DOI: 10.48011/asba.v2i1.1658

AutoRL-TSP: Sistema de Aprendizado por Reforço Automatizado para o Problema do Caixeiro Viajante

Lara T. Cordeiro Ottoni [*] **André Luiz C. Ottoni** [**]
Marcos S. de Oliveira [***] **Erivelton G. Nepomuceno** [****]

[*] *Faculdade Maria Milza, Governador Mangabeira, BA, Brasil (e-mail: laratoledocordeiro@gmail.com)*
[**] *Centro de Ciências Exatas e Tecnológicas, Universidade Federal do Recôncavo da Bahia, Cruz das Almas, BA, Brasil (e-mail: andre.ottoni@ufrb.edu.br)*
[***] *Departamento de Matemática e Estatística, Universidade Federal de São João del-Rei, São João del-Rei, MG, Brasil (e-mail: mso@ufsj.edu.br)*
[****] *Grupo de Controle e Modelagem, Departamento de Engenharia Elétrica, Universidade Federal de São João del-Rei, São João del-Rei, MG, Brasil (e-mail: nepomuceno@ufsj.edu.br)*

Abstract: The AutoML (Automated Machine Learning) aims at developing techniques to automate the entire machine learning process to obtain a system that fits the problem conditions. In this sense, one of the relevant aspects of AutoML systems is the definition of the initial simulation configurations, which can considerably influence the final learning result. It has been a topic of great interest; yet the literature lacks studies that present AutoML methods for Reinforcement Learning (RL) applications in combinatorial optimization problems. Thus, the objective of this work is to propose an Automated RL system (AutoRL-TSP) for automatic adjustment of RL parameters for application in the Traveling Salesman Problem.The modules of this method are composed by: reinforcement learning system, knowledge base, optimization and recommendation structures. The results show the parameters according to the instance (TSPLIB) and the simulated optimization method. In general, by adopting the knowledge base and the Variable Neighborhood Search (VNS) algorithm, the AutoRL-TSP system achieved the best results.

Resumo: O AutoML (Aprendizado de Máquina Automatizado) tem como objetivo desenvolver técnicas para automatizar todo o processo de aprendizagem de máquina, de forma a obter um sistema que se adeque as condições do problema. Nesse sentido, um dos aspectos relevantes de sistemas de AutoML é a definição das configurações iniciais da simulação, que podem influenciar consideravelmente no resultado final do aprendizado. No entanto, a literatura carece de trabalhos que apresentem métodos de AutoML para aplicações de Aprendizado por Reforço (AR) em problemas de otimização combinatória. Dessa forma, o objetivo deste trabalho é propor um sistema de AR Automatizado (AutoRL-TSP) para ajuste automático de parâmetros do AR para aplicação no Problema do Caixeiro Viajante. Para isso, são apresentados os módulos do AutoRL-TSP: sistema de AR, base de conhecimento, estruturas de otimização e recomendação. Os resultados apresentam os *rankings* de recomendação de parâmetros de acordo com a instância (TSPLIB) e o método de otimização simulado. Em geral, ao adotar a base de conhecimento e o algoritmo *Variable Neighborhood Search*, o AutoRL-TSP alcançou os melhores resultados.

Keywords: AutoML; Reinforcement Learning; Travelling Salesman Problem.
Palavras-chaves: AutoML; Aprendizado por Reforço; Problema do Caixeiro Viajante.

1. INTRODUÇÃO

O Aprendizado de Máquina, em inglês, *Machine Learning (ML)*, é um importante campo da Inteligência Artificial (Brazdil et al., 2009; Russell and Norving, 2013; Sutton and Barto, 2018; Hutter et al., 2019). De fato, diversas são as áreas de estudo do ML, com destaque para controle de processos, classificação de padrões, clusterização , sistemas de previsão e otimização (Russell and Norving, 2013; Silva et al., 2016).

Um dos aspectos relevantes de sistemas de aprendizado é a definição das configurações iniciais da simulação (parâmetros, algoritmos, arquitetura de uma rede neural) (Brazdil et al., 2009; Hutter et al., 2019). Um exemplo é o ajuste de parâmetros (ou hiperparâmetros) que podem influenciar consideravelmente no resultado final do aprendizado (Schweighofer and Doya, 2003; Even-Dar and Mansour, 2003; Hutter et al., 2019; Ottoni et al., 2020). Nesse sentido, técnicas de *Automated Machine Learning (AutoML)* vêm sendo propostas como forma de trabalhar o problema do aprendizado de máquina de ponta-a-ponta (Feurer et al., 2015; Hutter et al., 2019; Tsiakmaki et al., 2019; Mantovani et al., 2019; Cai et al., 2020).

O objetivo da área de AutoML é desenvolver técnicas para automatizar todo o processo de aprendizagem de máquina, de forma a obter um sistema que se adeque as condições do problema (Hutter et al., 2019). Nesse sentido, alguns importantes tópicos de AutoML são: utilização de bases de conhecimento (Brazdil et al., 2009), transferência de aprendizado entre aplicações (Taylor and Stone, 2009; Da Silva and Reali Costa, 2019) e desenvolvimento de sistemas de recomendação/otimização de parâmetros (Brazdil et al., 2009; Cunha et al., 2018; Hutter et al., 2019).

Outra vertente do auto aprendizado de máquina é o AutoRL (*Automated Reinforcement Leaning*) (Chiang et al., 2019; Faust et al., 2019), no qual, é destinado para sistemas de Aprendizado por Reforço (AR). O AR é uma área do ML que se baseia no aprendizado a partir de recompensas e na interação de um agente inteligente com o ambiente (Watkins and Dayan, 1992; Taylor and Stone, 2009; Sutton and Barto, 2018). Uma importante área de estudo do AR é a aplicação em problemas combinatórios, como Problema do Caixeiro Viajante (Gambardella and Dorigo, 1995; Lima Júnior et al., 2010), K-Servos (Lins et al., 2019) e *Sequential Ordering Problem* (Ottoni et al., 2020). No entanto, a literatura carece de trabalhos que apresentam sistemas de AutoRL para a otimização combinatória.

Dessa forma, o objetivo deste trabalho é propor um sistema de AutoRL para ajuste automático de parâmetros do AR para aplicação em um relevante domínio de estudo da otimização combinatória: Problema do Caixeiro Viajante, em inglês, *Travelling Salesman Problem (TSP)* (Gambardella and Dorigo, 1995; Lima Júnior et al., 2010; Ottoni et al., 2018). Para isso, são apresentados os módulos desse método, divididos em: sistema de AR, base de conhecimento, estruturas de otimização e recomendação. Além disso, é importante ressaltar que o algoritmo de AutoRL proposto utiliza do método *Variable Neighborhood Search (VNS)* (Hansen and Mladenović, 2001) para o ajuste dos parâmetros do algoritmo SARSA (Sutton and Barto, 2018).

Este artigo está organizado em seções. Na seção 2, são definidos aspectos teóricos do TSP, AR, AutoML e VNS. Já a seção 3, detalha o método de AutoML proposto. Por fim, os resultados e as conclusões do trabalho são apresentados, respectivamente, nas seções 4 e 5.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Problema do Caixeiro Viajante

O objetivo do Problema do Caixeiro Viajante (TSP - *Travelling Salesman Problem* (Gambardella and Dorigo, 1995; Lima Júnior et al., 2010) é minimizar a rota entre um conjunto de localidades. Além disso, o agente deve visitar cada cidade uma única vez e voltar ao nó inicial ao término do percurso. O TSP pode ser formulado como um grafo com um conjunto de nós e arcos, sendo que c_{ij} é o custo dado para cada aresta (i, j) (Bodin et al., 1983; Lima Júnior et al., 2010).

Uma possível formulação matemática para o TSP (Bodin et al., 1983) é apresentada nas Eqs. (1) à (5):

$$Min \sum_{i=1}^{N} \sum_{j=1}^{N} c_{ij} x_{ij}, \tag{1}$$

sujeito à:

$$\sum_{i=1}^{N} x_{ij} = 1 \qquad (\forall j = 1,..., N), \tag{2}$$

$$\sum_{j=1}^{N} x_{ij} = 1 \qquad (\forall i = 1,..., N), \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad (\forall i, j = 1,..., N), \tag{4}$$

$$X = x_{ij} \in S \qquad (\forall i, j = 1,..., N), \tag{5}$$

em que, N é o conjunto de nós. A Eq. (1) representa o objetivo de minimizar a distância na rota. Nesse sentido, c_{ij} é o custo entre as cidades (i e j) e $x_{i,j}$ é a variável decisão. As Eqs. (2) e (3) garantem que cada nó será visitado uma única vez. Já a Eq. (4) define x_{ij} como binária. Na Eq. (5) S representa qualquer conjunto de restrições que eliminam a formação de sub-rotas.

Neste trabalho, os experimentos foram realizados com dados da TSPLIB (*Travelling Salesman Problem Library*) [1] (Reinelt, 1991). A TSPLIB é uma biblioteca de instâncias para o TSP Simétrico e Assimétrico (ATSP) e outros domínios de otimização combinatória, como: *Sequential Ordering Problem* e Problema de Roteamento de Veículos. Nesse sentido, esse repositório de dados é frequentemente abordado na literatura (Lima Júnior et al., 2010; Alipour et al., 2018; Ottoni et al., 2020).

2.2 Aprendizado por Reforço

O AR é uma técnica de *Machine Learning* que se baseia no aprendizado a partir do sucesso e fracasso em uma tarefa (Russell and Norving, 2013; Sutton and Barto, 2018). Basicamente, o agente aprendiz observa o estado (s) do ambiente, seleciona/executa uma ação (a), recebe um reforço pelo par (s, a) e atualiza uma matriz de aprendizado Q (Sutton and Barto, 2018). O objetivo é

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/

aprender uma política (π) que maximize o retorno pela tomada de decisão. Nesse aspecto, as técnicas de AR são fundamentadas nos Processos de Decisão de Markov (Russell and Norving, 2013).

Um dos algoritmos de AR mais utilizados na literatura é o SARSA (Sutton and Barto, 2018). A Eq. (6) apresenta o método de atualização da matriz Q pelo Algoritmo SARSA (Sutton and Barto, 2018):

$$Q_{t+1} = Q_t(s,a) + \alpha[r(s,a) + \gamma Q_t(s',a') - Q_t(s,a)], \quad (6)$$

em que, s é o estado e a a ação no instante t; $r(s,a)$ é a recompensa pela execução de a em s; s' é o novo estado e a' é a nova ação selecionada; Q_t e Q_{t+1} são matrizes no instante atual e em $t+1$, respectivamente; α é a taxa de aprendizado; γ é o fator de desconto. O Algoritmo 1 representa o SARSA.

```
1.  Para cada (s,a) inicialize Qt(s,a)=0;
2.  Observe o estado s;
3.  Selecione a ação a usando a política e-greedy;
4.  Repita até o critério de parada ser satisfeito
5.      Execute a ação a;
6.      Receba a recompensa imediata  r(s,a);
7.      Observe o novo estado s';
8.      Selecione a nova ação a' usando e-greedy;
9.      Atualize Qt+1 de acordo com Eq.(6);
10.     s = s';
11.     a = a';
12. Fim Repita
```

Algoritmo 1: SARSA.

No Algoritmo 1 é adotada a política de seleção de ações $\epsilon - greedy$ (Sutton and Barto, 2018). Esse método utiliza o parâmetro ϵ no controle entre gula e aleatoriedade na tomada de decisão. Neste trabalho, ϵ foi fixado em 0,01.

Outros dois parâmetros do SARSA são taxa de aprendizado (α) e fator de desconto (γ). A taxa de aprendizado é responsável por controlar a velocidade que as novas informações sobrepõem o aprendizado acumulado até o instante t, conforme Eq. (6). Por outro lado, o fator de desconto regula o pesos entre o reforço imediato ($r(s,a)$) e as recompensas futuras (representada por $Q_t(s',a')$). Esses dois parâmetros podem ser definidos em qualquer valor entre 0 e 1. Conforme demonstrado por Ottoni et al. (2018), o ajuste desses parâmetros pode influenciar significativamente nas soluções do AR para o TSP.

2.3 AutoML

O Aprendizado de Máquina Automatizado, em inglês, *Automated Machine Learning* (AutoML) é uma área que tem como objetivo automatizar o processo de aprendizagem de máquina (Feurer et al., 2015; Hutter et al., 2019; Cai et al., 2020). Para isso, o campo de pesquisa de AutoML reúne trabalhos em diferentes vertentes, como: *meta-learning* (Brazdil et al., 2009; Cunha et al., 2018), sistemas de recomendação (Mantovani et al., 2015, 2019), otimização (Hutter et al., 2019) e transferência de aprendizado (Da Silva and Reali Costa, 2019).

Um dos desafios do AutoML é definir parâmetros para os algoritmos de aprendizado de máquina de acordo com o conjunto de dados analisados (Hutter et al., 2014, 2019). De fato, um algoritmo de ML pode possuir um espaço de configurações de parâmetros complexo, sendo alguns contínuos e outros categóricos, por exemplo (Hutter et al., 2019). Dessa forma, de acordo com Hutter et al. (2019), dois importantes objetivos da otimização de parâmetros consistem em reduzir o esforço para aplicar ML e melhorar o desempenho de algoritmos.

Nesse sentido, no AutoML existem três estratégias básicas para a definição de parâmetros de algoritmos de ML: otimização, recomendação e híbrida (Mantovani et al., 2015; Hutter et al., 2019). Em uma estrutura de otimização são utilizadas técnicas para ajuste de parâmetros, como *Support Vector Machines* (Mantovani et al., 2015), métodos Bayesianos (Hutter et al., 2019), Metodologia de Superfície de Resposta (Ottoni et al., 2018). Já em um sistema de recomendação são avaliadas configurações de parâmetros dentre as já existentes. Em seguida, podem ser construídos *rankings* de recomendação de acordo com as medidas de avaliação (Brazdil et al., 2009; Mantovani et al., 2019). Por fim, os sistemas híbridos envolvem características de otimização e recomendação.

Vale ressaltar ainda que a otimização e recomendação de parâmetros também vem sendo objeto de diversos trabalhos na área do AR. Alguns exemplos são: (Schweighofer and Doya, 2003; Even-Dar and Mansour, 2003; McAuley et al., 2012; Ottoni et al., 2018; Cardenoso Fernandez and Caarls, 2018; Ottoni et al., 2020).

2.4 VNS

O *Variable Neighborhood Search* (VNS), proposto por Mladenović and Hansen (1997), é uma metaheurística que explora o espaço de soluções realizando uma sequência de buscas locais por meio de trocas sistemáticas entre as estruturas de vizinhança pré-ordenadas. A ideia principal do VNS, apresentado no Algoritmo 2, é efetuar buscas no espaço de solução com duas ou mais estruturas de vizinhança, o que permite escapar de ótimos locais (Liao and Cheng, 2007; Liu and Zhou, 2013; Cordeiro and Batista, 2018).

```
1.  Solução inicial s;
2.  Repita até o critério de parada ser satisfeito:
3.      k=1;
4.      Repita até finalizar as estruturas de vizinhança
5.          s'= Gere um vizinho qualquer;
6.          s"= BuscaLocal(s');
7.          Se f(s") é melhor que f(s):
8.              s = s";
9.              k = 1';
10.         Senão
11.             k=k+1;
12.         Fim Se
13.     Fim Repita
14. Fim Repita
```

Algoritmo 2: *Variable Neighborhood Search* (VNS).

Uma convergência adequada das metaheurísticas baseadas em vizinhança depende de uma solução inicial de boa qualidade e das estruturas de vizinhanças utilizadas (Coelho, 2016). Neste sentido, os componentes do algoritmo VNS proposto neste trabalho são descritos na Seção 3.3.

A busca local no VNS foi implementada utilizando uma heurística de refinamento. Neste trabalho é usada a heurística *Variable Neighborhood Descent* (VND) (Mladenović and Hansen, 1997). Na heurística VND, o espaço de soluções é explorado por meio de trocas de estrutura de vizinhança, sendo aceitas somente soluções que apresentam melhoria em relação à solução atual. Utilizou-se a heurística de refinamento *First Improvement*, ou método de primeira melhora. O pseudocódigo do VND é apresentado no Algoritmo 3.

```
1.  solução inicial = s
2.  k=1;
3.  Repita até finalizar as estruturas de vizinhança
4.      s'= Gere um vizinho qualquer;
5.      Se f(s) é melhor que f(s'):
6.          s = s';
7.          k = 1;
8.      Senão
9.          k=k+1;
10.     Fim Se
11. Fim Repita
```

Algoritmo 3: *Variable Neighborhood Descent* (VND) .

3. SISTEMA DE APRENDIZADO POR REFORÇO AUTOMATIZADO

3.1 Modelo de Aprendizado por Reforço

O modelo de AR foi definido em termos de um conjunto de estados, ações e recompensas. Neste trabalho, essa estrutura foi baseada em trabalhos que aplicaram o AR em problemas de otimização combinatória (Gambardella and Dorigo, 1995; Bianchi et al., 2009; Lima Júnior et al., 2010; Ottoni et al., 2018). O modelo adotado é apresentado em seguida:

Estados: conjunto de localidades que o caixeiro viajante deve visitar para a formar a rota. Nesse aspecto, o espaço de estados está intimamente ligado ao número de nós (N) da instância utilizada.

Ações: representam as intenções de movimentação entre as localidades (estados). Vale ressaltar que o número de ações disponíveis para o agente aprendiz variam de acordo com o passo de execução da rota. Isso porque, o caixeiro viajante não deve repetir cidades, com exceção da primeira localidade para finalizar a rota.

Recompensas: os reforços foram definidos a partir das distâncias entre os nós de uma instância. Quando o caixeiro viajante decide movimenta-se entre duas localidades (i e j), quanto maior a distância entre i e j (c_{ij}), mais negativa será a penalidade, conforme Eq. (7):

$$R = -c_{ij} \tag{7}$$

3.2 Base de Conhecimento

A base de conhecimento adotada utiliza dos parâmetros (α e γ) ajustados em trabalho anterior para instâncias do TSP e algoritmo SARSA (Ottoni et al., 2018). Ottoni et al. (2018) utilizam de Modelos de Superfície de Resposta (RSM) para estimar a taxa de aprendizado e o fator desconto na busca de minimizar a variável resposta (y - distância na rota). A Eq. (8) apresenta um exemplo de modelo ajustado em (Ottoni et al., 2018) para a instância kroA200:

$$y = 191{,}3 - 237{,}0\alpha - 75{,}7\gamma + 147{,}3\alpha^2 + 14{,}8\gamma^2 + 62{,}8\alpha\gamma \tag{8}$$

A Figura 1 representa em duas dimensões a relação entre os parâmetros (α e γ) e a distância percorrida na rota para o modelo da Equação (8). A região em vermelho no gráfico indica o conjunto de pontos que tendem minimizar a variável resposta (distância na rota) no modelo RSM.

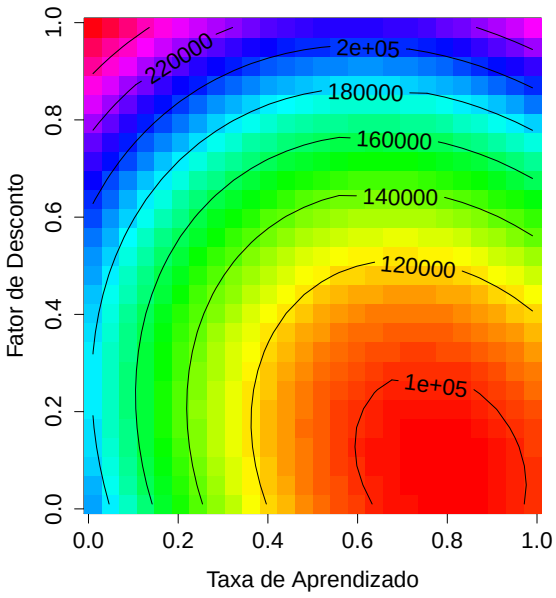

Figura 1. Representação em gráfico de contornos para o modelo de superfície de resposta da instância kroA200 e algoritmo SARSA (Ottoni et al., 2018).

Dessa forma, foi adotado como base de conhecimento os parâmetros ajustados (ponto mínimo da curva) via RSM (Ottoni et al., 2018) para seis instâncias da TSPLIB: berlin52, kroA100, kroA200, ftv33, ftv44 e ftv64. A Tabela 1 apresenta essas instâncias TSP com seus respectivos, tipos, número de nós, valor ótimo (menor solução possível para a rota, de acordo com a TSPLIB), e parâmetros (α e γ).

Tabela 1. Problemas da TSPLIB adotados e parâmetros de Ottoni et al. (2018).

Tipo	**Problema**	**Nós**	**Ótimo**	α	γ
TSP	berlin52	52	7.542	0,7421	0,0729
	kroA100	100	21.282	0,7782	0,0626
	kroA200	200	29.368	0,7854	0,0894
ATSP	ftv33	34	1.286	0,7074	0,1450
	ftv44	45	1.613	0,7491	0,0882
	ftv64	65	1.839	0,7879	0,0615

Os parâmetros da base de conhecimento (Tabela 1) foram utilizados na estrutura de otimização proposta, como detalhado na próxima seção.

3.3 Estrutura de Otimização

A estrutura de otimização do sistema de AutoML proposto adotou quatro métodos para buscar a melhor combinação entre a taxa de aprendizado (α) e o fator de desconto (γ). Conforme destacado em Ottoni et al. (2018), a combinação de (α, γ) gera fortes impactos no resultado da otimização do TSP. Os quatro métodos utilizados foram: Rand, Base, R-VNS e B-VNS.

Em seguida, esses métodos de otimização são explicados.

Rand: no método aleatório, os valores de (α) e (γ) são gerados aleatoriamente (entre 0 e 1) utilizando uma função aleatória. Foram gerados três conjuntos de parâmetros (α, γ). Em seguida, essas combinações foram passados para o sistema de AR, no qual simula o TSP para cada (α, γ) utilizando de 10 mil episódios. Os valores obtidos de distância da rota são armazenados e disponibilizados para a próxima etapa: Recomendação.

Base: neste método, são realizadas simulações com a base de conhecimento obtida em Ottoni et al. (2018) e apresentada na seção 3.2. De acordo com a instância a ser executada pelo AR são adotados os parâmetros (α,γ) apresentados na Tabela 1. O AR é executado três vezes para os mesmos valores de α e γ, obtendo três valores de distância da rota. Essas soluções também são armazenadas para a próxima etapa do AutoML.

R-VNS e B-VNS: estes métodos utilizam do VNS para realizar uma busca local na região dos parâmetros definidos por uma estrutura Rand ou Base, respectivamente. Nesse sentido, o Algoritmo 2 foi implementando adotando como uma solução inicial parâmetros gerados aletoariamente ou pela base de conhecimento.

A solução inicial para o método R-VNS foi a partir de parâmetros definidos pelo método Rand. O VNS realiza uma busca local explorando a vizinhança da combinação aleatória. Já como solução inicial do B-VNS foram utilizados os valores de parâmetros da base de conhecimento (Tabela 1). Ambos os métodos são executados em 3 repetições para cada instância.

Elaborou-se um conjunto de estruturas de vizinhanças específico para o problema de ajuste de parâmetros, no qual deseja-se explorar um espaço nas proximidades da solução inicial. Foram propostos duas estruturas de vizinhanças, adotadas nos Algoritmos 2 e 3:

1. Acréscimo ou decréscimo de 0,01 no valor de (α) e (γ). A soma ou subtração são definidas de forma aleatória com 50% de chance e verificada a restrição dos limites de (α) e (γ) (valores entre 0 e 1).
2. Acréscimo ou decréscimo de 0,05 no valor de (α) e (γ). A soma ou subtração é definido de forma aleatória com 50% de chance e também é verificado a restrição dos limites dos parâmetros.

3.4 Estrutura de Recomendação

Após a etapa de otimização é iniciada a fase de recomendação (Brazdil et al., 2009; Hutter et al., 2019) no sistema de AutoML proposto. Nesta estrutura, foram gerados *rankings* dos parâmetros (α e γ) de acordo com os resultados da medida de avaliação (distância mínima na rota) obtidos pelos métodos apresentados na Seção 3.3.

Para isso, foram recomendados parâmetros de acordo com o conjunto de dados analisado (instância) (Mantovani et al., 2019). Nesse sentindo, foram gerados 6 *rankings* (1 por problema TSPLIB) com 12 posições. Os resultados foram ordenados da melhor solução (1º lugar) até a maior distância (12º lugar).

3.5 Algoritmo AutoRL-TSP

A sequência de execução do sistema de AR automatizado para TSP é apresentada no algoritmo AutoRL-TSP (Algoritmo 4). Nesse sentido, o AutoRL-TSP reúne as funcionalidades apresentadas nas seções anteriores (base de conhecimento, estruturas de otimização e recomendação) para a execução do modelo de AR proposto no TSP.

```
1.  Para cada instância, Faça
2.      Para cada método (varia o índice i), Faça
3.         Repita até completar o número de épocas
4.            Se i < 3
5.               Gera os parâmetros de forma aleatória;
6.            Senão
7.               Busca parâmetros na base de conhecimento;
8.            Fim se
9.            Escolha método (i)
10.              Caso 1: Executa o método Rand;
11.              Caso 2: Executa o método R-VNS;
12.              Caso 3: Executa o método Base;
13.              Caso 4: Executa o método B-VNS;
14.           Fim Escolha
15.        Fim Repita
16.     Fim Para
17.     Realiza recomendação por ranking da instância;
18.     Salva ranking da instância;
19.  Fim Para
```

Algoritmo 4: AutoRL-TSP.

O algoritmo proposto foi configurado para executar no mínimo 10 mil episódios em 3 épocas (repetições) para condição da estrutura de otimização. Um episódio equivale ao agente realizar uma rota completa. Já uma época representa a sequência de aprendizado ao longo dos 10 mil episódios. Vale ressaltar que, ao executar as condições que adotam o VNS, o AutoRL-TSP define a quantidade de simulações (épocas) necessárias de acordo com o problema, conforme descrito na Seção 3.3. Além disso, como resultado final das simulações são gerados *rankings* de recomendação com os parâmetros e os respectivos métodos de otimização.

4. RESULTADOS

Os resultados obtidos pelo algoritmo AutoML proposto (AutoRL-TSP) descrito na Seção 3.5 para as instâncias da TSPLIB adotadas são apresentados em duas subseções. Inicialmente, apresenta-se os *Rankings* de Recomendação para cada instância. Em seguida, realiza-se uma análise dos resultados obtidos pelos métodos de otimização dos parâmetros utilizando a técnica de Análise de Variância (ANOVA).

4.1 Rankings de Recomendação

Os *Rankings* de Recomendação obtidos pelo método proposto (AutoRL-TSP) para as instâncias berlin52, kroA100, kroA200, ftv33, ftv44 e ftv64 são apresentados nas Tabelas de 2 a 7. A medida de avaliação é a menor distância (solução mínima) em uma rota ao longo dos episódios de uma época.

Tabela 2. Resultados para *ranking* de recomendação da instância berlin52.

Ranking	Método	α	γ	Distância
1º	B-VNS	0,7421	0,0729	7965,31
2º	B-VNS	0,7721	0,0629	8007,75
3º	B-VNS	0,7821	0,0529	8032,96
4º	Rand	0,7074	0,0369	8157,15
5º	Base	0,7421	0,0729	8242,04
6º	Base	0,7421	0,0729	8310,85
7º	Base	0,7421	0,0729	8433,54
8º	R-VNS	0,1455	0,3561	8747,51
9º	R-VNS	0,2575	0,4552	8811,83
10º	Rand	0,1113	0,8260	9375,33
11º	Rand	0,2383	0,9343	9504,58
12º	R-VNS	0,4637	0,6793	9566,66

Tabela 3. Resultados para o *ranking* de recomendação da instância kroA100.

Ranking	Método	α	γ	Distância
1º	B-VNS	0,7882	0,0526	23368,15
2º	B-VNS	0,7782	0,0626	23671,75
3º	R-VNS	0,6909	0,1086	23709,24
4º	B-VNS	0,7882	0,0726	23725,07
5º	Base	0,7782	0,0626	23782,13
6º	Base	0,7782	0,0626	24087,41
7º	R-VNS	0,4479	0,0318	24310,84
8º	Base	0,7782	0,0626	24326,49
9º	R-VNS	0,4673	0,6195	35396,60
10º	Rand	0,0923	0,8776	42464,26
11º	Rand	0,5487	0,9888	56148,53
12º	Rand	0,8147	0,9058	56950,94

Tabela 4. Resultados para o *ranking* de recomendação da instância kroA200.

Ranking	Método	α	γ	Distância
1º	B-VNS	0,7954	0,0394	32075,72
2º	B-VNS	0,8354	0,0294	32446,65
3º	B-VNS	0,7954	0,0094	32652,69
4º	Base	0,7854	0,0894	34896,09
5º	Rand	0,6749	0,0920	35496,89
6º	Base	0,7854	0,0894	36105,04
7º	Base	0,7854	0,0894	36222,75
8º	R-VNS	0,0846	0,1900	37489,69
9º	Rand	0,6186	0,2047	39427,45
10º	R-VNS	0,4923	0,3236	41724,57
11º	R-VNS	1,0372	0,2973	43280,38
12º	Rand	0,0748	0,4594	44291,48

Como pode ser visto nas Tabelas de 2 a 7, alguns métodos obtiveram um maior destaque quanto ao ranqueamento realizado pelo AutoRL-TSP. Em todas as instâncias os métodos B-VNS e Base produziram pelo menos uma solução que configurou-se entre as seis primeiras posições dos *rankings*. O que pode ser justificado pelo fato dos

Tabela 5. Resultados para o *ranking* de recomendação da instância ftv33.

Ranking	Método	α	γ	Distância
1º	B-VNS	0,6674	0,1850	1380
2º	B-VNS	0,7074	0,1650	1381
3º	R-VNS	0,5898	0,2758	1382
3º	Base	0,7074	0,1450	1382
3º	B-VNS	0,6974	0,1350	1382
6º	Base	0,7074	0,1450	1415
7º	Base	0,7074	0,1450	1423
8º	R-VNS	0,1155	0,5196	1453
9º	Rand	0,4479	0,2886	1466
10º	R-VNS	0,4802	0,5407	1480
11º	Rand	0,4499	0,4177	1501
12º	Rand	0,7800	0,4136	1569

Tabela 6. Resultados para o *ranking* de recomendação da instância ftv44.

Ranking	Método	α	γ	Distância
1º	R-VNS	0,8961	0,0427	1766
1º	B-VNS	0,7091	0,0482	1766
3º	B-VNS	0,7791	0,1582	1793
4º	B-VNS	0,7491	0,0882	1812
5º	R-VNS	0,1571	0,9670	1862
6º	Base	0,7491	0,0882	1877
6º	Base	0,7491	0,0882	1877
8º	Base	0,7491	0,0882	1879
9º	Rand	0,0127	0,3411	2122
10º	Rand	0,7302	0,6163	2190
11º	Rand	0,5354	0,8554	2367
12º	R-VNS	0,8463	0,9578	2692

Tabela 7. Resultados para o *ranking* de recomendação da instância ftv64.

Ranking	Método	α	γ	Distância
1º	B-VNS	0,7779	0,1115	2027
2º	R-VNS	0,5686	0,1580	2050
3º	B-VNS	0,7579	0,0715	2076
4º	Base	0,7879	0,0615	2103
5º	Base	0,7879	0,0615	2106
6º	B-VNS	0,7979	0,0515	2139
7º	Base	0,7879	0,0615	2158
8º	R-VNS	0,3536	0,3443	2220
9º	R-VNS	0,2828	0,4669	2412
10º	Rand	0,2931	0,4507	2534
11º	Rand	0,0446	0,6132	2849
12º	Rand	0,5610	0,9812	3728

parâmetros vindos da base de conhecimento (Base) já terem mostrado resultados positivos em Ottoni et al. (2018) e o método B-VNS utilizar desses parâmetros para realizar uma busca nas proximidades. Logo, esses dois métodos foram elaborados para terem α e γ próximos de ótimos locais.

No entanto, é possível notar que os métodos com soluções iniciais aleatórias (R-VNS e Rand) também produziram resultados relevantes nos *rankings*. Como por exemplo na Tabela, 2 o método Rand obteve a quarta posição. Além disso, na Tabela 6 o R-VNS alcançou a primeira colocação no *ranking* da instância ftv44. Logo, é possível concluir que mesmo com a eficiência dos métodos que utilizam a base de conhecimento, as técnicas que trabalham com a aleatoriedade também podem alcançar boas soluções.

Além disso, é possível notar que para as seis primeiras posições dos *rankings*, os valores de α e γ sofrem pouca variação para a maioria das instâncias. Por exemplo, na instância berlin52 (Tabela 2): o valor de α varia entre [0,7421 a 0,7821] e os valores de γ no intervalo [0,0369 a 0,0729].

4.2 Análise dos Métodos de Otimização

Na Tabela 8 são apresentadas as médias dos valores de distância por cada instância e método adotado no AutoRL-TSP.

Tabela 8. Média da distância mínima pelas três repetições dos métodos em cada instância.

Instância	Ótimo	Rand	Base	R-VNS	B-VNS
berlin52	7542	9012,35	8328,81	9042,00	**8002,00**
kroA100	21282	51854,58	24065,35	27805,56	**23588,32**
kroA200	29368	39738,61	35741,29	40831,55	**32391,69**
ftv33	1286	1512,00	1406,67	1438,33	**1381,00**
ftv44	1613	2226,33	1877,67	2106,67	**1790,33**
ftv64	1839	3037,00	2122,33	2227,33	**2080,67**

Pode-se verificar na Tabela 8 que para todas as instâncias o método B-VNS obteve menor média da distância mínima (valores em destaque). É possível justificar a melhor atuação do B-VNS devido as suas características. Inicializa-se os parâmetros (α, γ) obtidos pela base de conhecimento vindo de Ottoni et al. (2018) que já apresenta bons valores de distância quando comparado a outros trabalhos da literatura. Em seguida, através das estruturas de vizinhanças que realizam pequenas modificações nos parâmetros, é possível realizar uma busca local nas proximidades da solução inicial, encontrando assim valores para (α, γ) que resultam em distâncias menores na rota.

Na sequência, para verificar se existe diferença significativa entre as soluções (distâncias) dos métodos de otimização foi adotado o teste de Análise de Variância (ANOVA). A ANOVA foi aplicada para verificar a validade de aceitar a hipótese nula (H_0) de igualdade no desempenho dos quatro métodos de otimização do AutoRL-TSP por instância. Por outro lado, H_0 é rejeitada e a hipótese alternativa (H_1) é aceita se pelo menos um dos métodos apresenta diferença significativa em relação aos demais, considerando as médias de distâncias nas rotas (μ_i) (Montgomery, 2017):

$$\begin{cases} H_0: & \mu_1 = \mu_2 = \mu_3 = \mu_4 \\ H_1: & \mu_i \neq \mu_j \quad \text{para pelo menos um par } i, j \end{cases}$$

Adotando um critério de significância de 5%, os resultados da ANOVA indicaram que existe diferença entre os métodos de otimização (p-valor $< 0{,}05$) para as instâncias: berlin52, kroA100, kroA200, ftv33 e ftv64. Apenas para a instância ftv44, a ANOVA indicou que não existe diferença significativa entre os métodos (p-valor $> 0{,}05$). Vale ressaltar que, as premissas da normalidade dos resíduos, homoscedasticidade e independência foram observadas e respeitadas.

5. CONCLUSÃO

O objetivo deste trabalho foi propor um sistema de AutoML para ajuste de parâmetros do AR aplicado ao Problema do Caixeiro Viajante: AutoRL-TSP. Para isso, foi adotada uma base de conhecimento (Ottoni et al., 2018) e desenvolvidas estruturas de otimização e recomendação de α e γ.

Os resultados apresentam os *rankings* de recomendação de parâmetros de acordo com a instância (TSPLIB) e o método de otimização simulado. Vale ressaltar que, ao adotar a base de conhecimento e o algoritmo VNS, o sistema de AutoML alcançou os melhores resultados. Dessa forma, destaca-se a relevância da adoção de uma estrutura de otimização robusta, realizando uma busca local (VNS) em uma região de parâmetros já ajustados pela RSM.

Em trabalhos futuros, sugere-se a aplicação e análise do sistema proposto em outras instâncias e problemas de otimização combinatória. Além disso, serão incluidas outras funcionalidades na busca por deixar o AutoRL-TSP mais aprimorado na tentativa de se aproximar ainda mais dos valores ótimos dos problemas da TSPLIB. Nesse sentido, espera-se adicionar métodos para recomendação de algoritmos e funções de recompensa (Hutter et al., 2019; Ottoni et al., 2020), assim como, técnicas de transferência de aprendizado (Taylor and Stone, 2009; Da Silva and Reali Costa, 2019). Outra possibilidade é a análise de mais estruturas de vizinhança para o VNS (Mladenović and Hansen, 1997; Liu and Zhou, 2013).

AGRADECIMENTOS

Agradecemos à CAPES, CNPq/INERGE, FAPEMIG, FAMAM, UFRB e UFSJ (Edital nº 001/2019/Reitoria).

REFERÊNCIAS

Alipour, M.M., Razavi, S.N., Feizi Derakhshi, M.R., and Balafar, M.A. (2018). A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. *Neural Computing and Applications*, 30(9), 2935–2951.

Bianchi, R.A.C., Ribeiro, C.H.C., and Costa, A.H.R. (2009). On the relation between Ant Colony Optimization and Heuristically Accelerated Reinforcement Learning. *1st International Workshop on Hybrid Control of Autonomous System*, 49–55.

Bodin, L., Golden, B., Assad, A., and Ball, M. (1983). Routing and Scheduling of Vehicles and Crews – The State of the Art. *Computers and Operations Research*, 10(2), 63–211.

Brazdil, P., Carrier, C.G., Soares, C., and Vilalta, R. (2009). *Metalearning: Applications to data mining.* Springer Science & Business Media.

Cai, H., Lin, J., Lin, Y., Liu, Z., Wang, K., Wang, T., Zhu, L., and Han, S. (2020). Automl for architecting efficient and specialized neural networks. *IEEE Micro*, 40(1), 75–82. doi:10.1109/MM.2019.2953153.

Cardenoso Fernandez, F. and Caarls, W. (2018). Parameters Tuning and Optimization for Reinforcement Learning Algorithms Using Evolutionary Computing. In *2018 International Conference on Information Systems and Computer Science (INCISCOS)*, 301–305. IEEE.

Chiang, H.T.L., Faust, A., Fiser, M., and Francis, A. (2019). Learning navigation behaviors end-to-end with autorl. *IEEE Robotics and Automation Letters*, 4(2), 2007–2014.

Coelho, D. (2016). *Estruturas de Memória e Operadores Adaptativos em Meta-Heurísticas*. Ph.D. thesis, Tese de Doutorado. UFMG-PPGEE.

Cordeiro, L.T. and Batista, L.S. (2018). Análise multicritério do projeto de dimensionamento de tubulações de redes de gás natural sob condições de incerteza da evolução da carga. *XIII SIMMEC 2018 - Simpósio de Mecânica Computacional, 2018.*, 1–14.

Cunha, T., Soares, C., and de Carvalho, A.C. (2018). Metalearning and recommender systems: A literature review and empirical study on the algorithm selection problem for collaborative filtering. *Information Sciences*, 423, 128–144.

Da Silva, F. and Reali Costa, A. (2019). A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64, 645–703.

Even-Dar, E. and Mansour, Y. (2003). Learning Rates for Q-learning. *Journal of Machine Learning Research*, 5, 1–25.

Faust, A., Francis, A., and Mehta, D. (2019). Evolving rewards to automate reinforcement learning. *6th ICML Workshop on Automated Machine Learning (2019)*.

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., and Hutter, F. (2015). Efficient and robust automated machine learning. In C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, 2962–2970. Curran Associates, Inc.

Gambardella, L.M. and Dorigo, M. (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. *Proceedings of the 12th International Conference on Machine Learning*, 252–260.

Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3), 449–467.

Hutter, F., Hoos, H., and Leyton-Brown, K. (2014). An efficient approach for assessing hyperparameter importance. In *Proceedings of International Conference on Machine Learning 2014 (ICML 2014)*, 754–762.

Hutter, F., Kotthoff, L., and Vanschoren, J. (eds.) (2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer. In press, available at http://automl.org/book.

Liao, C.J. and Cheng, C.C. (2007). A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date. *Computers & Industrial Engineering*, 52(4), 404–413.

Lima Júnior, F.C., Neto, A.D.D., and Melo, J.D. (2010). *Traveling Salesman Problem, Theory and Applications*, chapter Hybrid Metaheuristics Using Reinforcement Learning Applied to Salesman Traveling Problem, 213–236. InTech.

Lins, R.A.S., Dória, A.D.N., and de Melo, J.D. (2019). Deep reinforcement learning applied to the k-server problem. *Expert Systems with Applications*, 135, 212–218.

Liu, L. and Zhou, H. (2013). Hybridization of harmony search with variable neighborhood search for restrictive single-machine earliness/tardiness problem. *Information Sciences*, 226, 68–92.

Mantovani, R.G., Rossi, A.L., Alcobaça, E., Vanschoren, J., and de Carvalho, A.C. (2019). A meta-learning recommender system for hyperparameter tuning: Predicting when tuning improves svm classifiers. *Information Sciences*, 501, 193–221.

Mantovani, R.G., Rossi, A.L., Vanschoren, J., Bischl, B., and Carvalho, A.C. (2015). To tune or not to tune: recommending when to adjust svm hyper-parameters via meta-learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.

McAuley, A., Sinkar, K., Kant, L., Graff, C., and Patel, M. (2012). Tuning of reinforcement learning parameters applied to OLSR using a cognitive network design tool. In *2012 IEEE Wireless Communications and Networking Conference (WCNC)*, 2786–2791. IEEE.

Mladenović, N. and Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11), 1097–1100.

Montgomery, D.C. (2017). *Design and analysis of experiments*. New York: John Wiley & Sons., 9th edition.

Ottoni, A.L.C., Nepomuceno, E.G., and de Oliveira, M.S. (2018). A response surface model approach to parameter estimation of reinforcement learning for the travelling salesman problem. *Journal of Control, Automation and Electrical Systems*, 29(3), 350–359.

Ottoni, A.L.C., Nepomuceno, E.G., de Oliveira, M.S., and de Oliveira, D.C.R. (2020). Tuning of reinforcement learning parameters applied to sop using the scott–knott method. *Soft Computing*, 24, 4441–4453.

Reinelt, G. (1991). TSPLIB - A Traveling Salesman Problem Library. *ORSA Journal on Computing*, 3(4), 376–384.

Russell, S.J. and Norving, P. (2013). *Artificial Intelligence*. Campus, 3st ed.

Schweighofer, N. and Doya, K. (2003). Meta-learning in reinforcement learning. *Neural Networks*, 16(1), 5–9.

Silva, I.N., Spatti, D.H., and Flauzino, R.A. (2016). *Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas: fundamentos teóricos e aspectos práticos*. ArtLiber.

Sutton, R. and Barto, A. (2018). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 2nd edition.

Taylor, M.E. and Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul), 1633–1685.

Tsiakmaki, M., Kostopoulos, G., Kotsiantis, S., and Ragos, O. (2019). Implementing automl in educational data mining for prediction tasks. *Applied Sciences*, 10(1).

Watkins, C.J. and Dayan, P. (1992). Technical note Q-learning. *Machine Learning*, 8(3), 279–292.