão, os alunos poderiam dar notas de 0 à 10.

A Tabela I apresenta as notas médias dos 12 estudantes participantes para cada item analisado. Os resultados apontam que os três módulos do LARSim e a metodologia proposta alcançaram notas médias maiores ou iguais a 9,0. Além disso, é importante ressaltar que o LARSim TSP obteve o melhor resultado. Já o LARSim MKP, alcançou a pior nota entre os módulos, indicando a necessidade da realização de possíveis ajustes.

VII. COMPARAÇÃO COM OUTROS TRABALHOS

Nesta seção, é apresentado um estudo comparativo sobre a interface gráfica proposta. São discutidas cinco características: idioma, ambiente de desenvolvimento, algoritmos implementados, módulos e funcionalidades disponíveis. A Tabela II apresenta uma comparação da presente proposta com outros trabalhos que desenvolveram GUI's para o AR: I [24], II [23] e III [43].

Observando a Tabela II, percebe-se que a LARSim é a única (entre as interfaces analisadas) que se apresenta no idioma Português. Esse aspecto reforça um importante propósito do presente trabalho, o desenvolvimento de uma interface gráfica para facilitar a difusão dos conceitos de AR entre estudantes de instituições de ensino em língua portuguesa. Quanto ao ambiente de desenvolvimento, outros trabalhos também desenvolveram GUIs em MATLAB [24], [43]. No entanto, cada uma dessas propostas se difere quanto aos módulos disponíveis para o usuário. Nesse aspecto, destaca-se na LARSim a implementação das bibliotecas de instâncias (TSPLIB e ORLIB) de relevantes problemas de otimização combinatória: TSP e MKP. Além disso, é válido salientar que outros módulos poderão ser adicionados pelos autores e por outros pesquisadores/estudantes.

Em relação aos algoritmos implementados, todas as interfaces optaram por dois dos métodos mais tradicionais de AR

TABELA II
COMPARAÇÃO COM DIFERENTES INTERFACES GRÁFICAS DESENVOLVIDAS PARA O APRENDIZADO POR REFORÇO: I [24], II [23] E III [43]

		LARSim	I [24]	II [23]	III [43]
Idioma	Português	✓	–	–	–
	Inglês	–	✓	✓	✓
	MATLAB	✓	✓	–	✓
Ambiente	Visual Studio	–	–	✓	–
	Python	–	–	–	✓
Algoritmos	*Q-learning*	✓	✓	✓	✓
	SARSA	✓	✓	✓	✓
	TSP	✓	–	–	–
	MKP	✓	–	–	–
	Navegação	✓	✓	–	–
Módulos	Sistema de Reservatórios	–	–	✓	–
	MCP	–	–	–	✓
	CPCP	–	–	–	✓
	Manipuladores Robóticos	–	–	–	✓
	Definir Parâmetros	✓	✓	✓	–
Funções	Gráficos	✓	✓	✓	–
	Relatórios	✓	✓	✓	–
	Seleção de Instâncias	✓	–	–	–

(*Q-learning* e SARSA), evidenciado a importância do aprendizado dessas técnicas para o aluno. Por fim, é interessante evidenciar as funcionalidades na interface proposta, como definição dos parâmetros de aprendizado, geração de gráficos e relatórios. Nesse sentido, a ferramenta proposta permite ao usuário analisar por meio de *plots* e arquivos de texto como os parâmetros do AR podem ser determinantes no desempenho do aprendizado [14]. Além disso, a LARSim permite em todos os módulos a seleção de várias instâncias (TSPLIB e ORLIB) ou mapas de navegação. Dessa forma, o usuário tem disponível diferentes ambientes para a realização de experimentos.

VIII. CONCLUSÃO

O objetivo deste trabalho foi apresentar uma interface gráfica interativa para fins educacionais de Aprendizado por Reforço. O ambiente proposto permite ao usuário definir condições de aprendizado, como algoritmos e parâmetros, e aplicar em domínios conhecidos: Problema do Caixeiro Viajante, Problema da Mochila Multidimensional e navegação de um agente. Nesse sentido, o LARSim visa auxiliar e facilitar estudantes de graduação e pós-graduação no aprendizado de conceitos do AR.

Em trabalhos futuros, espera-se a criação de novos módulos. Nesse sentido, o código completo do LARSim em MATLAB será disponibilizado em domínio público na internet, de forma a facilitar a difusão do simulador e também o desenvolvimento de módulos por outros pesquisadores e estudantes. Também almeja-se a inclusão de outros algoritmos de AR e outras políticas de seleção de ações [1] no código da interface.

TABELA III
LISTA DE ABREVIAÇÕES

AR	Aprendizado por Reforço
CPCP	*Cart Pole Control Problem*
GUI	*Graphical User Interface*
LARSim	Laboratório de Aprendizado por Reforço Simulado
MATLAB	*MATrix LABoratory*
MCP	*Mountain Car Problem*
MDP	Processos de Decisão de Markov
MKP	Problema da Mochila Multidimensional
ORLIB	*Operations Research Library*
TSP	Problema do Caixeiro Viajante
TSPLIB	*Traveling Salesman Problem Library*

Além disso, será proposta uma versão da interface gráfica do LARSim na linguagem R para facilitar a utilização de métodos estatísticos (Metodologia de Superfície de Resposta [14], Regressão Logística [11]) na análise de resultados do AR nos domínios propostos.

APÉNDICE

A Tabela III apresenta a lista das abreviações utilizadas.

AGRADECIMENTOS

FAPEMIG, CAPES, CNPq/INERGE, UFRB, UFSJ (Edital 001/2019/Reitoria).

REFERÊNCIAS

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA: MIT Press, 2018.
[2] C. J. Watkins and P. Dayan, "Technical note Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992.
[3] L. Kaelbling, M. Littman, and A. Moore, "Reinforcement learning: A survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.
[4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.
[5] S. J. Russell and P. Norvig, *Artificial Intelligence*. Campus, 3st ed., 2013.
[6] J. Morimoto, "Foraging decisions as multi-armed bandit problems: Applying reinforcement learning algorithms to foraging data," *Journal of theoretical biology*, vol. 467, pp. 48–56, 2019.
[7] J. J. Valletta, C. Torney, M. Kings, A. Thornton, and J. Madden, "Applications of machine learning in animal behaviour studies," *Animal Behaviour*, vol. 124, pp. 203–220, 2017.
[8] G. Wang, "Machine learning for inferring animal behavior from location and movement data," *Ecological Informatics*, vol. 49, pp. 69–76, 2019.
[9] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement Learning in Robotics: A Survey," *International Journal of Robotics Research*, vol. July, 2013.
[10] A. H. P. Selvatici and A. H. R. Costa, "Aprendizado da coordenação de comportamentos primitivos para robôs móveis," *Revista Controle & Automação*, vol. 18, no. 2, pp. 173–186, 2007.
[11] A. L. C. Ottoni, E. G. Nepomuceno, M. S. Oliveira, L. T. Cordeiro, and R. D. Lamperti, "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, 2016.
[12] L. M. Gambardella and M. Dorigo, "Ant-Q: A reinforcement learning approach to the traveling salesman problem," *Proceedings of the 12th International Conference on Machine Learning*, pp. 252–260, 1995.

[13] M. L. Costa, C. A. A. Padilha, J. D. Melo, and A. D. D. Neto, "Hierarchical reinforcement learning and parallel computing applied to the k-server problem," *IEEE Latin America Transactions*, vol. 14, no. 10, pp. 4351–4357, Oct 2016.
[14] A. L. C. Ottoni, E. G. Nepomuceno, and M. S. de Oliveira, "A response surface model approach to parameter estimation of reinforcement learning for the travelling salesman problem," *Journal of Control, Automation and Electrical Systems*, vol. 29, no. 3, pp. 350–359, 2018.
[15] A. L. C. Ottoni, E. G. Nepomuceno, M. S. de Oliveira, and D. C. R. de Oliveira, "Tuning of reinforcement learning parameters applied to sop using the scott–knott method," *Soft Computing*, pp. 1–13, Jul 2019.
[16] A. L. C. Ottoni, E. G. Nepomuceno, L. T. Cordeiro, R. D. Lamperti, and M. S. Oliveira, "Análise do Desempenho do Aprendizado por Reforço na Solução do Problema do Caixeiro Viajante," *XII SBAI - Simpósio Brasileiro de Automação Inteligente*, pp. 43–48, 2015.
[17] 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, 2017.
[18] A. Geramifard, C. Dann, R. H. Klein, W. Dabney, and J. P. How, "Rlpy: a value-function-based reinforcement learning framework for education and research." *Journal of Machine Learning Research*, vol. 16, pp. 1573–1578, 2015.
[19] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*. IEEE, 2016, pp. 1–8.
[20] J.-S. Chou and J. P. P. Thedja, "Metaheuristic optimization within machine learning-based classification system for early warnings related to geotechnical problems," *Automation in Construction*, vol. 68, pp. 65–80, 2016.
[21] J. C. Stålring, L. A. Carlsson, P. Almeida, and S. Boyer, "Azorange-high performance open source machine learning for qsar modeling in a graphical programming environment," *Journal of cheminformatics*, vol. 3, no. 1, p. 28, 2011.
[22] S. Gould, "Darwin: A framework for machine learning and computer vision research and development," *Journal of Machine Learning Research*, vol. 13, no. Dec, pp. 3533–3537, 2012.
[23] J. D. Rieker and J. W. Labadie, "An intelligent agent for optimal river-reservoir system management," *Water Resources Research*, vol. 48, no. 9, 2012.
[24] N. ALTUNTAŞ, E. Imal, N. Emanet, and C. N. Öztürk, "Reinforcement learning-based mobile robot navigation," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 24, no. 3, pp. 1747–1767, 2016.
[25] B. Blumberg, M. Downie, Y. Ivanov, M. Berlin, M. P. Johnson, and B. Tomlinson, "Integrated learning for interactive synthetic characters," *ACM transactions on graphics (TOG)*, vol. 21, no. 3, pp. 417–426, 2002.
[26] S. M. Campo, R. J. Bermudez, F. G. Serna, M. O. Oliveira, and O. E. Perrone, "Simulation and analysis of the radiation pattern of microstrip patch-type antenna," *IEEE Latin America Transactions*, vol. 11, no. 1, pp. 341–346, Feb 2013.
[27] J. H. Canossa, A. B. Neto, D. A. Alves, F. F. Putti, and L. R. A. G. Filho, "Development of an interactive program to study of the continuation power flow," *IEEE Latin America Transactions*, vol. 16, no. 4, pp. 1227–1235, April 2018.
[28] L. H. R. Jesus and L. C. Brito, "Interactive evolution strategies for minimizing single-objective functions," *IEEE Latin America Transactions*, vol. 15, no. 5, pp. 981–987, May 2017.
[29] A. Castillo, J. Ortegon, J. Vazquez, and J. Rivera, "Virtual laboratory for digital image processing," *IEEE Latin America Transactions*, vol. 12, no. 6, pp. 1176–1181, Sept 2014.
[30] C. A. G. Gutiérrez, J. R. Reséndiz, J. D. M. Santibáñez, and G. M. Bobadilla, "A model and simulation of a five-degree-of-freedom robotic arm for mechatronic courses," *IEEE Latin America Transactions*, vol. 12, no. 2, pp. 78–86, March 2014.
[31] D. B. Góes, G. M. da Silva, D. L. Guedes, and O. F. Silva, "Desenvolvimento de interface em matlab para aprendizado e comparação de métodos numéricos," *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, vol. 3, no. 1, 2015.
[32] V. C. Mariani, T. M. Preto, and A. L. P. Guedes, "Utilização do maple, matlab e scilab nos cursos de engenharia," in *Congresso Brasileiro de Ensino de Engenharia*, vol. 9, 2005.
[33] P. H. Silva, E. G. Nepomuceno, A. Vitorino, and S. A. Martins, "A visual chaotic system simulation in arduino platform controlled by android app," in *World Engineering Education Conference (EDUNINE), IEEE*. IEEE, 2017, pp. 62–66.
[34] P. H. O. Silva, L. G. Nardo, S. A. M. Martins, E. G. Nepomuceno, and M. Perc, "Graphical interface as a teaching aid for nonlinear dynamical systems," *European Journal of Physics*, vol. 39, no. 6, p. 065105, 2018.
[35] G. Reinelt, "TSPLIB - A Traveling Salesman Problem Library," *ORSA Journal on Computing*, vol. 3, no. 4, pp. 376–384, 1991.
[36] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa, "On the relation between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning," *1st International Workshop on Hybrid Control of Autonomous System*, pp. 49–55, 2009.
[37] F. C. Lima Júnior, A. D. D. Neto, and J. D. Melo, *Traveling Salesman Problem, Theory and Applications*. InTech, 2010, ch. Hybrid Metaheuristics Using Reinforcement Learning Applied to Salesman Traveling Problem, pp. 213–236.
[38] L. Bodin, B. Golden, A. Assad, and M. Ball, "Routing and Scheduling of Vehicles and Crews – The State of the Art," *Computers and Operations Research*, vol. 10, no. 2, p. 63–211, 1983.
[39] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of heuristics*, vol. 4, no. 1, pp. 63–86, 1998.
[40] V. Grassi Jr and J. Okamoto Jr, *Robótica Móvel*. LTC, 2014, ch. Arquiteturas de controle: tipos e conceitos, pp. 46–60.
[41] R. Brooks, "A robust layered control system for a mobile robot," *IEEE journal on robotics and automation*, vol. 2, no. 1, pp. 14–23, 1986.
[42] J. E. Beasley, "Or-library: distributing test problems by electronic mail," *Journal of the operational research society*, vol. 41, no. 11, pp. 1069–1072, 1990.
[43] J. A. Martin, *Software Tools for Reinforcement Learning, Artificial Neural Networks and Robotics (Matlab and Python)*, 2010, acesso em Junho/2019. [Online]. Available: https://jamh-web.appspot.com/download.htm

André Luiz Carvalho Ottoni possui graduação (2015) e mestrado (2016) em Engenharia Elétrica pela Universidade Federal de São João del-Rei (UFSJ). Atualmente é professor Assistente no Centro de Ciências Exatas e Tecnológicas (CETEC) da Universidade Federal do Recôncavo da Bahia (UFRB). É pesquisador no Grupo de Controle e Modelagem (GCOM) da UFSJ e membro da Sociedade Brasileira de Automática (SBA). Tópicos de pesquisa: Inteligência Artificial, Aprendizado por Reforço, Otimização Combinatória e Robótica Inteligente.

Erivelton Geraldo Nepomuceno Erivelton G. Nepomuceno received the BSc degree from UFSJ and the PhD in Electrical Engineering from the UFMG (Brazil). He is currently Associate Professor at UFSJ and leader of Control and Modelling Group (GCOM). Editorial experience: Associate Editor (2017-2019) and Steering Committe (2019-) of the IEEE LATAM; J. of Biomedical Research and Reviews and J. of Computer Science Research. Membership: IEEE CAS; Technical Committee on Nonlinear Circuits and Systems - IEEE CAS; Brazilian Association of Automatica.

Marcos Santos de Oliveira received the BSc degree from UNESP, MSc from USP and the Phd in Statistics from UFLA (Brazil). He is currently Associate Professor in the Department of Mathematics and Statistics at the Federal University of São João del-Rei (UFSJ). His research interests include applied statistics and probability, regression analysis, machine learning and statistics education.