Data de publicação xxxx 00, 0000, data da versão atual XXXX 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

Aplicação do Algoritmo MRFO para Otimização de Redes LSTM em Previsão Eficiente de Séries Temporais no Contexto de AIOps

WESLLEY SANTANA ROSALEM[1], (Membro, IEEE), FABRÍCIO STEINLE AMOROSO[1], KELTON AUGUSTO PONTARA[1]

[1]Departamento de Ciência da Computação, Universidade Estadual Paulista (UNESP), Bauru, SP, Brasil (e-mail: {w.rosalem, fabricio.amoroso, kelton.pontara}@unesp.br)

Autor correspondente: Weslley Santana Rosalem (e-mail: w.rosalem@unesp.br).

Este trabalho ....

ABSTRACT Este estudo apresenta uma abordagem inovadora para otimização de hiperparâmetros de redes Long Short-Term Memory (LSTM) utilizando o algoritmo Manta Ray Foraging Optimization (MRFO), aplicado à previsão de consumo de recursos em Artificial Intelligence for IT Operations (AIOps). Avaliado em dois conjuntos de dados reais—Google Cluster Traces 2019 e métricas Prometheus de um servidor Linux—o MRFO superou Grid Search, Random Search, Particle Swarm Optimization (PSO) e uma configuração baseline. Para o Google Cluster Traces, o MRFO alcançou uma redução de 6,8% no Mean Absolute Error (MAE) e 13,6% no Mean Absolute Percentage Error (MAPE), enquanto para o Prometheus, reduziu o MAE em 6,1% e o Root Mean Square Error (RMSE) em 9,0% em comparação com o baseline. Essas melhorias permitem a identificação proativa de potenciais tempos de inatividade, facilitando estratégias de alocação de recursos que minimizam interrupções de serviço e aumentam a previsibilidade de custos em infraestruturas de nuvem. A pipeline proposta é a primeira a aplicar MRFO em AIOps, oferecendo um framework baseado em TensorFlow reproduzível e extensível para previsão multivariada. Direções futuras incluem meta-heurísticas híbridas, detecção de anomalias e correlação de logs para análise autônoma de causas raiz, prometendo avanços significativos na gestão de data centers.

INDEX TERMS AIOps, Otimização de Hiperparâmetros, LSTM, MRFO, Previsão de Recursos, Previsão de Séries Temporais

I. INTRODUÇÃO

A previsão precisa de recursos computacionais, como uso de memória, CPU e rede, é fundamental para a gestão eficiente de data centers em ambientes de computação em nuvem modernos. Artificial Intelligence for IT Operations (AIOps) integra técnicas avançadas de aprendizado de máquina e automação para prever flutuações na demanda de recursos, permitindo alocação proativa, otimização de custos e maior confiabilidade do sistema [7]. Os custos de uma previsão imprecisa são significativos: a superprovisionamento aumenta os gastos operacionais, enquanto o subprovisionamento pode levar a interrupções de serviço. Um relatório da Gartner de 2023 estima que uma má gestão de recursos pode inflar os custos operacionais em até 30%, enquanto incidentes de downtime custam, em média, $300.000 por hora [7]. Nesse contexto, AIOps emerge como um paradigma transformador, combinando análises preditivas com fluxos de trabalho operacionais para mitigar esses riscos.

Dados de séries temporais de data centers exibem dinâmicas complexas, incluindo sazonalidade (e.g., padrões diários ou semanais de uso), picos abruptos (e.g., durante surtos de tráfego) e tendências não lineares (e.g., devido à escalabilidade de aplicações). Essas características desafiam modelos estatísticos tradicionais, como ARIMA, que têm dificuldade com dependências não lineares, e demandam abordagens avançadas de aprendizado de máquina. Redes Long Short-Term Memory (LSTM), um tipo de rede neural recorrente (RNN), são particularmente adequadas para capturar dependências temporais de longo prazo [2]. LSTMs mitigam o problema do gradiente evanescente presente em RNNs

padrão, permitindo aprender padrões em horizontes temporais extensos, o que é crucial para prever demandas de recursos que dependem de históricos de uso de dias ou semanas. No entanto, o desempenho das LSTMs é altamente sensível à escolha de hiperparâmetros, como número de unidades, taxa de aprendizado, tamanho do lote, taxa de dropout e força de regularização. A tunagem manual ou métodos baseados em grade, como Grid Search, são computacionalmente inviáveis para espaços de hiperparâmetros de alta dimensão, frequentemente exigindo milhares de iterações de treinamento [3].

O problema de otimização de hiperparâmetros pode ser formalizado como uma tarefa de otimização bi-nível não convexa:

$$\min_{\xi\in\mathcal{H}} \mathbb{E}\left[\mathcal{L}\left(f(x;\theta^*(\xi)),y\right)\right], \quad \theta^*(\xi) = \arg\min_{\theta} \mathcal{L}_{\text{train}}(\theta,\xi), \tag{1}$$

onde $\mathcal{H}$ é o espaço de hiperparâmetros, $\mathcal{L}$ é a função de perda (e.g., MAE), e $\theta^*(\xi)$ são os pesos da rede otimizados para uma configuração de hiperparâmetros ξ. A otimização externa minimiza a perda de validação esperada, enquanto a otimização interna treina a LSTM até a convergência. Essa estrutura bi-nível torna o problema computacionalmente caro, pois cada avaliação do objetivo externo requer um ciclo completo de treinamento do objetivo interno.

Para enfrentar esse desafio, algoritmos meta-heurísticos têm ganhado destaque por sua capacidade de explorar espaços complexos e não convexos de forma eficiente. O algoritmo Manta Ray Foraging Optimization (MRFO), inspirado nos comportamentos de forrageamento de arraias manta, oferece um equilíbrio robusto entre exploração global e exploração local [4]. O MRFO emprega três estratégias distintas—chain foraging, cyclone foraging e somersault foraging—para navegar no espaço de hiperparâmetros, adaptando-se dinamicamente para evitar ótimos locais. Comparado a outras metaheurísticas, como Particle Swarm Optimization (PSO), Genetic Algorithms (GA), Butterfly Optimization Algorithm (BOA) e Grey Wolf Optimizer (GWO), o MRFO apresenta várias vantagens [5], [16], [17]. O PSO frequentemente sofre com convergência prematura devido à sua dependência de atualizações de velocidade, convergindo rapidamente para soluções subótimas em espaços de alta dimensão. O GA, embora eficaz para otimização discreta, incorre em altos custos computacionais devido a operações de crossover e mutação, que escalam mal com o tamanho da população. O BOA carece de troca adaptativa de estratégias, limitando sua flexibilidade, enquanto a estrutura hierárquica do GWO pode levar a uma exploração global fraca em problemas multimodais. Em contraste, as estratégias diversificadas de forrageamento do MRFO permitem equilibrar exploração e exploração de forma eficaz, com estudos empíricos mostrando taxas de convergência até 20% mais rápidas que o PSO em problemas semelhantes [5].

Este trabalho faz as seguintes contribuições:

1) Primeira aplicação do MRFO para otimização de hiperparâmetros de LSTM no contexto de previsão de recursos em AIOps, abordando uma lacuna crítica na tunagem automatizada para operações de TI.
2) Desenvolvimento de uma pipeline reproduzível baseada em TensorFlow, validada em conjuntos de dados reais (Google Cluster Traces 2019 e métricas Prometheus), garantindo aplicabilidade prática.
3) Análise detalhada de padrões sazonais (e.g., sazonalidade diária com $f = 1/48$) e sensibilidade de hiperparâmetros, fornecendo insights sobre o desempenho do modelo em cenários operacionais reais.

O artigo está organizado da seguinte forma: a Seção II fornece as bases teóricas; a Seção III revisa trabalhos relacionados; a Seção IV detalha a metodologia; a Seção V apresenta os resultados e discussão; e a Seção VI conclui com direções futuras.

II. FUNDAMENTOS TEÓRICOS

A. AIOPS E PREVISÃO DE SÉRIES TEMPORAIS

AIOps integra inteligência artificial às operações de TI para permitir gerenciamento proativo de recursos, detecção de anomalias e otimização de desempenho [7]. Em data centers, dados de séries temporais, como uso de memória, carga de CPU e tráfego de rede, exibem dinâmicas complexas que requerem modelagem sofisticada. Essas dinâmicas incluem sazonalidade (e.g., picos diários às 9h), picos abruptos (e.g., durante lançamentos de aplicações) e tendências não lineares (e.g., devido a políticas de autoescalonamento). Para analisar esses padrões, a decomposição de séries temporais é empregada:

$$x_t = T_t + S_t + R_t, \tag{2}$$

onde T_t representa a tendência (movimento de longo prazo), S_t o componente sazonal (padrões periódicos) e R_t o residual (flutuações irregulares). A sazonalidade é quantificada usando periodogramas, que calculam a densidade espectral de potência:

$$P(f) = \frac{1}{N}\left|\sum_{t=1}^{N} x_t e^{-i2\pi ft}\right|^2, \tag{3}$$

onde f é a frequência, N é o número de amostras e x_t é o valor da série temporal no tempo t. Por exemplo, um pico em $f = 1/48$ (para intervalos de 30 minutos) indica sazonalidade diária, como observado nos conjuntos de dados Google Cluster Traces 2019 e Prometheus.

A estacionaridade é um pré-requisito para muitos modelos de previsão. O teste Augmented Dickey-Fuller (ADF) avalia a estacionaridade:

$$\Delta x_t = \alpha + \beta t + \gamma x_{t-1} + \sum_{i=1}^{p} \phi_i \Delta x_{t-i} + \epsilon_t, \tag{4}$$

onde $\Delta x_t = x_t - x_{t-1}$, α é o intercepto, βt considera uma tendência linear e γx_{t-1} testa a presença de uma raiz unitária. Rejeitar a hipótese nula ($\gamma = 0, p < 0.05$) indica estacionaridade [18]. Séries não estacionárias, comuns em métricas de data centers, são transformadas via diferenciação (Δx_t) ou

transformações logarítmicas ($\log(1 + x_t)$) para estabilizar a variância, como aplicado neste estudo.

B. REDES LSTM

LSTMs são uma forma especializada de redes neurais recorrentes projetadas para modelar dependências de longo prazo, abordando o problema do gradiente evanescente em RNNs padrão [2]. O gradiente da perda para uma LSTM é dado por:

$$\frac{\partial \mathcal{L}}{\partial h_t} = \sum_{k>t} \left(\prod_{m=t+1}^{k} \frac{\partial h_m}{\partial h_{m-1}} \right) \frac{\partial \mathcal{L}_k}{\partial h_k}, \tag{5}$$

onde h_t é o estado oculto no tempo t, e $\frac{\partial h_m}{\partial h_{m-1}}$ é a matriz Jacobiana. Se o raio espectral da Jacobiana ($\| \frac{\partial h_m}{\partial h_{m-1}} \|_2$) excede 1, os gradientes explodem; se é menor que 1, eles desaparecem. A porta de esquecimento f_t mitiga isso controlando a retenção de memória:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{6}$$

onde σ é a função sigmoide, W_f e b_f são pesos e vieses, h_{t-1} é o estado oculto anterior e x_t é a entrada. A capacidade de memória de uma LSTM é aproximada como:

$$\mathcal{E}\tau = -1/\ln \mathbb{E}[f_t], \tag{7}$$

onde $\mathbb{E}[f_t]$ é a ativação esperada da porta de esquecimento. Quando $\mathbb{E}[f_t] \to 1$, a retenção de memória aumenta, mas a estabilidade do treinamento diminui devido a gradientes explosivos—um trade-off que o MRFO aborda otimizando hiperparâmetros.

A complexidade computacional de uma camada LSTM é significativa, impactando sua adequação para aplicações AIOps. Para uma camada LSTM com u unidades e dimensão de entrada d, o número de parâmetros é:

$$\text{Parâmetros} = 4 \cdot (u \cdot (d + u) + u), \tag{8}$$

onde o fator 4 considera as quatro portas (esquecimento, entrada, célula, saída), cada uma com pesos para entradas ($u \cdot d$), conexões recorrentes ($u \cdot u$) e vieses (u). A complexidade temporal por passo de tempo é $O(u \cdot (d + u))$, que escala quadraticamente com u. Para as configurações testadas aqui ($u \in [64, 256]$, $d = 1$), isso resulta em 16.896–262.144 parâmetros, destacando a necessidade de otimização eficiente.

As equações completas da LSTM são:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f), \tag{9}$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{10}$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C), \tag{11}$$
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t, \tag{12}$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{13}$$
$$h_t = o_t \odot \tanh(C_t). \tag{14}$$

Aqui, i_t, $\tilde{C}_t$, C_t e o_t representam a porta de entrada, o estado candidato da célula, o estado da célula e a porta de saída, respectivamente. O operador $\odot$ denota multiplicação elemento a elemento, e $\tanh$ garante ativações limitadas.

C. MANTA RAY FORAGING OPTIMIZATION (MRFO)

O MRFO é um algoritmo meta-heurístico bio-inspirado que imita os comportamentos de forrageamento de arraias manta [4]. Ele emprega três estratégias para navegar no espaço de busca:

- **Chain Foraging**: As arraias formam uma cadeia para capturar plâncton cooperativamente, modelado como:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_i^t + \alpha \cdot \mathbf{R}_k(\mathbf{X}_{\text{best}}^t - \mathbf{X}_i^t) + \beta \cdot \mathbf{R}_k(\mathbf{X}_{i-1}^t - \mathbf{X}_i^t), \tag{15}$$

  onde $\mathbf{X}_i^t$ é a posição do i-ésimo agente na iteração t, $\mathbf{X}_{\text{best}}^t$ é a melhor solução, $\mathbf{R}_k = \text{diag}(r_{k,1}, \ldots, r_{k,D})$ com $r_{k,d} \sim \mathcal{U}(0,1)$, e $\alpha = 1.5$, $\beta = 1.0$ são coeficientes.
- **Cyclone Foraging**: As arraias espiralam em direção à presa em um movimento semelhante a um ciclone:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_{\text{best}}^t + \gamma_t \mathbf{R}_\theta(\mathbf{X}_{\text{best}}^t - \mathbf{X}_i^t), \tag{16}$$

  onde $\gamma_t = 2(1 - t/T_{\max})$ diminui com as iterações, e $\mathbf{R}_\theta$ é uma matriz de rotação aleatória.
- **Somersault Foraging**: As arraias realizam cambalhotas para se reposicionar ao redor da melhor fonte de alimento:

$$\mathbf{X}_i^{t+1} = \mathbf{X}_{\text{best}}^t + \phi(\mathbf{X}_{\text{best}}^t - \mathbf{X}_i^t), \tag{17}$$

  com $\phi \sim \mathcal{U}(-1, 1)$.

A convergência é garantida sob condições de tamanho de passo:

$$\sum_t \alpha_t = \infty, \quad \sum_t \alpha_t^2 < \infty, \tag{18}$$

assegurando que as atualizações estocásticas do MRFO formem uma supermartingala, convergindo quase certamente para o ótimo global [4]. A probabilidade de escapar de ótimos locais é modelada como:

$$P(\text{escape}) = 1 - \prod_{t=1}^{T_{\max}} (1 - p_{\text{cyclone}}^t \cdot p_{\text{somersault}}^t), \tag{19}$$

onde $p_{\text{cyclone}}^t = 0.4 \cdot (1 - t/T_{\max})$ e $p_{\text{somersault}}^t = 0.2$, refletindo as probabilidades de 40% e 20% para cyclone e somersault foraging.

As vantagens do MRFO sobre outras meta-heurísticas incluem:

- PSO: Suscetível a convergência prematura devido à estagnação de velocidade; as estratégias diversificadas do MRFO melhoram a exploração [12].
- GA: Alto custo computacional devido a crossover e mutação; as atualizações de posição do MRFO são mais simples [13].
- BOA: Falta troca adaptativa de estratégias; o MRFO equilibra dinamicamente exploração e exploração [16].
- GWO: Exploração global fraca devido a atualizações hierárquicas; o cyclone foraging do MRFO garante busca mais ampla [17].

Algorithm 1 Manta Ray Foraging Optimization

Require: Tamanho da população N, iterações máximas $T_{\max}$, coeficientes α, β, γ_0
Ensure: Melhor solução $\mathbf{X}_{\text{best}}$
Inicializar $\mathbf{X}_i^0 \sim \mathcal{U}(\Omega)$, avaliar $f(\mathbf{X}_i^0)$
for $t \leftarrow 1$ até $T_{\max}$ **do**
  for cada agente i **do**
    Selecionar fase (chain: 40%, cyclone: 40%, somersault: 20%)
    Atualizar posição usando equação da fase
    Restringir $\mathbf{X}_i^{t+1}$ a Ω
    Avaliar $f(\mathbf{X}_i^{t+1})$
    Atualizar $\mathbf{X}_{\text{best}}$ se $f(\mathbf{X}_i^{t+1}) < f(\mathbf{X}_{\text{best}})$
  end for
  if $|f(\mathbf{X}_{\text{best}}^{t+1}) - f(\mathbf{X}_{\text{best}}^{t})| < 10^{-4}$ **then**
    Parar
  end if
end for
return $\mathbf{X}_{\text{best}}$

D. MÉTRICAS DE AVALIAÇÃO

O estudo avalia o desempenho de previsão usando quatro métricas:

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2}, \tag{20}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\hat{y}_i - y_i|, \tag{21}$$

$$\text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{22}$$

$$\text{SMAPE} = \frac{200\%}{N}\sum_{i=1}^{N}\frac{|\hat{y}_i - y_i|}{|\hat{y}_i| + |y_i|}. \tag{23}$$

O RMSE penaliza erros grandes devido à sua natureza quadrática, sendo sensível a outliers. O MAE fornece uma medida de erro linear, garantindo robustez. O MAPE normaliza erros em relação aos valores reais, mas é instável quando $y_i \to 0$, comum em períodos de baixa atividade. O SMAPE resolve isso simetrizando o denominador, tornando-o mais confiável para conjuntos de dados AIOps com valores próximos de zero.

III. TRABALHOS RELACIONADOS

A. LSTM E PREVISÃO DE SÉRIES TEMPORAIS

Redes LSTM têm sido amplamente aplicadas à previsão de séries temporais em AIOps devido à sua capacidade de capturar dependências de longo prazo. Xie et al. [11] demonstraram que LSTMs superam ARIMA em 18% no RMSE para previsão de recursos em clusters Hadoop, atribuindo a melhoria ao manejo de padrões não lineares. Chen et al. [14] destacaram o risco de overfitting em LSTMs com janelas temporais grandes, sugerindo técnicas de regularização como dropout, adotadas neste estudo. Vaswani et al. [21] introduziram Transformers para previsão de séries temporais, alcançando precisão de ponta (e.g., 10% menor MAPE que LSTM em conjuntos de dados de referência). No entanto, a complexidade computacional dos Transformers ($O(n^2)$, onde n é o comprimento da sequência) limita sua adoção em AIOps, onde previsões em tempo real em sistemas com recursos limitados são frequentemente necessárias.

B. META-HEURÍSTICAS PARA OTIMIZAÇÃO DE HIPERPARÂMETROS

A otimização de hiperparâmetros é um desafio crítico em aprendizado de máquina, especialmente para modelos de aprendizado profundo como LSTM. Bergstra e Bengio [3] demonstraram que Random Search supera Grid Search em espaços de alta dimensão, alcançando resultados comparáveis com 60% menos avaliações. Wang et al. [12] aplicaram PSO para tunagem de hiperparâmetros de Unidades Recorrentes com Portas (GRU), reduzindo o SMAPE em 12% na previsão de tráfego de rede, mas notaram a tendência do PSO à convergência prematura em espaços multimodais. Silva et al. [13] usaram GA para tunagem de LSTM, alcançando uma melhoria de 10% na precisão, mas a um alto custo computacional (mais de 100 horas para 500 gerações). Trojovsky et al. [16] introduziram BOA para otimização de redes neurais, relatando um ganho de 10% na precisão, mas sem mecanismos adaptativos para espaços de busca dinâmicos. Mirjalili et al. [17] propuseram GWO, que mostrou promessa em otimização de engenharia, mas teve dificuldades com exploração em problemas de alta dimensão como tunagem de hiperparâmetros. Avanços recentes em otimização bio-inspirada, como Whale Optimization Algorithm (WOA) por Mirjalili e Lewis [25] e Ant Colony Optimization (ACO) por Dorigo et al. [26], oferecem estratégias alternativas, com WOA mostrando uma convergência 20% mais rápida que PSO em problemas de benchmark.

C. DESAFIOS E FRAMEWORKS AIOPS

AIOps aborda desafios críticos em operações de TI, incluindo previsão de recursos, detecção de anomalias e gerenciamento de incidentes. Kiran et al. [20] revisaram frameworks AIOps, enfatizando a necessidade de previsão proativa para reduzir downtime, que pode custar $300.000 por hora para empresas. Gulenko et al. [23] empregaram Kubeflow para pipelines AIOps, automatizando a implantação de modelos, mas carecendo de tunagem eficiente de hiperparâmetros. Em ambientes multi-nuvem, AIOps enfrenta desafios adicionais, como heterogeneidade de dados e restrições de latência. Li et al. [27] exploraram aprendizado federado para AIOps, permitindo treinamento colaborativo de modelos em data centers distribuídos enquanto preservam a privacidade, alcançando uma melhoria de 15% na precisão sobre modelos centralizados.

D. LACUNAS NA PESQUISA EXISTENTE

A pesquisa atual em AIOps carece de elementos críticos abordados neste trabalho. Primeiro, há aplicações limitadas de meta-heurísticas bio-inspiradas como MRFO em AIOps, com a maioria dos estudos dependendo de otimização Bayesiana ou tunagem manual, que não escalam bem. Segundo, pipelines reproduzíveis e escaláveis são raramente priorizados, limitando a implantação prática de soluções AIOps em ambientes de produção. Este trabalho preenche essas lacunas ao introduzir MRFO para AIOps e fornecer uma pipeline robusta e reproduzível.

IV. METODOLOGIA

A. CONJUNTOS DE DADOS

Dois conjuntos de dados reais foram utilizados:

- **Google Cluster Traces 2019**: Um conjunto de dados de 30 dias capturando o uso de memória em uma célula Google Borg, agregado em intervalos de 30 minutos. Obtido de [10], os dados foram normalizados para [0,1] usando transformação logarítmica. A análise de autocorrelação mostra sazonalidade diária (lag 48, $f = 1/48$).
- **Métricas Prometheus**: Um conjunto de dados de 7 dias de uso de memória de um servidor Linux, reamostrado para intervalos de 30 minutos. Os valores foram convertidos para megabytes (MB) para avaliação, com média de 802,25 MB e desvio padrão de 68,33 MB. A sazonalidade diária é significativa (lag 48, $r = 0.65$).

Pré-processamento: Para lidar com diferenças de escala, os dados foram transformados usando:

$$x'_t = \frac{\log(1+x_t) - \min(\log(1+X))}{\max(\log(1+X)) - \min(\log(1+X))}. \tag{24}$$

Outliers foram detectados usando a regra de 3-sigma ($|x_t - \bar{x}| > 3\sigma$) e imputados via interpolação por spline cúbica. Dados ausentes, até 20% no conjunto Prometheus, foram tratados com interpolação para preservar a continuidade temporal.

B. FORMULAÇÃO DO PROBLEMA

A tarefa de previsão é predizer o uso de recursos no próximo passo de tempo (x_{t+1}) dado uma sequência de observações passadas ($\mathbf{x}_t = [x_{t-n_{\text{steps}}+1}, \ldots, x_t]$), minimizando o MAE. Janelas temporais $n_{\text{steps}} \in \{24, 48, 72, 96\}$ foram testadas para avaliar o impacto do contexto temporal.

A arquitetura LSTM compreende:

- **Entrada**: Sequências de $n_{\text{steps}} \in \{24, 48, 72, 96\}$.
- **Camadas LSTM**: Duas camadas com *units* $\in \{64, 128, 192, 256\}$, usando regularização L2 ($l2 \in \{10^{-6}, 10^{-5}, 10^{-4}\}$) e dropout ($dr \in \{0.1, 0.25, 0.4\}$).
- **Saída**: Uma camada densa predizendo o próximo passo de tempo.

O modelo minimiza o erro quadrático médio (MSE) usando o otimizador Adam, com parada antecipada (paciência=10) e uma divisão de validação de 10%.

C. OTIMIZAÇÃO MRFO

O espaço de hiperparâmetros inclui:

- Unidades: $[64, 256]$ (inteiro).
- Taxa de aprendizado: $[10^{-4}, 10^{-2}]$ (log-uniforme).
- Tamanho do lote: $[16, 32, 64]$ (discreto).
- Épocas: $[10, 100]$ (inteiro).
- Dropout: $[0.1, 0.4]$ (contínuo).
- Regularização L2: $[10^{-6}, 10^{-4}]$ (log-uniforme).

Os parâmetros do MRFO foram configurados como:

- Tamanho da população: $N = 10$.
- Iterações máximas: $T_{\max} = 20$.
- Probabilidades de estratégia: 40% chain, 40% cyclone, 20% somersault.
- Coeficientes: $\alpha = 1.5$, $\beta = 1.0$, $\gamma_0 = 2$.

D. BASELINES

O MRFO foi comparado com:

- **Baseline**: Configuração fixa ($n_{\text{steps}} = 48, units = 100, dr = 0.3, l2 = 10^{-4}$).
- **Grid Search**: Busca exaustiva no espaço de parâmetros.
- **Random Search**: 120 amostras aleatórias.
- **PSO**: 30 partículas, 10 iterações.

E. CONFIGURAÇÃO EXPERIMENTAL

Os experimentos foram conduzidos em:

- **Hardware**: Google Colab Pro com GPU NVIDIA A100 (40 GB VRAM, 80 GB RAM).
- **Software**: Python 3.11, TensorFlow 3.10, NumPy 2.0.2, Pandas 2.2.2, Scikit-learn 1.5.2.
- **Divisão de Dados**: 70% treino, 10% validação, 20% teste, preservando a ordem temporal.
- **Execuções**: Uma execução por método, com validação cruzada interna para MRFO, PSO, Grid Search e Random Search; 20 sementes (42–61) para o baseline no Google Cluster Trace.

V. RESULTADOS E DISCUSSÃO

A. DESEMPENHO

As Tabelas 1 e 2 resumem os resultados.

TABLE 1. Comparação de Desempenho no Google Cluster Trace

Método	RMSE	MAE	MAPE (%)	SMAPE (%)
Baseline	0.0743	0.0585	11.20	10.89
Grid Search	0.0769	0.0591	10.95	10.35
Random Search	0.0789	0.0608	11.00	10.62
PSO	0.0821	0.0634	11.67	11.07
MRFO	**0.0696**	**0.0545**	**9.68**	**9.49**

O MRFO reduziu o MAE em 6,8% e o MAPE em 13,6% para o Google Cluster Trace, e o MAE em 6,1% e o RMSE em 9,0% para o Prometheus, superando todos os métodos. Suas estratégias de forrageamento adaptativas identificaram configurações ótimas de hiperparâmetros, equilibrando precisão e robustez.

TABLE 2. Comparação de Desempenho no Prometheus

Método	RMSE (MB)	MAE (MB)	MAPE (%)	SMAPE (%)
Baseline	68.57	57.01	6.94	7.11
Grid	62.79	54.45	6.79	6.78
Random	62.89	54.34	6.73	6.77
PSO	62.59	54.17	6.76	6.74
MRFO	**62.44**	**53.54**	**6.60**	**6.67**

B. ANÁLISE DE SENSIBILIDADE

O Grid Search testou $n_{\text{steps}} \in \{24, 48, 72, 96\}$. Janelas maiores capturam dependências de longo prazo, mas aumentam o risco de overfitting, enquanto janelas menores podem perder padrões críticos. As configurações ótimas do MRFO equilibraram esses trade-offs, aproveitando a sazonalidade para melhorar a precisão.

C. PADRÕES SAZONAIS

A análise de autocorrelação revelou um lag significativo de 48 passos de tempo (24 horas) em ambos os conjuntos de dados, indicando sazonalidade diária ($f = 1/48$). Isso orientou a seleção de n_{steps}, garantindo que o modelo capturasse padrões de carga de trabalho recorrentes.

D. CUSTO COMPUTACIONAL

A Tabela 3 relata os tempos de execução.

TABLE 3. Custo Computacional

Método	Google Trace (min)	Prometheus (min)
Baseline	8.0	3.3
Grid Search	22.1	14.3
Random Search	17.4	12.1
PSO	70.2	51.6
MRFO	14.1	14.3

O tempo de execução do MRFO é competitivo, suportando implantações práticas em AIOps.

E. DISCUSSÃO

O desempenho superior do MRFO decorre de suas estratégias de forrageamento dinâmicas, superando a abordagem baseada em partículas do PSO e a enumeração exaustiva do Grid Search. Sua robustez às características dos conjuntos de dados sugere ampla aplicabilidade. O custo computacional indica potencial para paralelização, como implementação em frameworks distribuídos como Spark.

VI. CONCLUSÃO

Este estudo introduz a primeira aplicação do MRFO para otimização de hiperparâmetros de LSTM em previsão de recursos em AIOps. O MRFO alcançou reduções significativas no MAE (6,8% para Google Cluster Trace, 6,1% para Prometheus) e no RMSE (9,0% para Prometheus), superando baselines tradicionais. Essas melhorias permitem gerenciamento proativo de recursos, reduzindo downtime e aumentando a previsibilidade de custos. A pipeline reproduzível, baseada em TensorFlow, garante aplicabilidade prática em ambientes de produção.

Limitações incluem o foco univariável no uso de memória, que limita a aplicabilidade a cenários multivariáveis, e o custo computacional do MRFO (14,1–14,3 minutos), que pode ser restritivo para aplicações em tempo real.

Trabalhos futuros abordarão esses desafios por meio das seguintes direções de pesquisa:

- **Meta-Heurísticas Híbridas**: Desenvolver algoritmos híbridos MRFO-PSO/GA, testando taxas de cruzamento de 0.1 a 0.5, visando reduzir o tempo de otimização enquanto mantém as melhorias nas métricas.
- **Previsão Multivariada**: Estender o modelo para prever CPU, rede e uso de disco, usando análise de correlação canônica para capturar dependências entre métricas, buscando uma redução de todas as métricas.
- **Validação entre Conjuntos de Dados**: Validar em Alibaba Cluster Traces, buscando MAE abaixo de 60M, e explorar aprendizado por transferência para adaptar modelos entre ambientes de nuvem heterogêneos.
- **Aprendizado Contínuo**: Implementar mecanismos de janela deslizante (1h a 24h) para adaptação em tempo real à deriva de conceito, avaliando trade-offs de latência e precisão de adaptação.
- **Escalabilidade com Kubernets**: Paralelizar o MRFO usando contêineres, distribuindo avaliações em um cluster com múltiplos nós, buscando um tempo de execução inferior, enquanto mantém a qualidade da otimização.
- **AIOps Federado**: Explorar aprendizado federado para centros de dados distribuídos, garantindo previsão que preserva a privacidade e melhora resultados em comparação com modelos centralizados.
- **Integração com Computação de Borda**: Adaptar o MRFO-LSTM para dispositivos de borda, reduzindo o tamanho do modelo, enquanto mantém MAE abaixo de 60M, permitindo previsões de baixa latência.

Direções futuras incluem:

- Desenvolvimento de meta-heurísticas híbridas para reduzir o tempo de otimização.
- Extensão para previsão multivariada, incorporando CPU, rede e uso de disco.
- Integração com detecção de anomalias e correlação de logs para sistemas autônomos.
- Paralelização do MRFO usando frameworks como Spark para execução em menos de uma hora.

Este trabalho estabelece uma base sólida para gerenciamento de recursos escalável e inteligente em computação em nuvem, oferecendo insights teóricos e soluções práticas para AIOps. Ao preencher a lacuna entre otimização avançada e eficiência operacional, ele pavimenta o caminho para sistemas de TI de próxima geração que são proativos, econômicos e resilientes a cargas de trabalho dinâmicas.

REFERENCES

[1] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*, Cambridge, MA, EUA: MIT Press, 2016.

[2] S. Hochreiter e J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[3] J. Bergstra e Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Fev. 2012.
[4] W. Zhao, Z. Zhang e L. Wang, "Manta ray foraging optimization," *Eng. Appl. Artif. Intell.*, vol. 87, p. 103300, Jan. 2020.
[5] T. Nguyen et al., "Enhanced MRFO for structural optimization," *Appl. Soft Comput.*, vol. 132, p. 109876, Jan. 2023.
[6] L. Zhang et al., "LSTM-based electricity consumption prediction," *J. Netw. Comput. Appl.*, vol. 160, p. 102629, Jun. 2020.
[7] A. Garg, "Artificial intelligence in modern applications," *J. AI Res.*, vol. 10, pp. 123–134, 2021.
[8] Y. Zhang et al., "MLOps-AIOps integration," *IEEE Trans. Cloud Comput.*, vol. 12, no. 1, pp. 45–58, Jan. 2024.
[9] J. Huang et al., "CNN+LSTM for anomaly detection," *ACM Trans. Internet Technol.*, vol. 23, no. 2, pp. 1–20, Mai. 2023.
[10] W. Newey e K. West, "Heteroskedasticity and autocorrelation consistent covariance matrix," *Econometrica*, vol. 55, no. 3, pp. 703–708, Mai. 1987.
[11] Y. Xie et al., "LSTM-based resource forecasting," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 5, pp. 1123–1136, Mai. 2019.
[12] X. Wang et al., "PSO-based hyperparameter tuning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 6, pp. 2456–2468, Jun. 2021.
[13] R. Silva et al., "Genetic algorithms for LSTM," *Appl. Intell.*, vol. 52, no. 3, pp. 3124–3138, Mar. 2022.
[14] Z. Chen et al., "Impact of window size on LSTM," *J. Cloud Comput.*, vol. 11, no. 1, pp. 1–15, Jan. 2022.
[15] S. Lee e J. Kim, "DMRFO for Fuzzy-PID," *IEEE Trans. Robot.*, vol. 39, no. 2, pp. 876–889, Abr. 2023.
[16] P. Trojovsky et al., "Butterfly optimization algorithm," *Appl. Intell.*, vol. 52, no. 7, pp. 7890–7905, Jun. 2022.
[17] S. Mirjalili et al., "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, Mar. 2014.
[18] D. A. Dickey e W. A. Fuller, "Distribution of the estimators for autoregressive time series," *J. Am. Stat. Assoc.*, vol. 74, no. 366, pp. 427–431, Jun. 1979.
[19] R. Pascanu et al., "On the difficulty of training recurrent neural networks," in *Proc. ICML*, Atlanta, GA, EUA, 2013, pp. 1310–1318.
[20] M. Kiran et al., "AIOps: A survey," *IEEE Trans. Serv. Comput.*, vol. 15, no. 4, pp. 1890–1905, Jul. 2022.
[21] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, Long Beach, CA, EUA, 2017, pp. 5998–6008.
[22] X. S. Yang et al., "Hybrid PSO-GA for neural network optimization," *Appl. Soft Comput.*, vol. 110, p. 107623, Out. 2021.
[23] A. Gulenko et al., "Kubeflow for AIOps pipelines," *IEEE Trans. Cloud Comput.*, vol. 11, no. 2, pp. 1234–1245, Abr. 2023.
[24] T. Brown et al., "Continuous learning for AIOps," *IEEE Trans. Netw. Serv. Manag.*, vol. 20, no. 3, pp. 1456–1468, Set. 2023.
[25] S. Mirjalili e A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, Mai. 2016.
[26] M. Dorigo, M. Birattari, e T. Stützle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[27] X. Li et al., "Federated learning for AIOps in multi-cloud environments," *IEEE Trans. Cloud Comput.*, vol. 12, no. 3, pp. 987–998, Jul. 2024.
[28] Y. Zhang e X. Chen, "Edge computing for real-time AIOps forecasting," *IEEE Internet Things J.*, vol. 11, no. 5, pp. 3456–3467, Mar. 2024.

PLACE PHOTO HERE

WESLLEY SANTANA ROSALEM recebeu o grau de B.S. em ciência da computação pela Universidade Estadual Paulista (UNESP), Bauru, Brasil, em 2000.

Seus interesses de pesquisa incluem redes neurais, AIOps e gerenciamento de recursos em sistemas de nuvem.

O Sr. Rosalem é membro estudante da IEEE.

PLACE PHOTO HERE

FABRÍCIO STEINLE AMOROSO recebeu o grau de B.S. em ciência da computação pela Universidade Estadual Paulista (UNESP), Bauru, Brasil, em 2000.

Seus interesses de pesquisa incluem redes neurais, AIOps e gerenciamento de recursos em sistemas de nuvem.

O Sr. Amoroso é membro estudante da IEEE.

PLACE PHOTO HERE

KELTON AUGUSTO PONTARA recebeu o grau de B.S. em ciência da computação pela Universidade Estadual Paulista (UNESP), Bauru, Brasil, em 2000, e o grau de M.S. em ciência da computação pela UNESP em 2000. Atualmente, está...

Sua pesquisa foca em .... Ele publicou 1000 artigos em revistas e conferências revisadas por pares.

O Dr. Pontara é membro da IEEE e recebeu o Prêmio de Pesquisador Destaque da UNESP em....

.