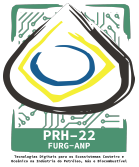


UNIVERSIDADE FEDERAL DO RIO GRANDE - FURG

CENTRO DE CIÊNCIAS COMPUTACIONAIS

PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

CURSO DE MESTRADO EM ENGENHARIA DE COMPUTAÇÃO

Dissertação de Mestrado

Aprendizado profundo como suporte para algoritmo de distribuição de tarefa

Cleverton Bueno dos Santos Júnior

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal do Rio Grande - FURG, como requisito parcial para a obtenção do grau de Mestre em Engenharia de Computação

Orientador: Prof. Dr. Paulo Lilles Jorge Drews Junior

Rio Grande, 2025

S237a Santos Júnior, Cleverton Bueno dos
Aprendizado profundo como suporte para algoritmo de distribuição de tarefa / Cleverton Bueno dos Santos Júnior. – 2025.
88 f.

Dissertação (Mestrado) – Universidade Federal do Rio Grande – Programa de Pós-Graduação em Computação, 2025.

Orientador: Dr. Paulo Lilles Jorge Drews Junior.

1. Computação. 2. Veículos guiados automaticamente. 3. Aprendizado de máquina. 4. Aprendizagem profunda. 5. Q-learning. I. Drews Junior, Paulo Lilles Jorge. II. Título.

CDU 004

Catalogação na fonte: Bibliotecário Clériston Ribeiro Ramos CRB10/1889

Universidade Federal do Rio Grande
Centro de Ciências Computacionais
Programa de Pós-Graduação em Computação
Curso de Mestrado em Engenharia de Computação

DISSERTAÇÃO DE MESTRADO

Aprendizado profundo como suporte para algoritmo de distribuição de tarefa

Cleverton Bueno dos Santos Júnior

Banca examinadora:

Documento assinado digitalmente
gov.br
EDER MATEUS NUNES GONCALVES
Data: 16/05/2025 17:05:20-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. Eder Mateus Nunes Gonçalve

Documento assinado digitalmente
gov.br
FELIPE MARTINS MULLER
Data: 14/06/2025 10:52:25-0300
Verifique em https://validar.iti.gov.br

_____

Prof. Dr. Felipe Martins Muller

Assinado de forma digital por Paulo Lilles Jorge Drews Junior:00124140009
Dados: 2025.05.16 16:47:07 -03'00'

_____

Prof. Dr. Paulo Lilles Jorge Drews Júnior
Orientador

Dedico. . . À minha família, principalmente meus pais.
Aos meus amigos.

AGRADECIMENTOS

O autor agradece o apoio financeiro do Programa de Recursos Humanos da Agência Nacional do Petróleo, Gás Natural e Biocombustíveis (PRH/ANP – PRH22.1/FURG), suportado com recursos provenientes do investimento de empresas petrolíferas qualificadas na Cláusula de PDI da Resolução ANP nº 50/2015. O presente trabalho foi realizado com apoio da Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP), Brasil. Processo nº 2024/10523-5..

O autor agradece também grupo de pesquisa LOGNAV 4.0, sob o qual este projeto foi desenvolvido, pelo apoio dos colegas e professores que foram fundamentais durante o percurso do desenvolvimento deste trabalho.

We live in a society exquisitely dependent on science and technology,
in which hardly anyone knows anything about science and technology.
— CARL SAGAN

RESUMO

JÚNIOR, Cleverton Bueno dos Santos. **Aprendizado profundo como suporte para algoritmo de distribuição de tarefa**. 2025. 88 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

De tempos em tempos, surgem inovações tecnológicas tão impactantes que revolucionam certos aspectos da indústria. Nos últimos anos, cada vez mais pesquisas evidenciam os sistemas multiagentes e a robótica como uma dessas inovações na indústria de manejo de materiais. Um dos problemas mais importantes deste cenário, se encontra na distribuição de tarefas, sendo uma área amplamente explorada. O problema de coleta e entrega é um dos problemas oriundos do Problema do Caixeiro Viajante, que juntamente com a otimização de rotas é um dos principais problemas logísticos de sistemas autônomos dos últimos tempos. Atualmente, existem diversos algoritmos que lidam com este problema. Entretanto, dada a natureza do problema, conforme a complexidade do sistema aumenta, é necessária a utilização de técnicas cada vez mais complexas, computacionalmente custosas e com uma intricada implementação. Redes Neurais, por serem capazes de aprendizado e extrapolação de padrões, surgem como uma grande ferramenta para uma possível solução desta necessidade. Neste trabalho, é apresentado o uso de uma dupla de redes neurais como ferramentas de suporte para algoritmos de distribuição de tarefas e também verifica a efetividade desta técnica. Esta proposta visa alavancar a eficácia de um algoritmo com uma implementação simples, e com isso, diminuir as perdas de desempenho causadas por implementações mais complexas. Esta dissertação traz uma breve fundamentação teórica para um melhor entendimento dos assuntos abordados, bem como uma análise do cenário atual de pesquisa e, então, apresenta os resultados obtidos, que demonstram uma possibilidade positiva para do proposto neste estudo quando aplicada ao problema do caixeiro viajante, mas em uma versão mais complexa, os resultados demonstram incapacidade em encontrar soluções viáveis.

Palavras-chave: Aprendizado de Máquina, Aprendizado Profundo, Aprendizado-Q, Distribuição de Tarefas.

ABSTRACT

JÚNIOR, Cleverton Bueno dos Santos. **Deep Learning as a support tool for a task distribution algorithm**. 2025. 88 f. Dissertação (Mestrado) – Programa de Pós-Graduação em Computação. Universidade Federal do Rio Grande - FURG, Rio Grande.

From time to time, technological innovations emerge that are so impactful that they revolutionize certain aspects of the industry. In recent years, growing research evidence has highlighted multi-agent systems and robotics as one such innovation in the materials handling industry. One of the most significant problems in this industry lies in the pickup and delivery problem, so much so, that it became a widely explored area. The pickup and delivery problem is one of the main problems stemming from the Traveling Salesman Problem and, along with route optimization, is one of the primary logistical problems in the autonomous systems of recent times. Currently, there are various algorithms for solving this problem. Unfortunately, due to the nature of the problem, as the complexity of the system rises, it's necessary the use of more complex techniques, more computational costs and with difficult implementation. Neural Networks, due to their learning and pattern extrapolation capabilities, are powerful tools for a potential solution to these problems of optimization loss. This work brings forth a pair of neural networks that act as a support tool for task distribution algorithms, and also verifies the effectiveness of this approach. This proposal aims to raise the efficiency of a task distribution algorithm with a simple implementation, and with that, curb the losses coming from the implementation and design of more complex algorithms. This dissertation provides a brief theoretical foundation for a better understanding of the topics addressed, as well as an analysis of the current research landscape, and it presents the achieved results, that show a positive outlook when the neural networks were used to solve the traveling salesman problem, but show a very negative outlook when apllied to a more complex version of the problem.

Keywords: Machine Learning, Deep Learning, Q-Learning, Task Distribution.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

DL	Deep Learning
DNN	Deep Neural Network
DQL	Deep Q-Learning
IA	Inteligência Artificial
ML	Machine Learning
QL	Q-Learning
RL	Reinforcement Learning
RNA	Rede Neural Artificial

SUMÁRIO

1 INTRODUÇÃO

Com o passar do tempo, vão surgindo novas tecnologias, algumas tão impactantes que revolucionam completamente alguns aspectos da indústria. Ultimamente, percebe-se um aumento em pesquisas relacionadas a sistemas multiagentes e robótica no âmbito do manuseio de materiais. Apesar de os veículos autoguiados (AGVs) serem utilizados para realizar estas tarefas desde meados da década de 50 (Wurman, D'Andrea e Mountz [73]), sua aplicação tinha como foco o transporte de cargas volumosas e/ou pesadas, como blocos de motores e rolos de papel não cortados. Entretanto, isso mudou com a chegada das tecnologias acessíveis que proporcionam comunicação sem fio, um poder computacional mais robusto e componentes robóticos mais sofisticados [73].

Wurman, D'Andrea e Mountz [73] dizem que estas inovações vêm contribuindo para uma certa "democratização"dos veículos autônomos, isto é, eles têm se tornado mais baratos, compactos e com uma maior capacidade de realização de tarefas, podendo realizar tarefas muito mais complexas. Isto tem mudado as possibilidades e a eficiência da gestão logística, permitindo que tarefas anteriormente desafiadoras sejam facilmente automatizadas, o que aumenta a flexibilidade das operações. Conforme a tecnologia evolui, espera-se que a integração de sistemas multiagentes e robótica proporcione não apenas uma revolução na eficiência, mas também abra portas para novos paradigmas de operações industriais e logísticas.

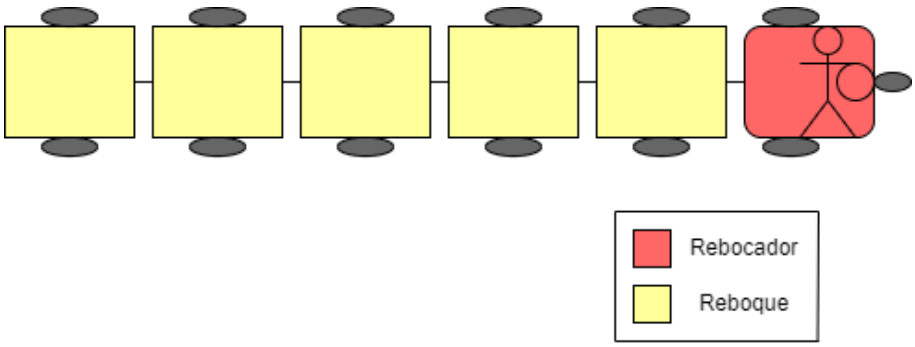
Neste estudo é abordado o problema de alocação e ordenação de tarefas em um armazém híbrido, isto é, um armazém onde unidades autônomas agem em conjunto com funcionários humanos, onde os veículos autônomos são responsáveis pelo transporte de materiais e ferramentas entre os seus devidos depósitos e as estações de trabalho dos funcionários, sendo aplicado, especificamente, no cenário de uma empresa de refino petrolífero. Inicialmente, foi utilizada uma versão simplificada do cenário, onde havia apenas uma unidade autônoma para realizar todas as tarefas, capaz de realizar apenas uma tarefa por vez, ou seja, a unidade possui capacidade para transportar apenas um tipo de objeto e não havia restrições em sua movimentação, o que torna o cenário um problema de roteamento de veículos.

Este problema inicial trata-se de uma das diversas formas em que um problema lo-

gístico famoso pode ser representado, conhecido problema do caixeiro viajante (PCV). Originalmente apresentado por Gavish e Graves [25], no PCV clássico, um caixeiro viajante necessita visitar um conjunto de cidades uma única vez, com o intuito de entregar objetos ou realizar diferentes tarefas e, ao final, retornar à cidade de origem, realizando esse trajeto com a menor distância possível.

Após os testes iniciais deste estudo, o problema toma sua forma real, com as unidades assumindo a forma a qual existem no sistema estudado, um rebocador que puxa diversos reboques, um pequeno comboio, permitindo que cada unidade tenha a capacidade de realizar mais de uma tarefa de forma conjunta. Além disso, os rebocadores podem ser pilotados por um colaborador ou agir de forma autônoma. A Figura 1 traz um conceito simples dos agentes, nesta fase, o problema torna-se um sistema multiagente, com várias unidades agindo ao mesmo tempo. Nesta nova forma do problema, devido às características intrínsecas dos veículos puxarem vários reboques, surge uma restrição física de extrema relevância, em que as unidades perdem a capacidade de se movimentarem em marcha à ré. Diferente de trens que se movimentam em trilhos fixos e conseguem emular uma marcha à ré com locomotivas em ambas as extremidades do comboio, a unidade retratada é capaz de movimentação livre no chão de fábrica e, com isso, no momento em que o rebocador tentar retornar pelo caminho percorrido sem realizar uma manobra para manter-se puxando sua carga, o conjunto do veículo torna-se um sistema caótico, com cada um dos reboques comportando-se de maneira distinta, acarretando no travamento total da unidade, na perda das cargas ou até mesmo em algum dano físico à unidade.

Figura 1: Imagem conceitual dos agentes utilizados



Fonte: Autor (2025)

Com estas características novas, o problema inicial, representado como um PCV simples, passa a ter uma complexidade exponencialmente maior, tornando-se um problema de coleta e entrega (PCE), que também é um problema NP-Difícil, citefurtado$_{o2}$017.$NoPCE, oslocaiscomdemandasaseremcoletadaseentreguessotipicamenterepresent$ $seumaquantidade$q$_i$ que deve ser coletada. Essa mesma quantidade deve ser entregue no nó $n+i$, que possui demanda $q_{n+i} = -q_i$, representando a respectiva entrega. Com isso,

o número de coletas deve ser igual ao número de entregas. Toda coleta deve ser realizada antes de sua respectiva entrega (restrição de precedência) e ambas devem estar na rota de um mesmo veículo (restrição de pareamento).

Além das restrições específicas dos AGVs, o problema possui limitações intrínsecas temporais, dado que as tarefas do sistema possuem janelas de tempo onde elas podem ser realizadas, sendo que estas tarefas podem ser subdivididas em carga e descarga, cada uma destas subtarefas com seus limites de início e término próprios, e como cada tipo de material ou ferramenta necessita de maneiras e cuidados diferentes durante seu carregamento e descarregamento, cada subtarefa também possui sua própria duração, com estas novas características, o problema torna-se uma variação do PCE, chamado de problema de coleta e entrega com janelas temporais (PCEJT).

Este problema pode, então, ser formalizado como um problema de Programação Linear Inteira Mista (MILP, do inglês *Mixed-Integer Linear Programming*). Nele, procura-se minimizar ou maximizar uma função objetivo linear, que está sujeita a um conjunto de restrições, também lineares, com parte das variáveis de decisão restringidas a valores inteiros. Essa característica torna a MILP particularmente adequada para representar decisões discretas, como alocação de tarefas, roteamento de veículos e sequenciamento de operações [14, 43].

Conforme discutido previamente, o problema discutido visa minimizar o custo total associado à execução das tarefas, considerando tanto o tempo de deslocamento entre os pontos quanto penalidades aplicadas a tarefas não concluídas e infrações das restrições de movimento. A seguir, o problema é expresso matematicamente através de sua função objetivo e formalização das restrições.

Seja $\mathcal{A}$ o conjunto de agentes, $\mathcal{T}$ o conjunto de tarefas e $\mathcal{N}$ o conjunto de nós que compõem o grafo do ambiente físico. Cada tarefa $j \in \mathcal{T}$ é composta por duas subtarefas (coleta e entrega), cada uma associada a nós distintos o_j (origem) e d_j (destino), com janelas temporais definidas por $[a_j^o, f_j^o]$ e $[a_j^d, f_j^d]$, respectivamente. Define-se c_{ij}^{uv} como o custo temporal para o agente i deslocar-se do nó u ao nó v para executar a tarefa j. A fim de representar as decisões, define-se a variável $x_{ij}^{uv} \in \{0, 1\}$, que assume valor 1 se o agente i realiza a tarefa j com deslocamento entre os nós u e v, e 0 caso contrário.

A função objetivo é expressa por:

$$\min \sum_{i \in \mathcal{A}} \sum_{j \in \mathcal{T}} \sum_{(u,v) \in \mathcal{N}^2} c_{ij}^{uv} \cdot x_{ij}^{uv}$$

sujeita às seguintes restrições:

- **Precedência entre subtarefas:**

$$t_j^d \geq t_j^o + \tau_j, \quad \forall j \in \mathcal{T}$$

- **Janelas temporais:**

$$a_j^o \leq t_j^o \leq f_j^o, \quad a_j^d \leq t_j^d \leq f_j^d, \quad \forall j \in \mathcal{T}$$

- **Definição da falha:**
  - se o AGV chegar após encerramento das janelas temporais;
  - se o AGV não a carga referente à tarefa atual;
  - se o AGV não possuí espaço para realizar a carga;
  - se o AGV infringir a restrição de movimento.

Inúmeras pesquisas nesta área focam em explorar aprimoramentos nos sistemas de estoque, como os artigos por Ashayeri e Gelders [3], Yang, Wu e Ma [74], Dotoli et al. [21] e Pazour e Carlo [54], através de diversas estratégias, como modificar o leiaute utilizado, dessa forma, reduzindo o custo para a locomoção entre os pontos das tarefas ou a utilização de algoritmos "estado-da-arte", que buscam organizar o itinerário de forma ótima, o que reduz o custo total do itinerário.

A partir das informações apresentadas, pode-se observar que o problema abordado nesta dissertação possui altíssima complexidade. Assim, com o intuito de facilitar o estudo, a formulação da solução proposta e o cumprimento das restrições temporais para a realização do trabalho, o autor adotou as restrições listadas a seguir como verdades durante o desenvolvimento deste estudo:

- Os tempos característicos de cada tarefa são imutáveis;
- Não ocorrem acidentes;
- Não existem obstruções;
- As unidades possuem bateria com duração máxima maior que o período de trabalho contínuo;
- As baterias dos AGVs são recarregadas por completo durante o período em que a empresa não trabalha;
- Diversos AGVs podem realizar diferentes instâncias a mesma tarefa ao mesmo tempo;
- Os tempos de movimentação entre cada nodo são os menores possíveis;
- Todos AGVs iniciam o dia de trabalho no mesmo nodo neutro;
- Todos nodos, exceto o neutro, estão conectados;

- O nodo neutro está conectado apenas aos armazéns.

Algoritmos de alocação de tarefas, como heurísticas e meta-heurísticas, apesar de possuírem implementação mais fácil, apresentam restrições quanto aos sistemas que conseguem atuar de forma efetiva. Cada um possuindo um conjunto específico de condições sob as quais sua performance é ótima, como número baixo de agentes, ausência de restrições complexas ou ambientes de baixa variabilidade. Entretanto, quando aplicados a sistemas logísticos mais exigentes como ambientes com múltiplos AGVs, janelas temporais, regras de precedência e restrições de movimentação, é comum que enfrentem limitações relacionadas à coordenação entre agentes, adaptação a eventos dinâmicos e escalabilidade.

A construção de um algoritmo robusto o suficiente para lidar com todas essas variáveis simultaneamente é bastante desafiador, exigindo uma modelagem detalhada, previsão das possíveis exceções e um custo computacional aceitável, o que pode vir a tornar a solução inflexível e difícil de ser generalizada para outros contextos.

A partir disso, propõe-se utilizar técnicas de aprendizado profundo como suporte aos algoritmos existentes, visando tirar proveito que uma Rede Neural Artificial (RNA), ao ser treinada no sistema a ser estudado, possui capacidade de aprender uma política de otimização passível de auxiliar na tomada de decisão de outros algoritmos. Essa política teria o potencial de ampliar as possibilidades de aplicações dos métodos tradicionais, permitindo que eles operem em cenários que extrapolam suas capacidades originais. Além disso, essa abordagem híbrida pode oferecer vantagens relevantes, como a redução da necessidade de ajustes manuais, um aumento na capacidade de adaptação a mudanças bruscas no sistema e também uma melhora na qualidade das soluções em contextos dinâmicos.

Dentre as diversas técnicas existem no âmbito do aprendizado profundo, o autor escolheu o aprendizado-Q [71] para a ser aplicado neste estudo, pois esta técnica permite ao agente aprender de maneira análoga à aprendizagem humana, isto é, de uma forma exploratória, recebendo recompensas quando executa uma tarefa corretamente e sendo punido em caso contrário. A introdução dessa técnica em conjunto com uma rede neural de aprendizado profundo representa um salto qualitativo no desenvolvimento do sistema, resultando em uma possível solução robusta capaz de realizar tarefas de maneiras complexas que superariam as capacidades criativas de um programador.

A sinergia entre o aprendizado-Q e uma RNA proporciona ao sistema a capacidade de aprender padrões complexos e adaptar-se dinamicamente às diversas situações que são passíveis de ocorrer [31]. Ao contrário das abordagens onde os algoritmos são rigidamente definidos, com regras estruturadas e imutáveis, o aprendizado de máquina permite que o sistema evolua e se aprimore com base em experiências passadas, criando comportamentos e estratégias que podem ser extraordinariamente complexos e adaptáveis. Assim, ao incorporar técnicas de aprendizado profundo, este sistema possui a capacidade de não apenas superar as limitações dos métodos convencionais, mas também de abrir

novos horizontes para a autonomia e a inteligência adaptativa em ambientes complexos e dinâmicos.

1.1 Objetivos Gerais e Específicos

Neste estudo, é apresentada a aplicação de uma RNA baseada em aprendizado por reforço, utilizando técnicas de Aprendizado-Q (QL), com o objetivo de proporcionar suporte às soluções existentes, simplificando a complexidade dos estudos necessários para a eficaz implementação de soluções e algoritmos em ambiente mais complexos que aqueles para os quais foram originalmente desenvolvidos. Este trabalho busca, além de apresentar a proposta, avaliar e analisar os resultados obtidos durante as fases experimentais, conforme será definido na seção 3. Ademais, como objetivos gerais, o trabalho visa propor aprofundamentos e delineamentos para futuras continuações dos estudos abordados.

No âmbito dos objetivos gerais do estudo proposto, a intenção é estabelecer um caminho para a expansão do conhecimento, identificando áreas que possam se beneficiar de futuras pesquisas e propor possíveis aprofundamentos. Esses objetivos gerais visam contribuir para a evolução contínua do entendimento sobre a aplicação de redes neurais em problemas de otimização, indo além do escopo imediato do PCEJT. Ao atingir esses objetivos, este trabalho aspira não apenas fornecer *insights* valiosos para a atual compreensão do PCEJT, mas também estabelecer bases sólidas para investigações subsequentes que possam expandir e enriquecer o campo de pesquisa em otimização de rotas por meio do emprego de técnicas avançadas, como as redes neurais.

1.2 Sumário do Capítulo

Neste capítulo é apresentado o tema estudado nesta dissertação, o cenário envolvido nos estudos, bem como aquilo que o autor buscou alcançar no decorrer destes e os motivos que levaram à escolha do assunto abordado. O capítulo seguinte aprofunda a fundamentação teórica do estudo, apresentando as principais metodologias e conceitos relacionados ao tema abordado neste trabalho. Após a apresentação teórica, o capítulo seguinte traz uma revisão bibliográfica de artigos que o autor julgou relevantes para o tema abordado, este é então seguido pelo capítulo apresentando a metodologia utilizada pelo autor no desenvolvimento desta dissertação, bem como os resultados obtidos durante os experimentos desenvolvidos, após o qual o autor apresenta suas conclusões no capítulo final deste estudo.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo, serão apresentados os tópicos fundamentais para a compreensão deste trabalho. Os assuntos abordados estão agrupados nas duas grandes áreas abordadas: Distribuição de Tarefas e Aprendizado de Máquina. Cada uma dessas áreas é subdividida em várias seções. A primeira seção de cada tema proporciona uma definição abrangente, apresentando o tema de maneira geral. As seções subsequentes aprofundam os conceitos essenciais para uma compreensão sólida do trabalho.

Assim, a próxima seção enfoca o tema da Distribuição de Tarefas, proporcionando uma visão geral abrangente, seguida por seções que detalham os conceitos cruciais para a compreensão profunda do trabalho.

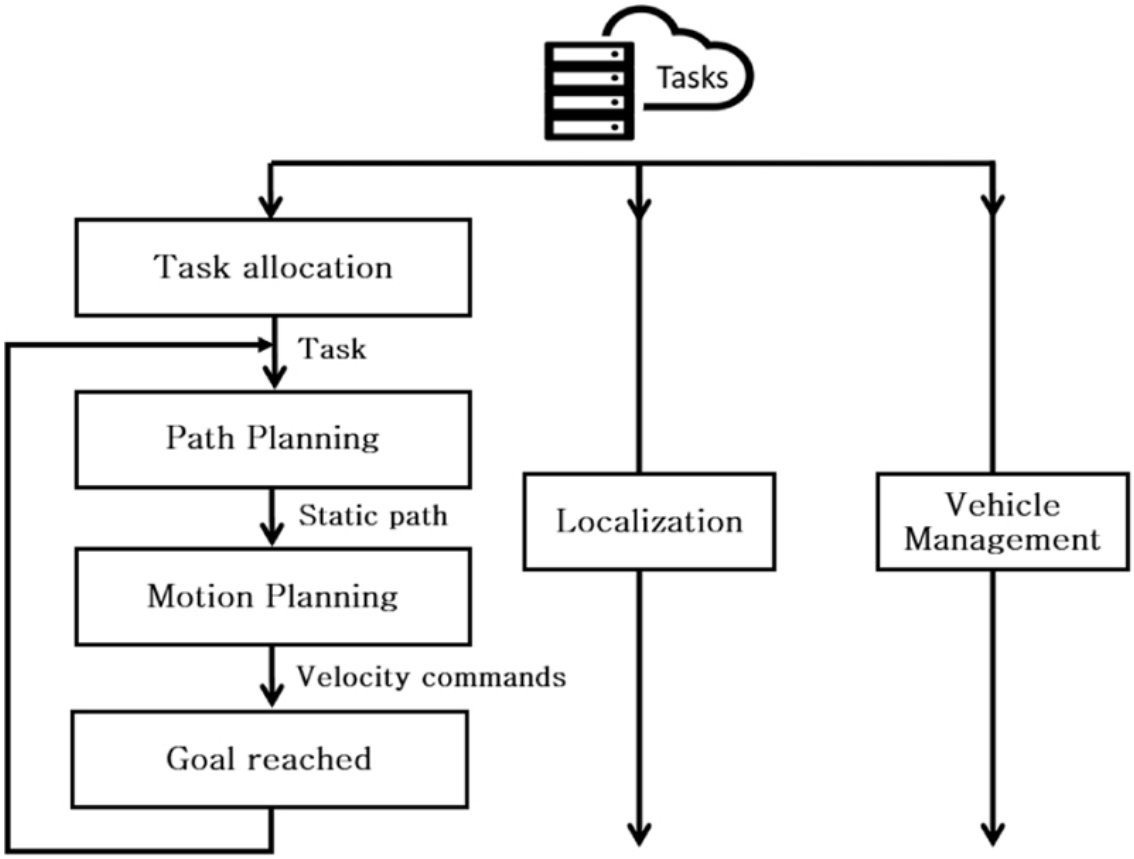
2.1 Distribuição de tarefas

Nesta seção, é apresentada uma breve introdução à distribuição de tarefas, seguida por definições cruciais deste domínio. Após são delineados dois tipos de abordagens para solucionar o desafio proposto e são apresentadas algumas das soluções amplamente empregadas na indústria atual.

A distribuição de tarefas representa um dos maiores desafios para sistemas de veículos autoguiados (AGVs). Este desafio consiste em alocar um conjunto de tarefas a um grupo de AGVs. Classificado como um problema NP-difícil de otimização restrita [30], a distribuição de tarefas busca minimizar o custo total das tarefas atribuídas. Por tratar de um problema NP-difícil, quando se lida com um grande número de tarefas e robôs, o espaço de soluções potenciais se torna vasto. Portanto, não existe um algoritmo eficiente capaz de proporcionar uma solução exata para o problema em um tempo finito [17]. A Figura 2 demonstra as principais áreas da distribuição de tarefas.

Para enfrentar os desafios inerentes aos problemas NP-difíceis, são utilizadas técnicas de otimização aproximada, como heurísticas [36] e meta-heurísticas [50]. Na distribuição de tarefas, uma tarefa pode se manifestar como um objeto específico que precisa ser recolhido em um determinado local e entregue em outro dentro de um armazém, ou também pode envolver uma tarefa de vigilância, onde um grupo de AGVs deve realizar uma

Figura 2: Distribuição de tarefas e suas principais áreas



Fonte: De Ryck, Versteyhe e Debrouwere [17]

varredura eficiente em uma determinada área [17].

2.1.1 Propriedades e Restrições

Nesta seção, são delineadas as principais características e significativas restrições associadas à distribuição de tarefas. Na alocação de tarefas, conforme destacado por Mosteo e Montano [47], a distribuição de tarefas possui duas propriedades de extrema relevância, que são a divisibilidade, que diz respeito à eficiente divisão da carga total de trabalho, visando maximizar a utilização de todos os recursos disponíveis, isto é, busca evitar a ociosidade de alguns AGVs enquanto outros estão ocupados. A segunda propriedade vital é a tolerância a falhas, que visa garantir que a alocação de tarefas funcione de maneira adequada, independente das falhas que possam ocorrer no sistema.

De Ryck, Versteyhe e Debrouwere [17] destacam outras propriedades que, embora menos proeminentes do que as anteriores, possuem um impacto significativo no desempenho de um sistema de distribuição de tarefas, elas são escalabilidade, flexibilidade e responsividade.

- Escalabilidade é a forma como o sistema pode ser aumentado sem problemas. Os algoritmos de distribuição devem continuar funcionando com uma frota de AGVs maiores sem atingir limites computacionais e de memórias.

- Flexibilidade é a necessidade do algoritmo deve, caso necessário, continuar operando se adaptando continuamente as mudanças no sistema.

- A distribuição das tarefas deve ser responsiva, o que significa que deve ter também uma alta performance em ambientes dinâmicos.

Algumas aplicações empregam tarefas independentes que são atribuídas a um AGV no momento em que se tornam disponíveis, como exemplificado por Wurman, D'Andrea e Mountz [73] no sistema Kiva, atual Amazon Robotics, onde as tarefas consistem na coleta e entrega de caixas, chamadas de *pods* no sistema, de armazenamento, onde os produtos estão guardados. Nestas instâncias, as informações de maior relevância são os locais de coleta e de entrega. Após esta alocação, o robô executa a tarefa, ganhando conhecimento sobre ela. Embora este seja um exemplo simplificado, aplicações no mundo real enfrentam diversas restrições: [17]

- Temporais: Tarefas podem possuir uma janela de tempo, onde existe uma duração para a conclusão desta tarefas, isto é, há um tempo de início e um período máximo para o término relacionados a cada tarefa individual.

- Precedência: Existem também restrições que fazem com que as tarefas sejam interdependentes. As tarefas podem ser parcialmente ordenadas, o que significa que algumas tarefas devem ser concluídas primeiro ou depois de outra tarefa. Tarefas podem ser casadas, onde duas ou mais tarefas devem ser executadas simultaneamente. Existem também incompatibilidades, onde tarefas tornam outras tarefas obsoletas.

- Interferência na mobilidade: Uma tarefa pode requerer que o AGV passe por uma rota com impedimentos como um corredor estreito onde apenas um agente pode atravessar por vez, ou como existir um outro agente parado na rota que deve ser percorrida para a realização da tarefa.

- Recursal: Esta pode prevenir que uma tarefa seja executada quando recursos ficam vazios. Por exemplo, a bateria de um AGV, que necessita de recarga para que o AGV possa dar continuidade a mais tarefas, ou então as peças necessárias para a realização da tarefa alocada tenha se esgotado.

Para demonstrar efetivamente estas restrições, várias representações são propostas por diversos autores, como as Redes Temporais Simples de Vidal e Bidot [69] e as Redes Hierárquicas de Tarefas de Minglei, Hongwei e Chao [44]. Essas estruturas oferecem

uma maneira robusta de modelar e compreender a complexidade da alocação de tarefas em ambientes dinâmicos. Ao alocar agentes a um conjunto de tarefas, sempre existe um objetivo a ser otimizado, como discutido por Mosteo e Montano [47], diversos objetivos de otimização podem ser empregados para orientar esse processo. A Tabela 1 apresenta alguns elementos que podem ser otimizados e também alguns dos objetivos de otimização. É importante notar que a escolha do objetivo de otimização varia conforme o contexto e as metas do sistema em que serão empregados.

Elemento otimizável	Definição
Custo	Custo para um robô realizar uma tarefa, podendo ser temporal, espacial, consumo de combustível, entre outros.
Fitness	Quão bem um robô consegue realizar uma tarefa.
Prioridade	Urgência para completar uma tarefa.
Recompensa	Ganho por completar uma tarefa.
Utilidade	A substração do custo da recompensa ou fitness.
Objetivo de otimização	
Custo Total	Busca minimizar o custo do pior robô.
Equalitário	Maximiza a utilidade do pior robô.
Maximizar o somatório das utilidades individuais.	
Maximizar a taxa de transferência.	
Minimizar o custo médio por tarefa.	
MinMax	Busca minimizar o custo do pior robô.

Fonte: Autor

Tabela 1: Elementos otimizáveis e Objetivos de otimização

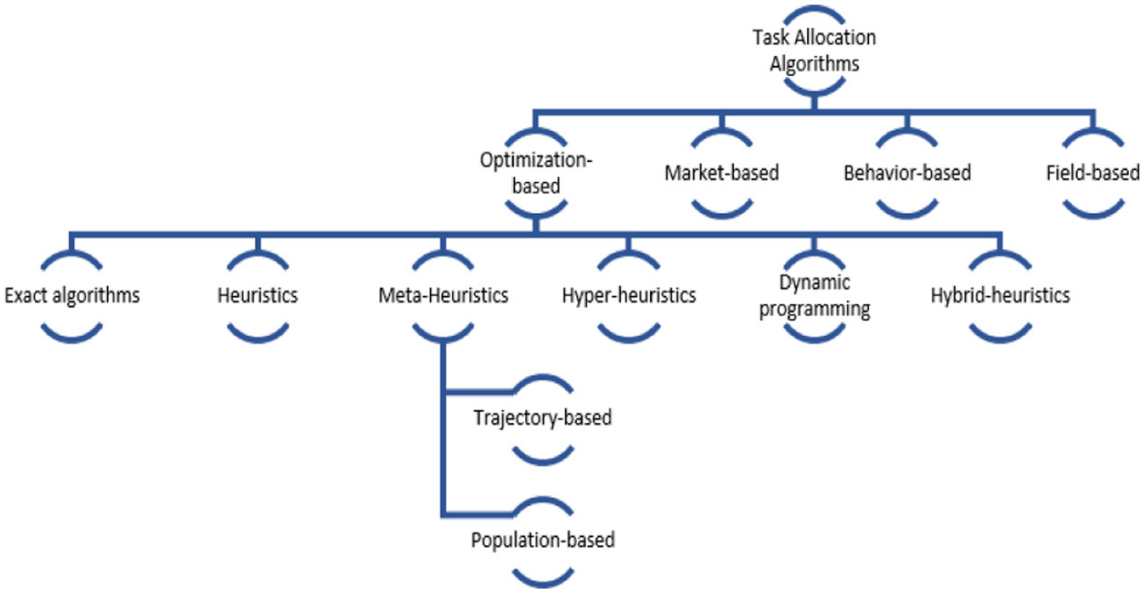
Considerando as propriedades desejadas no sistema e as restrições associadas às tarefas, a Distribuição de Tarefas em sistemas multiagentes atinge um nível de extrema complexidade e, a fim de lidar com essa complexidade, algoritmos específicos estão constantemente sendo desenvolvidos. Estes algoritmos buscam diferentes objetivos de otimização, cada um procurando se aproximar do ótimo global de maneiras diferentes. Na próxima seção, estas soluções são apresentadas e discutidas de forma concisa, junto com alguns de seus algoritmos.

2.1.2 Modelos de Soluções

Na última década, uma quantidade significativa de pesquisas foi dedicada à resolução do problema da distribuição de tarefas, gerando diversas soluções documentadas na literatura. Este trabalho adota a classificação proposta por De Ryck, Versteyhe e Debrouwere [17], que distingue os algoritmos de solução em duas grandes categorias: soluções baseadas em otimização e soluções baseadas no mercado, estas abordagens são discutidas em duas seções, onde são destacadas algumas das principais soluções de cada uma de-

las. Vale mencionar que métodos como os baseados em comportamento e baseados em campo, devido às suas aplicações limitadas, não serão discutidos. A Figura 3 ilustra essa distinção.

Figura 3: Classificação dos diferentes tipos de soluções



Fonte: De Ryck, Versteyhe e Debrouwere [17]

2.1.2.1 Métodos baseados em otimização

Nos métodos baseados em otimização, conforme destacado por Khamis, Hussein e Elmogy [32], um algoritmo busca a solução ótima em um espaço de soluções que visa maximizar algum lucro ou minimizar algum custo, utilizando as informações globais e levando em consideração todas as restrições. Enquanto o espaço de soluções for relativamente pequeno, será possível encontrar soluções exatas, por exemplo, em cenários com um número pequeno de tarefas e com poucos AGVs, é possível gerar uma matriz que representa os custos de locomoção de cada robô para cada tarefa, incorporando as restrições, é viável empregar um algoritmo exato capaz de encontrar a solução ótima para a distribuição das tarefas em um tempo finito. Entretanto, conforme o número de agentes e tarefas aumentam, o espaço de soluções cresce exponencialmente, tornando impossível utilizar métodos de solução exata. Esses problemas são mais conhecidos como problemas NP-difíceis, onde é muito pouco provável que existam algoritmos exatos que possam encontrar a solução ótima em um tempo finito, nesses casos, é imperativo recorrer a algoritmos de busca aproximada, tais como heurísticas, meta-heurísticas ou hiper-heurísticas. Nas próximas seções, alguns desses métodos serão descritos brevemente, apresentando alguns dos algoritmos mais utilizados.

2.1.2.1.1 Algoritmos Exatos

Alguns dos primeiros algoritmos de busca exata são os *Branch & Bound* e *Branch & Cut* para problemas de programação linear de inteiros mistos (MILP) [4], onde os objetivos e restrições são traduzidos como equações contendo variáveis inteiras e lineares, que são, então, resolvidas por algum algoritmo específico. Outro algoritmo exato é a Busca por força bruta, também conhecido por Busca Exaustiva [46], que avalia cada uma das soluções possíveis, e escolhe a melhor. Conforme dito anteriormente, estes algoritmos podem ser utilizados, apenas em espaços de soluções pequenos, significando um poucos robôs e tarefas, quando existem n robôs e p tarefas, então existem $n * p$ possíveis alocações, este número cresce rapidamente, tornando algoritmos exatos obsoletos para sistemas médios e grandes, também é o caso quando existem restrições muito complexas. [17]

2.1.2.1.2 Heurísticas

Um algoritmo heurístico é caracterizado pelo uso de uma função heurística. Conforme explicado por Pearl [55], a função heurística, mais conhecida apenas como heurística, é uma função que classifica alternativas diferentes em algoritmos de busca ou pesquisa em cada estágio das possíveis ramificações, tomando como base as informações disponíveis para escolher uma entre elas. Exemplos de heurísticas incluem a busca aleatória [35] e a subida de montanha [10], que geram uma solução a cada passo temporal durante um número fixo de passos, comparando essas soluções com a melhor já encontrada.

2.1.2.1.3 Meta Heurísticas

Conforme observado por De Ryck, Versteyhe e Debrouwere [17], heurísticas buscam melhorar soluções por meio de comparações, o que pode resultar em ficarem presas em ótimos locais. Esta é uma das causas que deu origem às meta-heurísticas, que, temporariamente, permitem soluções piores como estratégia viável, a fim de escapar dos mínimos locais. Essas meta-heurísticas são, na realidade, estratégias para orientar o processo de busca, e podem ser divididas em duas categorias principais: baseadas em trajetória e baseadas em população:

- Métodos baseados na trajetória são métodos onde um espaço de soluções é percorrido e a probabilidade de se escolher uma solução melhor ao invés de uma pior depende do momento da trajetória no intervalo de tempo do algoritmo. Exemplos são Busca local iterativa [66], busca em vizinhança variável [37] e Busca Tabu [35].
- Métodos baseados em população, são métodos onde populações são utilizadas para percorrer o espaço de soluções. Exemplos são Algoritmos Genéticos [33], Enxame de Partículas [34], Algoritmos Meméticos [41] e Otimização por Colônia de Formigas [38].

2.1.2.1.4 Hiper Heurísticas

Conforme destacado por Burke et al. [11], Hiper Heurísticas (HHs) são empregadas na automação do processo de seleção, combinação, adaptação ou geração de diversas heurísticas, com o objetivo de resolver problemas de busca. Dada a grande diversidade de heurísticas disponíveis para a solução de um problema, cada uma com suas vantagens e desvantagens, as HHs buscam escolher automaticamente as mais adequadas dentro de um conjunto de heurísticas a qualquer momento durante o processo de solução, dependendo do estado do problema. Essas heurísticas, em conjunto com as meta-heurísticas, são capazes de lidar com espaços de soluções muito maiores do que os algoritmos exatos, porém, à medida que o espaço de soluções cresce, a probabilidade de se encontrar o ótimo global diminui, portanto quanto maior o espaço de soluções e as restrições associadas, maior é a dificuldade em alcançar esse ótimo global [17].

2.1.2.1.5 Programação Dinâmica

A programação dinâmica (PD) é outro método que, geralmente, não se enquadra no que foi mencionado anteriormente, com o problema sendo dividido em vários subproblemas menores e mais simples, onde, nesses subproblemas, soluções podem ser encontradas de maneiras mais simples e rápida do que no problema completo. Isso acontece quando os subproblemas podem ser aninhados recursivamente no problema global, permitindo que o problema inicial seja resolvido através das soluções dos subproblemas mais simples, as soluções desses subproblemas são armazenadas em uma tabela e, ao decorrer da resolução do problema global, essas soluções podem ser utilizadas conforme necessário [62].

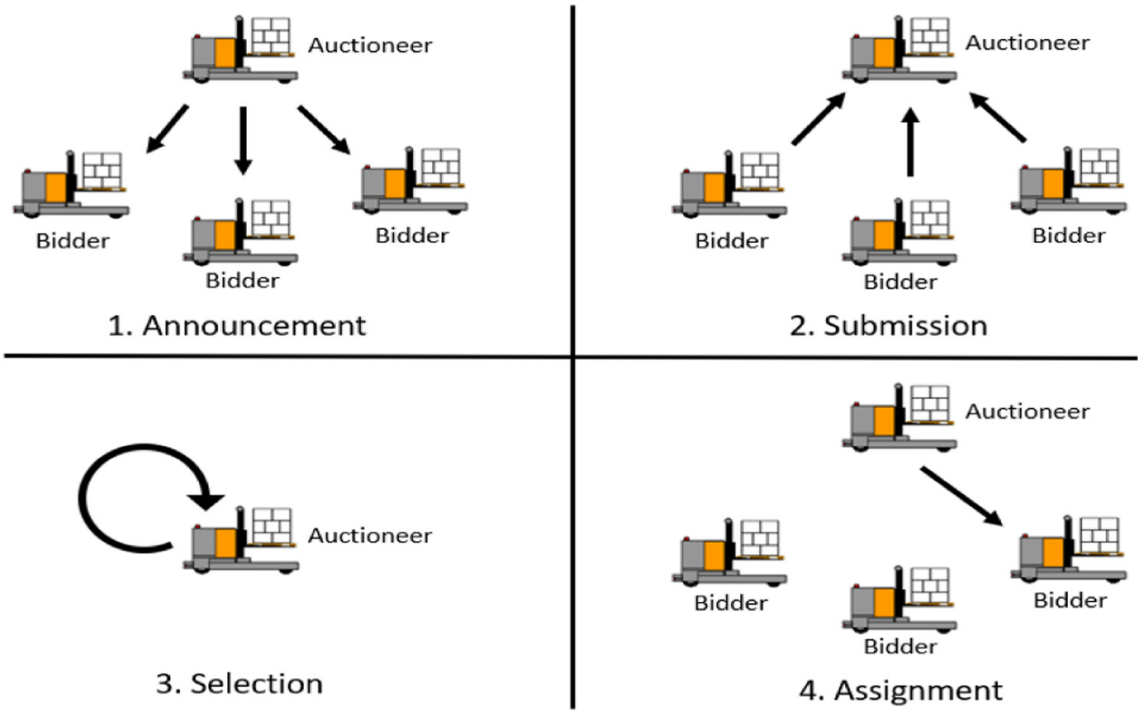
Durante a otimização, a PD simplifica uma decisão, dividindo esta decisão global em decisões menores e mais simples, seguindo o princípio de 'dividir e conquistar', devido à divisão do problema global, a programação dinâmica parece ter maior capacidade de encontrar uma solução mais otimizada do que as heurísticas. Entretanto, quando são utilizadas frotas muito grandes, manter um desempenho eficiente do sistema pode acabar se tornando desafiador [17].

2.1.2.2 Métodos baseados no mercado

Nas soluções baseadas no mercado, um princípio econômico é empregado para resolver o problema de alocação de tarefas, desta forma, a divisão não é realizada através da execução de um processo de otimização que usa todas as informações disponíveis, mas sim por um método específico de leilões, onde cada robô utiliza suas informações locais para dar lances [32]. Uma versão fundamental desse princípio é o protocolo CNET, onde um leiloeiro anuncia uma única tarefa por vez aos compradores, cada um dos quais oferece um lance pela tarefa, com base no custo de execução dessa tarefa para o comprador. O leiloeiro avalia todos os lances, incluindo o seu próprio, e então aloca a tarefa ao robô

com o lance mais alto [39]. Esse método é conhecido como uma solução gulosa, pois escolhe a solução melhor e mais simples, e após um AGV ser alocado para uma tarefa, ele não pode mais dar lances. O processo pode ser visualizado na Figura 4. Essa é a forma mais simples das abordagens baseadas no mercado, e outras variantes são apresentadas na Tabela 2.

Figura 4: Representação dos passos de um leilão



Fonte: De Ryck, Versteyhe e Debrouwere [17]

2.1.2.2.1 Restrições das soluções baseadas no mercado

Soluções baseadas no mercado também enfrentam os desafios relacionados a restrições, existindo diversas variações dos algoritmos mencionados anteriormente que levam essas restrições em consideração. Uma destas variações é o algoritmo TeSSI, apresentado por Nunes e Gini [51]. No contexto de um leilão simples, os agentes podem dar lances em mais de uma tarefa simultaneamente, no entanto, enquanto fazem seus lances, cada robô leva em consideração restrições temporais usando uma Rede Temporal Simples (STN), que é a agenda individual de cada robô, buscando um espaço vazio nessa agenda para uma nova tarefa antes de dar seus lances.

A principal vantagem dos métodos baseados no mercado é o fato de ser uma abordagem descentralizada, o que permite computações mais complexas em cada dispositivo sem forçar seus limites computacionais, viabilizando a adição de mais restrições nos cálculos dos lances nas tarefas ou alocação das tarefas. Em uma arquitetura centralizada, um

grande número de restrições pode acabar sobrecarregando a unidade central, responsável pela computação, ao realizar a otimização, o que acaba aumentando as chances de não se encontrar a solução ótima de maneira significativa [17].

A Tabela 2 levanta os algoritmos baseados em otimização que foram discutidos, trazendo de uma forma resumida suas vantagens e desvantagens brevemente apresentadas. A Tabela 3 traz algoritmos baseados no mercado, ressalta-se que alguns dos algoritmos apresentados não foram discutidos a fim de manter uma certa brevidade neste trabalho e também por serem variações do algoritmo que foi apresentado e discutido nesta seção.

Algoritmo	Vantagens	Desvantagens
Algoritmos Exatos	Encontra a solução ótima para frotas bem pequenas	Usável apenas em frotas bem pequenas
Heurísticas	Encontra a solução aproximada para frotas pequenas	Performance baixa em frotas grandes e complexas
Meta-Heurísticas	Encontra a solução aproximada para frotas médias	Alto custo computacional e temporal
Hiper-Heurísticas	Encontra a solução aproximada para frotas médias	Não flexível, não robusta e não escalável
Programação Dinâmica	Encontra a solução aproximada para frotas médias	Não flexível, não robusta e não escalável

Fonte: De Ryck, Versteyhe e Debrouwere [17]

Tabela 2: Vantagens e Desvantagens dos diferentes algoritmos baseados em otimização

Algoritmo	Vantagens	Desvantagens
Leiloeiro Central	Boa ponte entre arquiteturas centralizadas e descentralizadas	Ponto singular de Falha
Leiloeiro Flutuante	Nenhum ponto singular de falha (Robusto)	Algoritmo de troca de leiloeira mais complexo
Leilão de item único sequencial	Algoritmo simples, introduz flexibilidades e escalabilidade	Muitas soluções subótimas
Leilão de item único paralelo	Algoritmo simples, introduz flexibilidades e escalabilidade	Muitas soluções subótimas
Leilão combinatório	Soluções quase ótimas, considera sinergias	Pesado computacionalmente
Participação Restrita	Robô só precisa se preocupar com uma única tarefa	Falta de flexibilidade, não considera sinergias
Participação não-restrita	Robôs podem otimizar a lista de tarefa localmente, considera sinergias	Robôs precisam se preocupar com mais de uma tarefa por vez
Cálculo de lance simples	Cálculos simples reduz a necessidade computacional	Cálculo de lances não representa os custos reais
Cálculo de lance marginal	Representa os custos reais, considera sinergias	Nenhuma
Leilões restritos por tempo	Descrição do problema próximo ao mundo real	Computação mais complexa
Leilões restritos por precedência	Descrição do problema próximo ao mundo real	Computação mais complexa
Leilões restritos por recursos	Descrição do problema próximo ao mundo real	Computação mais complexa
Leilões com consenso	Aumenta a performance	Necessita de cooperação mais complexa

Fonte: De Ryck, Versteyhe e Debrouwere [17]

Tabela 3: Vantagens e desvantagens dos diferentes algoritmos baseados no mercado

2.2 Aprendizado de Máquina

Em 1959, Arthur Samuel definiu o aprendizado de máquina, do inglês Machine Learning (ML), em seu artigo "Some Studies in Machine Learning Using the Game of Checkers", como o "campo de estudo que dá aos computadores a capacidade de aprender sem serem programados de forma explícita". A essência do ML reside na exploração e construção de algoritmos capazes de aprender com seus erros e realizar previsões com base nos dados. Esses algoritmos constroem modelos a partir de entradas amostrais, buscando fazer previsões ou tomar decisões orientadas por essas entradas, indo além das simples instruções programadas. Enquanto a inteligência artificial abrange duas grandes áreas de raciocínio (indutivo e dedutivo), o aprendizado de máquina se concentra exclusivamente no raciocínio indutivo [64].

Mitchell [45] oferece uma definição formal comumente citada: "Diz-se que um programa de computador aprende pela experiência(E), com respeito a algum tipo de tarefa (T) e performance (P), se P em T, conforme medida, melhora com E."Essa definição das tarefas envolvidas no aprendizado de máquina é gerada de maneira operacional, em contraste com uma abordagem cognitiva. Isso substitui a pergunta proposta por Alan Turing em seu artigo "Computadores e inteligência", que indaga se as máquinas são capazes de pensar, por "Será que as máquinas possuem a capacidade de realizar o que nós, enquanto entidades pensantes, podemos fazer?"[45].

Os tipos de aprendizado utilizados em ML podem ser normalmente classificados em três categorias, dependendo da natureza do "sinal"ou "feedback"disponível para o sistema. Sendo eles:

- Aprendizado supervisionado: São apresentados ao computador exemplos de entradas e saídas desejadas, sendo o objetivo deste tipo de aprendizado, a compreensão de uma regra geral que mapeia um "caminho"das entradas para as saídas.

- Aprendizado não supervisionado: Não são dadas etiquetas ao algoritmo de aprendizado, deixando este sozinho para encontrar a lógica e estrutura nas entradas fornecidas.

- Aprendizado por reforço: Um algoritmo interage com um ambiente, onde deve cumprir um determinado objetivo (por exemplo, alocar tarefas dinamicamente). É fornecido, ao programa, reforços positivos ou negativos, na medida em que o espaço do problema é navegado.

As tarefas de ML podem ser classificadas quanto a saída desejada:

- Classificação: Divide as entradas do sistema em duas ou mais categorias

- Regressão: Modela as relações entre variáveis dependentes e independentes através de métodos estatísticos.

- Agrupamento: Entradas são divididas em diferentes grupos de "afinidades".

- Estimativa de densidades: Encontra a distribuição das entradas em um espaço multidimensional.

- Redução dimensional: Simplifica as entradas ao traduzi-las para um espaço de menor dimensão.

2.2.1 Aprendizado Profundo

Conforme Deng e Yu [18], aprendizado profundo, ou *Deep Learning* (DL), representa uma classificação de técnicas do campo do Aprendizado de Máquina (ML), estando situado na interseção de diversas disciplinas da Tecnologia da Informação (TI) como a modelagem gráfica, otimização, processamento de sinais, reconhecimento de padrões e redes neurais. O aumento significativo na popularidade do aprendizado profundo que tem ocorrido nos últimos anos, foi impulsionado pelo grande aumento na capacidade computacional, especialmente com o uso de placas de vídeo (*GPUs*), que oferecem uma alternativa de hardware mais eficiente, junto com os avanços substanciais na pesquisa em aprendizado de máquina e processamento de informações [18].

Como mencionado por Pereira [56], a execução eficiente dos treinos, obtida através do uso das GPUs, ocorre graças às numerosas operações matemáticas envolvidas no treinamento das redes neurais profundas, que fazem extenso uso de vastas matrizes. Operações como essas demandam um grande tempo de computação nos processadores comuns, enquanto as GPUs são capazes de realizar essas tarefas de forma muito mais eficiente.

A expressão *aprendizado profundo* fica mais compreensível quando comparada com o termo "Rede Neural Artificial"(RNA), o que contribuiu para sua popularização. A terminologia "profundo"refere-se à quantidade de camadas que uma RNA possui, enquanto uma RNA "comum"possui uma ou duas camadas escondidas, uma rede profunda pode incluir um número muito maior de camadas, dependendo de sua arquitetura e aplicação específica. Um conjunto de técnicas de DL permitiu ultrapassar o limite tradicional do número máximo de camadas em uma RNA, solucionando desafios como o *vanishing gradient problem* (Problema do Desaparecimento de Gradiente) e possibilitando o treinamento de RNAs com um maior número de camadas escondidas. Algumas dessas técnicas são abordadas a seguir.

2.2.1.1 ReLu

A função de ativação não linear, *Rectified Linear Unit* (ReLu), criada por Nair e Hinton [49] e segue a seguinte equação:

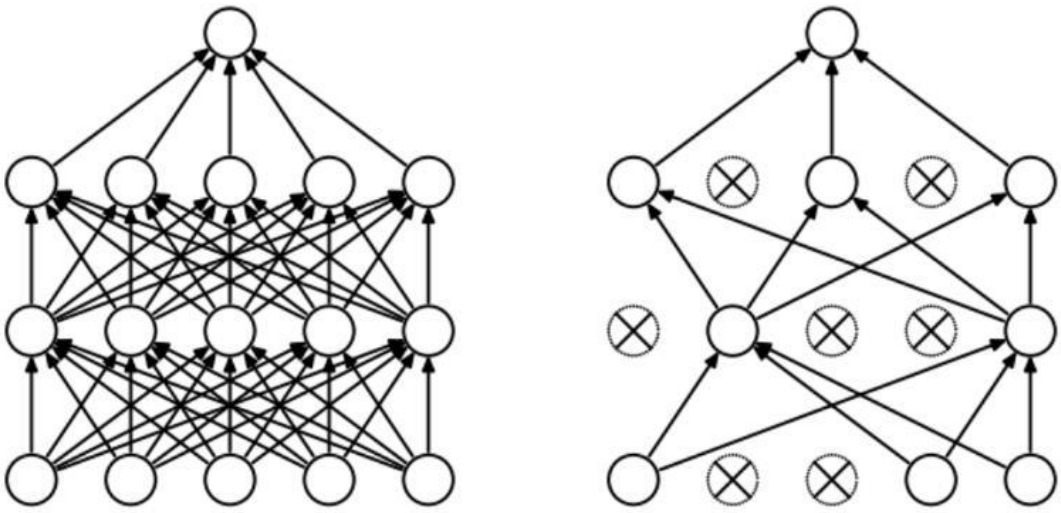
$$f(x) = Max(0, x)$$

onde $Max(0, x)$ é uma função que retorna 0 quando $x < 0$, e x quando $x > 0$. Essa é uma função amplamente utilizada em ML, devido à sua rapidez de cálculo quando comparada a outras funções de ativação não lineares comuns, como a função sigmóide, representada por $\frac{1}{1+e^{-x}}$ que retorna um valor entre 0 e 1, ou então, a função tangente hiperbólica, dada por $tanh(x) = \frac{e^{2x}-1}{e^{2x}+1}$ e que retorna um valor de -1 a 1. Pela simplicidade computacional de uma ReLU, o tempo de treinamento acaba sendo bastante reduzido, o que facilita o treinamento de redes com um número maior de camadas [31].

2.2.1.2 *Dropout*

Como afirmado por Srivastava et al. [67], o *Dropout* é uma técnica bastante empregada para acelerar o treinamento de RNAs com múltiplas camadas, a técnica consiste na desativação aleatória de alguns dos neurônios da Rede, bem como das conexões destes durante o processo de treinamento, visando prevenir o fenômeno conhecido como *overfitting*, onde a RNA treinada aprende a resolver o problema apresentado apenas no conjunto de dados utilizado para seu treinamento, resultando em uma rede incapaz de generalizar o conhecimento adquirido, dessa forma não conseguindo resolver outros problemas. Em uma rede mais profunda, é possível aprender e representar modelos consideravelmente mais complexos. A Figura 5 ilustra o funcionamento do *Dropout*.

Figura 5: Exemplo do funcionamento do *Dropout*



Fonte: Srivastava et al. [67]

2.2.1.3 *Backpropagation*

A Retropropagação, mais conhecida como *Backpropagation*, abreviatura de "Retropropagação de erros", foi concebida por Linnainmaa [40] na década de 70, mas se popularizou em meados dos anos 80 por meio do trabalho de Rumelhart, Hinton e Williams [63]. O *Backpropagation* é um algoritmo de ML que faz uso do método de gradiente, onde em uma RNA e sua função de erro, o algoritmo calcula o gradiente da função de erro em relação aos pesos da rede, sendo isto uma generalização da lei delta dos perceptrons para RNAs *feedforward* multicamada.

A parte 'retro' do nome tem origem no fato de que o cálculo do gradiente ocorre de maneira retroativa através da rede, começando pelo gradiente da última camada de pesos e progredindo até o gradiente da primeira camada. As computações parciais do gradiente de

uma camada são reutilizadas para o cálculo do gradiente da camada anterior, esse fluxo de informação do erro permite o cálculo eficiente do gradiente em cada camada, ao contrário de uma abordagem ingênua, onde o gradiente de cada camada é calculado separadamente. De uma forma geral o algoritmo de retropropagação segue os seguintes passos:

1. Inicialização. Inicia os pesos de cada camada da RNA aleatoriamente ou conforme algum método;

2. Processamento direto. É apresentado um padrão à rede, todos os neurônios da rede são ativados e o erro é calculado;

3. Passo reverso. Os novos pesos dos neurônios da RNA são calculados, da saída para a entrada, camada a camada;

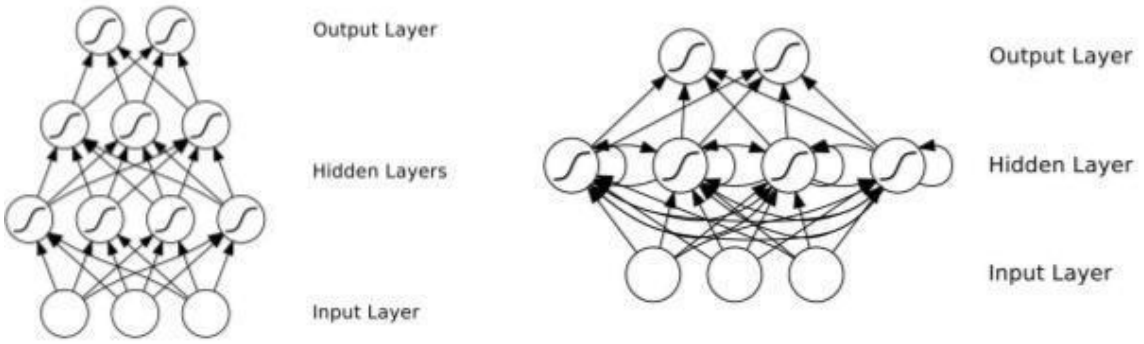
4. Teste de parada. É realizado um teste para verificar se o critério de parada foi atingido, se sim, o algoritmo é terminado, se não recomece do passo 2.

2.2.2 Redes Neurais Recorrentes

As Redes Neurais Recorrentes (RNNs), também conhecidas como *Recurrent Neural Networks*, representam uma categoria de redes neurais artificiais em que as conexões entre os neurônios de cada camada formam um ciclo direcionado, sendo bastante reconhecidas e demonstrando muito potencial em diversas das áreas de aprendizado de máquina, como processamento de linguagem natural, geração automática de legendas para imagens, tradução e reconhecimento de voz.[7]

As RNNs são denominadas recorrentes porque executam a mesma tarefa para cada elemento da entrada, com a saída dependendo dos resultados anteriores. A Figura 6 fornece uma comparação entre uma rede *feedforward* e uma rede recorrente para demonstrar esse conceito. Como destacado por Pereira [56], o conceito fundamental subjacente às RNNs é a capacidade de lidar com informações sequenciais, o que contrasta com as redes neurais convencionais, que presumem que todas as entradas e saídas são independentes, as RNNs se destacam em situações em que a dependência sequencial é crucial. Por exemplo, ao prever a próxima palavra em uma frase, é essencial considerar o contexto das palavras anteriores.

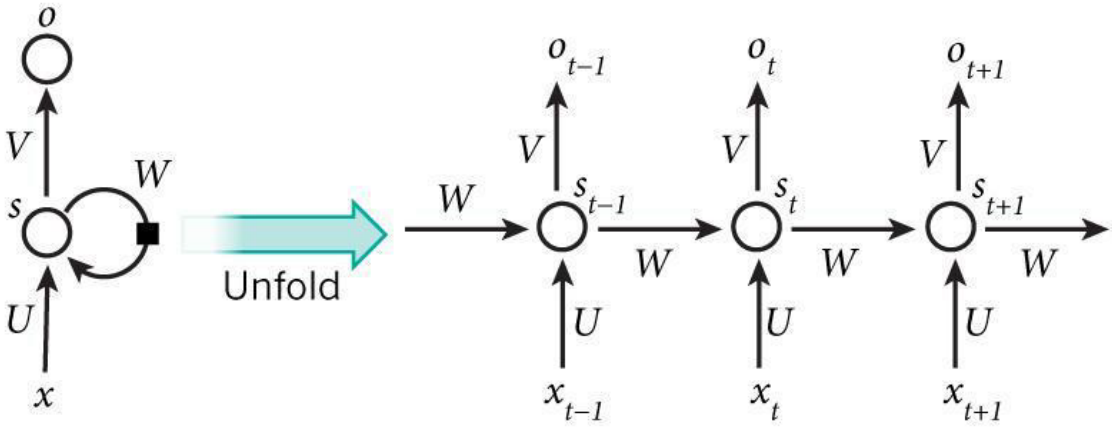
Figura 6: Comparativo entre uma rede *feedforward* e uma rede recorrente



Fonte: Sapunov, G. (2016) Disponível em: https://www.slideshare.net/grigorysapunov/multidimensional-rnn

Uma maneira simplificada de compreender as RNNs é concebê-las como redes que possuem uma "memória"capaz de reter informações sobre os cálculos realizados até o momento. O diagrama na Figura 7 representa o desdobramento de uma RNN em uma rede completa, destacando a sequência da rede, transformando, por exemplo, uma sequência de cinco estados sequenciais em uma rede neural com cinco camadas, cada uma correspondendo a uma dos estados.

Figura 7: Desdobramento de uma Rede Neural Recorrente



Fonte: Britz [7]

A descrição dos símbolos da Figura 7 é como seguem:

- X_t é a entrada no passo de tempo t;
- W são os pesos das camadas escondidas;
- U são os pesos da entrada;

- V é o vetor de saída;
- S_t é um estado escondido no passo de tempo t, calculado baseado no estado escondido anterior e na entrada do passo atual: $S_t = (U_x + W_{St-1})$
- S_{t-1} é utilizado para calcular o primeiro estado escondido, sendo geralmente iniciado com 0;
- O_t é o vetor de saída V no passo t, após a aplicação da função de ativação.

A parte mais importante do diagrama da Figura 7 é o estado escondido S_t, correspondente à memória da rede, que guarda informações sobre o ocorrido nos passos anteriores.

O processo de treinamento de RNN possui grandes semelhanças com o treinamento de uma RNA convencional, fazendo uso do algoritmo de retropropagação, contudo uma modificação crucial é incorporada para considerar o caráter temporal e sequencial da rede, resultando na utilização de um algoritmo adaptado denominado retropropagação através do tempo (*Backpropagation Through Time* - BPTT) [7].

Um desafio comum enfrentado pelas RNNs diz respeito às "dependências de longo prazo", isto é, dependendo da natureza do problema que uma RNN busca resolver, é essencial que ela acesse as informações mais recentes para desempenhá-la de uma maneira eficiente, entretanto, quando a informação necessária tem sua origem em períodos muito distantes, a habilidade da RNN em aprender essa informação corretamente acaba ficando comprometida, e quanto mais velha forem essas informações, mais desafiador se torna para a RNN incorporar essa informação em seu aprendizado. Uma estratégia eficaz para contornar o obstáculo das dependências de longo prazo é a implementação de Redes de Memória de Curto Prazo Longa, conhecidas como *Long Short Term Memory* (LSTM) [31]. A seção subsequente aborda e discute brevemente as redes LSTM.

2.2.3 Long Short Term Memory

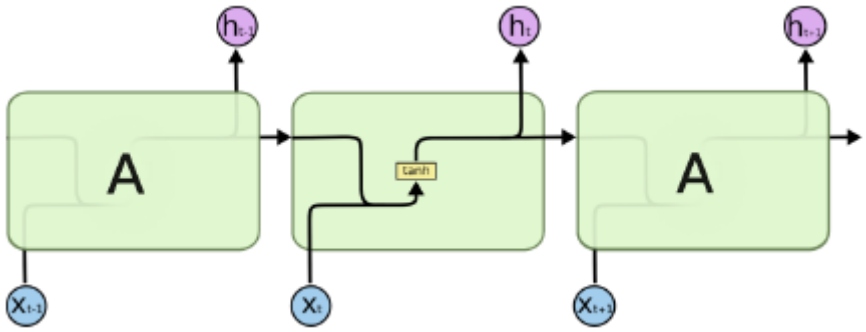
As Redes LSTM, representam uma categoria especial das RNNs com uma capacidade para capturar relações de interdependência a longo prazo, inicialmente apresentadas por Hochreiter e Schmidhuber [27] na sua forma moderna, as redes LSTM apresentam desempenho excepcional em uma grande número de aplicações, tornando uma escolha comum na comunidade de aprendizado de máquina contemporânea.

O autor optou por utilizar esta arquitetura, devido principalmente a robustez da mesma, que permite com que ela seja capaz de lidar com problemas bastante complexos, como é o caso do problema abordado neste estudo. Outro fator que influenciou na escolha pela LSTM para este estudo, é a familiaridade que o autor possui com esta rede e suas aplicações.

Ao contrário de RNNs convencionais, as LSTMs apresentam uma arquitetura de camadas com uma arquitetura especial, unindo elementos diferentes, como unidades de me-

mória e portas de ativação. Enquanto todas as RNNs compartilham a estrutura de camadas na forma de uma cadeia de módulos repetidos, a diferença crucial reside na complexidade desses módulos, em uma RNN padrão, o módulo possui uma estrutura relativamente simples, exemplificada por uma camada com a função de ativação $tanh$, conforme ilustrado na Figura 8.

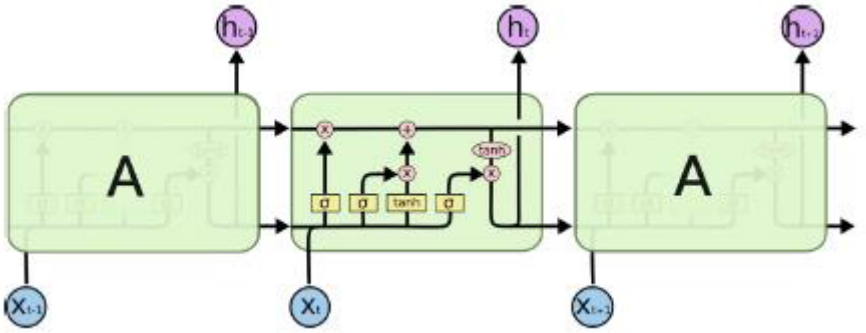
Figura 8: Repetição de módulo em uma RNN convecional



Fonte: OLAH [52]

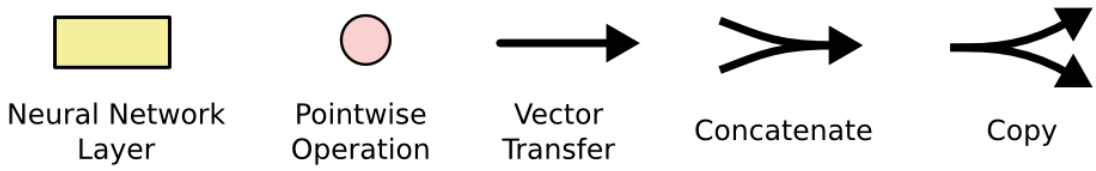
As Redes LSTM mantêm a estrutura em cadeia semelhante à das RNNs convencionais, mas o módulo repetido é configurado de maneira única, ao invés de consistir em apenas uma única camada, as LSTMs possuem quatro camadas interagindo de maneira altamente especializada. A Figura 9 ilustra esse conceito de forma mais detalhada com a Figura 10 apresentando a notação utilizada.

Figura 9: Módulo representativo de uma LSTM, que contém quatro camadas que interagem



Fonte: OLAH [52]

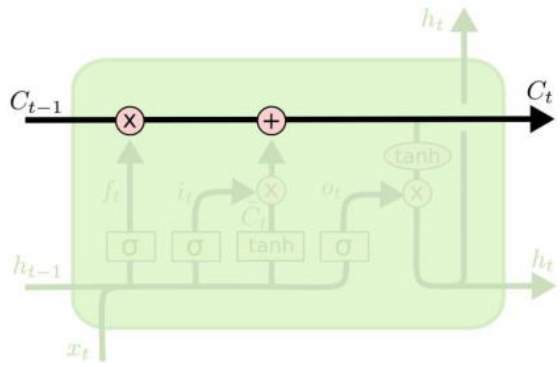
Figura 10: Notação utilizada no exemplo. Lê-se da esquerda para direita: Camada da rede neural, operação ponto a ponto, transferência de vetor, concatenar, copiar.



Fonte: OLAH [52]

A inovação central das LSTMs reside na introdução de uma representação do estado da célula através do tempo, visível como a linha horizontal destacada na parte superior da Figura 11. Este estado da célula percorre toda a sequência temporal, passando por interações lineares que permitem que a informação flua com poucas mudanças substanciais servindo, então, como uma memória para a rede.

Figura 11: Estado da célula, representado pela linha horizontal no topo dela

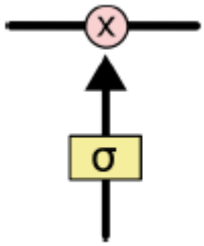


Fonte: OLAH [52]

Além disso, as LSTMs incorporam mecanismos para controlar a adição ou remoção de informações do estado da célula, realizando essa operação por meio de portões que atuam como reguladores do fluxo de informações na rede neural e sendo compostos por uma camada sigmoidal combinada com uma operação de multiplicação, a Figura 12 esclarece este conceito.

A camada sigmoidal gera saídas situadas no intervalo de 0 a 1, representando o grau de permissão para a passagem de cada um dos componentes pelo portão, em outras palavras, quanto maior o valor resultante, maior a quantidade de informação que passa pelo portão. Em uma rede LSTM, três destes portões estão presentes, desempenhando um papel crucial na manutenção e regulação do estado da célula.

Figura 12: Exemplo de um portão com uma camada sigmóide de uma rede neural e uma operação de multiplicação pontual



Fonte: OLAH [52]

2.2.4 Aprendizado-Q

O Aprendizado-Q (*Q-Learning* - QL) é uma técnica de aprendizado por reforço muito empregada no campo de ML, visando adquirir e aprimorar uma política que orienta um agente sobre quais ações devem ser tomadas em cada situação. O que dá um maior destaque ao QL é a sua capacidade de operar sem a necessidade da construção de um modelo detalhado do ambiente, sendo adequado para lidar com problemas que envolvem transições e recompensas estocásticas, sem a necessidade de ajustes complexos [31].

Em qualquer Processo de Decisão de Markov Finito (FMDP), o QL demonstra a capacidade de convergir para uma política que seja ótima, ou seja, aquela que maximiza o valor esperado da recompensa total ao longo de uma sequência de ações, começando a partir do estado atual. O QL se destaca pela habilidade de identificar uma política ótima de seleção de ações para qualquer FMDP, desde que haja um tempo infinito para exploração e uma política parcialmente aleatória. O termo "Q"representa a função que estima a recompensa, utilizada como guia para o processo de reforço, sendo interpretada como uma medida de "qualidade"da ação realizada em um estado específico [60]. Conforme classificado por Queiroz [60] as seções seguintes apresentam as variáveis importantes.

2.2.4.1 Exploração versus Aproveitamento

A taxa de exploração, também conhecida como passo de aprendizado, é crucial para determinar como as informações mais recentes impactam as informações já existentes, quando o fator de aprendizado é fixado em 0, o agente não incorpora os novos conhecimentos, isto é, há exploração mínima. Já um fator de aprendizado 1 leva o agente a dar maior importância às informações mais novas, priorizando o aproveitamento das descobertas atuais. Em ambientes determinísticos, onde os resultados são previsíveis, uma taxa de exploração igual a 1 geralmente é considerada a escolha ideal. Entretanto, em ambientes estocásticos, onde existe aleatoriedade, a convergência do algoritmo pode demandar uma redução gradual da taxa de aprendizado. Na prática, é comum utilizar uma taxa de aprendizado constante, mesmo em ambientes estocásticos, o que simplifica o processo

computacional. A escolha adequada da taxa de exploração depende da natureza específica do problema, sendo crucial considerar a interação entre exploração e aproveitamento para otimizar o desempenho do algoritmo em diferentes contextos.

2.2.4.2 Fator de Desconto

O papel do fator de desconto reside na definição da importância atribuída as recompensas futuras, em algoritmos de ML quando o fator de desconto é estabelecido em 0, o agente adota uma abordagem "miópica", focando exclusivamente nas recompensas imediatas, já um valor mais próximo de 1 incentiva o agente a buscar recompensas de longo prazo. Entretanto, se o fator de desconto for maior que 1, os valores das ações podem divergir, especialmente em cenários sem estados terminais ou quando o agente não consegue alcançar os estados de término, os históricos de ambiente se estendem infinitamente, e as utilidades associadas a recompensas acumulativas tendem ao infinito.

Mesmo com um desconto ligeiramente abaixo de 1, o uso do aprendizado da função Q pode resultar na propagação de erros e instabilidades, especialmente ao empregar aproximações da função valor, nestes casos iniciar com um fator de desconto menor e, gradualmente aumentá-lo em direção ao valor desejado pode acelerar o processo de aprendizado. Essa abordagem pode mitigar possíveis divergências e contribuir para a estabilidade do algoritmo, principalmente em situações onde o horizonte temporal é indefinido. A seleção cuidadosa do fator de desconto é essencial para garantir uma ponderação adequada entre recompensas imediatas e futuras, otimizando o desempenho do agente em ambientes dinâmicos e complexos.

2.2.4.3 Condições Iniciais

Dado que o QL é um algoritmo iterativo, ele pressupõe uma condição inicial antes de realizar a primeira atualização, a estratégia em que são atribuídos valores iniciais altos, pode vir a promover a exploração, assim, independente da ação inicial escolhida, o processo de atualização levará a valores menores para essa ação quando comparada com as alternativas, aumentando a probabilidade de sua escolha. A primeira recompensa r é frequentemente empregada para redefinir as condições iniciais, seguindo essa abordagem, na primeira ocorrência de uma ação, a recompensa associada é utilizada para estabelecer o valor de Q, permitindo uma aprendizagem imediata, especialmente em cenários de recompensas determinísticas e fixas.

A introdução de um modelo que incorpora o conceito de redefinição das condições iniciais é esperada para oferecer uma previsão mais precisa do comportamento dos agentes, quando comparada a um modelo que assume condições iniciais arbitrárias. Essa abordagem busca melhorar a capacidade do algoritmo de aprendizado em se adaptar a diferentes ambientes, maximizando a eficácia da exploração e otimizando o processo de atualização do QL.

2.2.4.4 Quantização

A fim de simplificar e reduzir o espaço de ações viáveis, é comum empregar a quantização, onde diversos valores são atribuídos a intervalos específicos, permitindo uma representação mais densa dos estados. Usando como exemplo o desafio de aprender a equilibrar uma vassoura em um dedo, a descrição de um estado em um determinado momento envolve vários atributos, como a posição espacial do dedo, sua velocidade, o ângulo do bastão e a velocidade angular do bastão, o que resulta em um vetor composto por quatro valores, que caracterizam um estado específico, isto é, o estado em um dado instante, representado pelos quatro elementos. Usando a quantização, ao invés de lidar com uma infinidade de possíveis valores para a posição espacial do dedo, podemos agrupá-los em intervalos discretos, simplificando o problema.

Essa abordagem não apenas alivia a carga computacional, mas também facilita o processo de aprendizado, pois o agente passa a lidar com um conjunto mais gerenciável de estados ao invés de um espectro contínuo e amplo. A quantização, portanto, emerge como uma estratégia prática para enfrentar desafios da alta dimensionalidade, promovendo uma representação mais eficiente e manejável dos estados em ambientes complexos.

2.3 Sumário do capítulo

Neste capítulo, foram abordados e discutidos os dois temas centrais deste trabalho, ambos muito extensos e bastante complexos, no entanto, a abordagem adotada foi focar nos pontos relevantes para este trabalho, com o intuito de proporcionar uma compreensão mais pontual, destacando aspectos que têm uma ligação direta e significativa com os objetivos desta pesquisa. O próximo capítulo apresenta uma revisão literária abrangente, explorando artigos relevantes relacionados ao tema tratado, posteriormente, no capítulo subsequente, são apresentados e discutidos alguns resultados preliminares alcançados até o momento da redação desta monografia.

3 TRABALHOS RELACIONADOS

A partir da discussão apresentada nos capítulos anteriores, fica evidente que o problema da distribuição de tarefas, sendo NP-difícil, apresenta uma complexidade que apesar das diversas abordagens disponíveis para a sua resolução, cada solução potencial requer uma análise para sua adaptação ao mundo real. Dessa forma, técnicas de aprendizado de máquina, principalmente RNAs, sendo ferramentas versáteis, não apenas tem o potencial para oferecer suporte às abordagens existentes, mas também de constituir soluções eficazes para o problema em questão.

Neste estudo, é abordada a aplicação de uma RNA baseada em aprendizado por reforço, utilizando técnicas de QL, com o objetivo de proporcionar suporte às soluções existentes, simplificando a complexidade dos estudos necessários para a eficaz implementação de soluções e algoritmos no ambiente real.

Para encontrar os trabalhos relacionados, utilizou-se de ferramentas de busca do *google scholar* e *Arxiv* bem como pesquisas nos sites de editoras como *Elsevier* e IEEE. Para a realização das buscas, foram utilizadas as palavras-chave deste trabalho (Aprendizado de Máquina, Aprendizado Profundo, Aprendizado-Q e Distribuição de Tarefas), utilizando diferentes combinações destas, assim como de forma avulsa, delimitando os resultados para publicações a partir do ano de 2018, a fim de manter uma base de conhecimento mais próxima do estado da arte atual.

Na seção a seguir, são apresentados alguns desses trabalhos, ao final deste capítulo, uma tabela compilará todos os artigos relacionados a este trabalho, delineando seus objetivos e incluindo breves reflexões do autor sobre o conteúdo de cada um. Esse levantamento pretende contribuir para preencher a lacuna existente na pesquisa ao destacar a aplicabilidade e eficácia das técnicas de aprendizado por reforço, especialmente no contexto do campo da distribuição de tarefas.

3.1 Principais Trabalhos

Cai, Harasha e Lynch [13] em artigo publicado no ano de 2022, apresenta um framework genérico para a modelagem de algoritmos de enxame, o qual opera sob algumas

premissas, como a presença de um conjunto finito de agentes movendo-se em uma grade retangular. Desenvolvido em Python e utilizando a biblioteca Pygame para simulações, este framework serviu como base para experimentos utilizando dois algoritmos de enxame distintos: o algoritmo House Hunting Task Allocation (HHTA), o algoritmo de propagação de tarefas (PdT) e o algoritmo de caminhada aleatória (CA).

Os resultados obtidos pelos autores revelam que, quando a demanda total para os agentes é constante, a densidade de tarefas exerce uma influência significativa em ambos algoritmos. Já em cenários com tarefas dispersas, o HHTA demonstra um desempenho superior em relação ao CA, de outro lado, o PdT apresenta um desempenho superior ao CA quando o número de tarefas é elevado. Esses achados destacam a sensibilidade dos algoritmos à distribuição espacial das tarefas, fornecendo insights valiosos sobre o desempenho relativo dessas abordagens em diferentes contextos.

Com o objetivo de minimizar o número de tarefas não concluídas ao final do horizonte operacional, Choudhury et al. [15] apresenta um algoritmo de alocação multi-robô que desvincula de maneira hierárquica a incerteza fundamental, a coordenação multiagente e os destinos. O algoritmo denominado *Stochastic Conflict-Based Allocation* (SCoBA) busca enfrentar esses desafios em uma abordagem hierárquica de dois níveis, superior e inferior.

No nível inferior, a sequência ótima de tarefas para cada agente é determinada de maneira independente, desconsiderando as ações dos outros agentes, já no nível superior, são resolvidos os potenciais conflitos originados na distribuição anterior, resultando em uma alocação multiagente válida. Teoricamente, o SCoBA é um algoritmo ótimo em termos de expectativa e abrange uma gama significativa de situações, desde que o sistema onde for utilizado não tenha uma alta complexidade.

Na aplicação prática, em dois domínios distintos e realistas, o SCoBA exibe um desempenho sólido e competitivo, consistentemente superando várias referências e, além disso, mostra-se viável em termos de tempo de computação, tanto para os agentes quanto para as tarefas. Esses resultados validam a eficácia do SCoBA como uma ferramenta promissora para abordar desafios complexos na alocação de tarefas em ambientes dinâmicos e incertos.

Uma técnica que tem ganhado popularidade recentemente é a adoção de um gêmeo digital, onde uma simulação, conhecida como gêmeo digital, é criada para replicar em tempo real a situação do sistema físico, sendo esta simulação a base para o cálculo da distribuição de tarefas. Antoniuk et al. [2] e Wang, Jiang e Yue [70], utilizam esta metodologia em seus artigos, com Antoniuk et al. [2] buscando aprimorar a eficiência da logística interna, e apresentando uma metodologia fundamentada nos conceitos e ferramentas da Indústria 4.0. Essa abordagem propõe a criação de um sistema de logística autônoma, com destaque para a utilização de um algoritmo estratégico. Este algoritmo também faz uso de uma metodologia logística conhecida como ABC, onde produtos ou

tarefas são classificados de acordo com o princípio de pareto, de acordo com a frequência da necessidade destes itens, com os mais importantes sendo itens A, seguidos por B e C respectivamente [20]. Este algoritmo orienta o processo de otimização por meio dos seguintes passos:

1. Analisar o estado inicial;
2. Classificar conforme metodologia ABC os produtos a serem transportados;
3. Selecionar os AGVs e periféricos apropriados;
4. Desenvolver o circuito logístico;
5. Avaliar estaticamente o sistema logístico;
6. Verificar e otimizar via simulação;
7. Criar o gêmeo digital.

Em artigo publicado no ano de 2021, Wang, Jiang e Yue [70] emprega a técnica do gêmeo digital para conceber um sistema de distribuição de materiais, utilizando uma rede LSTM para prever os tempos de demanda, foi implementado em uma linha de produção de cartões assíncronos, responsável por produzir diferentes tipos desses cartões para diversos sistemas de controle. Os resultados obtidos foram promissores, com uma média de erro na previsão da LSTM de apenas 4,01%. A análise da rota de distribuição de materiais mostrou uma redução de 40% no número de partidas dos veículos, um aumento de 36,68% na taxa de carga total, e uma redução de 24,45% no custo total de distribuição. Embora os dados demonstrem a eficácia do método proposto pelo autor, é importante observar que este método foi aplicado apenas na linha de produção de cartões assíncronos, isto é, o ciclo completo da distribuição de materiais não foi coberto, portanto estudos mais aprofundados são necessários para validar esses resultados em um contexto mais abrangente.

No contexto de seu artigo, publicado em 2022, Alitappeh e Jeddisaravi [1] explora o ambiente para a realização de tarefas distribuídas, visando minimizar o custo global do sistema. Para alcançar esse objetivo, é proposto um framework que constrói dois grafos a partir do mapa inserido no sistema, denominados mapa de Grade e mapa GVD, que desempenham papéis distintos, sendo o primeiro utilizado para a divisão de áreas e o segundo para o roteamento. No processo de implementação, as áreas delineadas no mapa de Grade são atribuídas a AGVs, cada um seguindo o roteamento gerado pelo mapa GVD. O autor adota duas metodologias distintas na busca pela solução ótima na distribuição de tarefas, uma delas emprega um algoritmo evolucionário que emprega uma estratégia elitista para descobrir soluções Pareto-ótimas para problemas multiobjetivos, sendo também eficiente no tratamento de várias restrições, denominado NSGA-II e a

outra se baseia no uso de QL, visando encontrar a solução ótima por meio da exploração do ambiente.

Para avaliar a eficácia das abordagens propostas, o autor implementa diversos métodos comparativos, alguns acabam se revelando inaplicáveis, dada a inadequação ao problema estudado. Contudo, as metodologias propostas demonstram melhores resultados na maioria dos mapas utilizados, sugerindo que nos casos analisados, as metodologias propostas apresentam uma compatibilidade e consistência superiores em comparação aos algoritmos utilizados como referência, respaldando a eficiência das abordagens propostas para a resolução do problema no ambiente em questão.

O artigo de Ferretti e Marchi [24], publicado em 2024, apresenta um modelo próximo ao apresentado neste estudo, sendo também uma RNA de aprendizado profundo utilizando aprendizado Q. Em seu artigo, o autor foca seus estudos no controle de estoque de um único produto em uma única parte da cadeia de suprimento, sendo este o controle de pedidos de matéria de um conjunto de máquinas. O modelo é comparado com uma das metodologias amplamente utilizadas atualmente, onde um pedido de ressuprimento é feito toda vez que o estoque disponível para alguma das máquinas encontra-se abaixo de um ponto determinado x. Os resultados obtidos pelo autor demonstram que, não somente o modelo proposto é uma solução viável, mas como também é uma solução mais eficiente.

O trabalho desenvolvido por Rahman e Nielsen [61], publicado no ano de 2019, desempenhou um papel fundamental no desenvolvimento do projeto proposto neste estudo, que devido à semelhança de seu cenário e aquele estudado nesta dissertação, serviu como fonte inspiradora para a formulação da *baseline* utilizada na validação da metodologia proposta pelo autor desta dissertação. No âmbito do artigo, é apresentada uma abordagem inovadora para resolver o problema de ordenação de tarefas em veículos de transporte autônomo.

O autor introduz dois algoritmos distintos, um deles baseado em técnicas de algoritmos genéticos, e o outro adotando uma abordagem iterativa gulosa. O cenário abordado no artigo é de um depósito portuário com duas unidades autônomas. No algoritmo genético, as filas de tarefas dos dois AGVs são utilizadas como cromossomos dos pais, passando por mutações para gerar a progênie. Já no algoritmo iterativo guloso, o processo se inicia com uma solução inicial, buscando aprimorá-la ao longo de quatro estágios: destruição, construção, melhorias e aceitação.

Na etapa de destruição, alguns de seus elementos são excluídos, gerando uma solução parcial que é, então, reconstruída na fase seguinte onde uma heurística construtiva gulosa guia a reintegração dos elementos excluídos, resultando na fase de melhorias. O processo é repetido até que uma condição de parada pré-definida seja alcançada, caracterizando a fase de aceitação, com a decisão de aceitar ou substituir a solução inicial baseada em algum critério pré-estabelecido. Os resultados obtidos no artigo são promissores, destacando as principais vantagens desses métodos na resolução do problema proposto, com

o trabalho oferecendo não apenas uma solução eficaz, mas também apontando possíveis direções de pesquisa futura.

Conforme será discutido na seção a seguir, durante a fase final deste estudo, houve a necessidade de uma nova busca por artigos, durante a qual o artigo de Bello et al. [6], publicado em 2017, se destacou e serviu de inspiração para o autor durante os experimentos finais deste trabalho. Neste artigo, os autores exploram um cenário próximo ao estudado durante o desenvolvimento desta dissertação. Em seu trabalho, os autores utilizaram duas redes LSTMs agindo em conjunto para solucionar o problema. Além das duas LSTMs parceiras, os autores também realizam uma avaliação dos resultados intermediários, os resultados atingidos pelos autores foram bastante positivos, superando até mesmo aqueles alcançados pelo algoritmo OR-Tools [57] da Google.

3.1.1 Considerações do autor

Conforme discutido previamente, é notável a falta de exploração do proposto neste trabalho, de utilizar RNAs como uma ferramenta para elevar o desempenho dos algoritmos existentes, havendo uma predominância muito grande do uso de RNAs como forma direta para encontrar soluções para o problema. Esta falta de estudos na área também foi notada por De Assis et al. [16], que durante seu estudo encontrou apenas 143 trabalhos e restando somente 26 destes após filtragem, na área referente à utilização de RNAs no processo de solução do PCV, sendo que dos trabalhos encontrados, sua grande maioria trata o ML como solução em si e não como uma possível ferramenta capaz de elevar as soluções já praticadas. Conforme citado, durante a exploração dos artigos existentes, há uma grande predominância da utilização de RNAs diretamente para buscar as soluções ótimas, como pode-se observar a partir dos artigos de Bello et al. [6], Ferretti e Marchi [24] e Wang, Jiang e Yue [70], demonstrando que vem sendo bastante explorado a aplicação das RNAs como alternativas para algoritmos já existentes.

3.2 Sumário do capítulo

O capítulo 3 apresentou uma análise abrangente de trabalhos relacionados ao tema abordado no texto, destacando uma seção dedicada aos principais estudos citados. Notavelmente, a pesquisa revelou uma lacuna na literatura existente, visto que foram encontrados poucos artigos específicos abordando o tema da aplicação de uma RNA como ferramenta de suporte para os algoritmos de resolução de problemas relacionados à distribuição de tarefas, com a maioria dos estudos revisados utilizando RNAs diretamente.

A seguir, a Tabela 4 resume de maneira concisa os objetivos e as opiniões do autor a respeito dos trabalhos relacionados, proporcionando uma visão consolidada dos elementos-chave discutidos ao longo do texto. Este levantamento de trabalhos correlatos e a abordagem singular da RNA para a distribuição de tarefas oferecem uma contribuição

inexplorada ao campo, destacando a originalidade e relevância do presente estudo, ao mesmo tempo em que ressalta a necessidade de mais investigações específicas nesse campo emergente.

Autor(es)	Título	Objetivos	Opinião do autor
Cai, Harasha e Lynch [13]	A Comparison of New Swarm Task Allocation Algorithms in Unknown Environments with Varying Task Density	Criar um framework genérico para a modelagem e comparação de diversos algoritmos de enxame.	Resultados promissores, mas necessita de estudos mais aprofundados, testando em mais situações e com mais de dois algoritmos.
Hussain, Belardinelli e Piliouras [28]	Asymptotic Convergence and Performance of Multi-agente Q-learning Dynamics	Demonstrar que existem casos onde o QL converge para uma solução não ótima com uma taxa de exploração não-zero.	Alcança seus objetivos, demonstrando essa convergência em certas situações, ótima base para estudos mais aprofundados no assunto.
De Ryck, Versteyhe e Debrouwere [17]	Automated guided vehicle systems, state-of-the-art control algorithms and techniques	Introduzir e exemplificar os atuais métodos e algoritmos estado da arte sendo utilizados no mercado para a solução de problemas envolvendo AGVs	Apresenta muito bem os diversos aspectos, com uma descrição sucinta mas compreensível e indica quais artigos para aprofundação de conhecimento.
Choudhury et al. [15]	Dynamic Multi-Robot Task Allocation under Uncertainty and Temporal Contraints	Minimizar o número de tarefas não sucedidas, através de um algoritmo que desacopla os aspectos de distribuição de tarefas	O algoritmo apresentado é promissor, mas é de minha opinião que ele deve ser testado de maneira mais extensiva e comparado com novas metodologias para se averiguar sua real eficácia.
Bruno e Antonelli [8]	Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells	Propôr um método de classificação das tarefas, partindo de uma decomposição hierárquica	O método parte de uma ideia surrealista em que os tempos de produção serão respeitados, o que é não ocorre, visto que há participação de um ser humano, estas incertezas devem ser levadas em contas em uma possível revisão deste método
Ivanov, Zisman e Chernyshev [29]	Mediated Multi-agente Reinforcement Learning	Resolver o problema de cooperação através de mediadores	Premissa interessante, com bons resultados, mas precisa ser explorada de maneira mais profunda a fim de verificar a real eficácia do método
Antoniuk et al. [2]	Methodology of design and optimization of internal logistics in the concept of Industry 4.0	Criar uma nova metodologia de planejamento e otimização de logística interna	O algoritmo desenvolvido não é nada novo, apenas colocando de uma forma mais "formal"o que é feito durante o desenvolvimento de um sistema produtivo, com a única novidade sendo a utilização de um gêmeo digital
Wang, Jiang e Yue [70]	Model Construction of Material Distribution System Based on Digital Twin	Criar um sistema de distribuição de material baseado ná técnica de gêmeo digital	Resultados muito promissores com ótimas oportunidades para futuros estudos e aprofundamentos
Liu, Chen e Huang [42]	Multi-agente Pathfinding Based on Improved Cooperative A in Kiva System	Apresentar um novo algoritmo baseado em cooperação, tomando o algoritmo A* como base	Resultados muito interessantes, principalmentes os insights sobre os métodos e algoritmos utilizados pela Kiva Systems, atual Amazon Robotics
Alitappeh e Jeddisaravi [1]	Multi-robot exploration in task allocation problem	Dividir o problema de divisão de tarefas em dois subproblemas	Alcançou seu objetivos com êxito, demonstrando que o método apresentado possui certas vanteagens sobre os métodos exatos utilizados atualmente.
Burganova et al. [9]	Optimalisation of Internal Logistics Transport Time Through Warehouse Management Case Study	Visa melhorar logística e armazenagem com métodos existentes utilizando uma quantia pequena de capital e diminuindo o tempo de transporte	Foi alcançada uma melhora significativa na produtividade e lucros, mas existem metodologias além das exploradas que podem auxiliar em melhorar ainda mais estes resultados, deixando aberta a porta para novos estudos.
Bänziger, Kunz e Wegener [12]	Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions	Aborda o uso de uma padronização da descrição de ordens de serviço para geração autônoma de unidades robóticas de auxílio	A proposta não garante a otimização, apesar de ter sido encontrada uma solução para o sistema retratado, esta requer uma diminuição do número de células de trabalho, o que remove qualquer oportunidade de escalabilidade, portanto novos estudos devem ser realizados.
Wojciechowski, Cisowski e Małek [72]	Route optimization for city cleaning vehicle	Apresentar uma solução para otimização de rota de um veículo de coleta de descarte	Os autores criam um algoritmo para solução do problema de roteamento, apesar de eficaz, existem algoritmos mais genéricos e eficientes sendo colocados em prática hoje em dia
Rahman e Nielsen [61]	Scheduling automated transport vehicles for material distribution systems	Propor uma nova metodologia para solução da distribuição de tarefas para dois AGVs portuários	Resultados muito promissores, abrindo as portas para estudos mais avançados, como a utilização desta metodologia em sistemas com um grande número de agentes a fim de ratifiar os resultados obtidos.
Deng et al. [19]	Task allocation algorithm and optimization model on edge collaboration	Investiga um ambiente de computação móvel de ponta para tarefas de análise de vídeo	Resultados promissores foram obtidos, evidenciando o desempenho superior do algoritmo proposto em comparação com os algoritmos utilizados. Mesmo em situações desfavoráveis, sua robustez e eficácia foram consolidadas. A continuidade dos estudos, aplicando o algoritmo em diversas situações e introduzindo novos comparativos, reforçará sua relevância no cenário atual.

Dunn et al. [22]	Deep Reinforcemente Learning for picker routing problem in warehousing	Introduzir um modelo de aprendizado por reforço capaz de modelar as rotas de AGVs em um armazém	Resultados muito promissores que, apesar de tratarem um outro aspecto do problemas retratado neste estudo, demonstram uma visão positiva para o futuro da área.
Perumaal Subramanian e Kumar Chandrasekar [58]	Simultaneous allocation and sequencing of orders for robotic mobile fulfillment system using reinforcement learning algorithm	Propor um sistema de alocação e sequenciamento de ordens simultânea por aprendizado por reforço.	Resultados interessantes, mas o algoritmo proposto sofre do mesmo problema que alguns dos algoritmos já em uso, com seu custo computacional aumentando exponencialmente para sistemas maiores e mais complexos.
Nahhas, Kharitonov e Turowski [48]	Deep Reinforcement Learning for Solving Allocation Problems in Supply Chain: An Image-Based Observation Space	Apresentar dois modelos de aprendizado profundo como possíveis soluções ao problema de alocação de tarefas em sistemas de armazenamento autônomos	Resultados bem interessantes, demonstrando certos níveis de eficiência dos modelos propostos como solução para o problema retratado.
Ferretti e Marchi [24]	Q-Learning for Inventory Management: an application case	Aplicar a técnica de aprendizado Q no controle de um armazém	Resultados muito promissores, com o algoritmo proposto obtendo um desempenho equiparavel a alguns dos algoritmos em uso atualmente.
Bello et al. [6]	Neural Combinatorial Optimization with Reinforcement Learning	Introduzir uma nova forma de solucionar problemas logísticos	Artigo muito interessante, servindo de inspiração para a forma final da solução estudada neste trabalho, atingindo resultados bastante impressionantes.

Fonte: Autor

Tabela 4: Artigos relacionados a este trabalho, seus objetivos e opinião do autor

4 METODOLOGIA

Neste capítulo, aprofunda-se a discussão do desenvolvimento da pesquisa, dos experimentos realizados e do simulador desenvolvido para a validação das soluções geradas, com o intuito de esclarecer a origem dos resultados obtidos, a metodologia adotada para alcançá-los e os desafios enfrentados ao longo desse processo. Em seguida, são apresentados e discutidos os resultados obtidos durante este trabalho.

Conforme explicitado anteriormente, para avaliar o desempenho da solução proposta, foram utilizados 2 algoritmos, sendo eles o *Earliest Due Date*(EDD) e o Algoritmo Genético(GA), aplicados conforme aquilo que foi definido a partir do proposto e apresentado por Rahman e Nielsen [61] no artigo discutido no capítulo anterior. Estes métodos, suas aplicações e os cenários onde são aplicados são brevemente discutidos a seguir.

Este estudo, por explorar um cenário de alta complexidade, foi dividido em duas partes, no primeiro momento foi explorada uma versão simples do cenário, a fim de gerar uma prova de conceito para a solução proposta pelo autor, assim como verificar sua eficácia quando comparada com soluções bastante utilizadas. Já no segundo momento, a solução proposta foi aplicada na versão completa do cenário, considerando e respeitando todas as restrições explicitadas anteriormente pelo autor, e então foram selecionadas 4 RNAs que obtiveram desempenhos distintos para serem aplicadas em diferentes cenários a fim de observar o impacto que mudanças nas características do sistema têm no desempenho das redes.

Em ambos os momentos da pesquisa, a solução foi desenvolvida na linguagem Python [68], devido à sua ampla utilização para a concepção de soluções que utilizam RNAs, havendo uma gama de informações e soluções para problemas comuns que podem ser encontradas prontamente através de simples buscas na internet. Outro fator que levou à escolha da linguagem é a diversidade em bibliotecas disponíveis para a elaboração das diversas arquiteturas de rede, sendo utilizada especificamente neste estudo a biblioteca PyTorch [53] por tratar-se da biblioteca com a qual o autor possui maior familiaridade e expertise na utilização.

4.1 Algoritmos utilizados

4.1.1 Algoritmo Earliest Due Date

O algoritmo EDD é um algoritmo heurístico simples, que utiliza os tempos limites para completar cada tarefa, que neste cenário são os tempos de encerramento das janelas de entrega, para definir a ordenação do itinerário das tarefas, ordenando-as de forma crescente, de acordo com a proximidade do encerramento da janela. Por exemplo, as tarefas 1-2-3-4, estão ativas simultaneamente, possuindo janelas de entrega que se encerram em 22, 34, 20 e 40 minutos, respectivamente. Utilizando o EDD a ordenação na fila do AGV passa a ser 3-1-2-4 caso a fila esteja vazia. Caso já existam tarefas na fila, o algoritmo adiciona as tarefas na fila e a reorganiza conforme a lógica citada.

4.1.2 Algoritmo Genético

Dado aumento em complexidade com a subdivisão das tarefas, a capacidade de realizar várias tarefas simultaneamente e o fato de o algoritmo receber uma única fila de tarefas composta pelas filas de cada agente, o autor optou por aumentar significativamente o tamanho da população e do número de gerações, comparado aos primeiros momentos desta pesquisa, em que foram utilizados 10 gerações e uma população com 100 membros. Na segunda etapa do estudo, esses números foram aumentados para 150 gerações com 500 membros cada.

O Algoritmo Genético (AG), um algoritmo famoso amplamente utilizado para resolver diversos problemas de computação [2]. Neste trabalho, o autor implementou o algoritmo utilizando a ordenação das tarefas como os cromossomos dos indivíduos da população, e para avaliar a performance da população, o *Fitness* implementado realiza a soma do tempo levado para realizar cada tarefa da fila com o tempo em ócio total e as penalidades acarretadas para cada tarefa que não foi concluída com sucesso, a seguir é ilustrada esta soma.

$$F_i = T_t + T_o + P_f$$

Onde,

- F_i é a fitness do indivíduo;
- T_t é o tempo total levado para concluir as tarefas;
- T_o é o tempo em ócio total;
- P_f é a penalidade por falhas.

Após a avaliação de todos os indivíduos da população, estes são classificados de acordo com seus valores de *fitness* em ordem decrescente, os melhores indivíduos são então selecionados para reprodução e darem início à nova geração. Durante o processo de

reprodução, é escolhido um ponto de corte aleatório nos cromossomos dos pais, a partir do qual ocorre a troca entre estes. Há também uma possibilidade de que ocorra uma mutação nos genes dos novos indivíduos. Essa chance foi definida com um valor de 1% pelo autor, e quando ocorre, 2 posições aleatórias são trocadas no cromossomo do indivíduo que sofreu a mutação.

4.1.3 Redes Neurais Artificiais

Conforme previamente citado, o autor desenvolveu as RNAs utilizando um conjunto de técnicas, especificamente LSTMs de aprendizado profundo utilizando a técnica de Q-Learning como método de aprendizado, como explicitado nas seções 2.2.3 e 2.2.4, onde essas técnicas são apresentadas e brevemente discutidas. O autor optou por utilizar este agrupamento de metodologias principalmente devido à sua familiaridade com seu desenvolvimento e aplicações. Foram desenvolvidas 3 arquiteturas diferentes, com uma utilizada apenas nos momentos iniciais deste estudo, a fim de estabelecer uma prova de conceito, e duas que seriam utilizadas nos experimentos finais. Com base naquilo que foi apresentado no capítulo 2, quando unido ao apresentado nesta seção, percebe-se que a proposta apresentada trata-se de uma heurística híbrida, sendo a união de uma heurística, neste estudo o EDD, com outras técnicas, as RNAs apresentadas. A figura 13 apresenta a classificação dos algoritmos utilizados nas categorias apresentadas no referencial teórico.

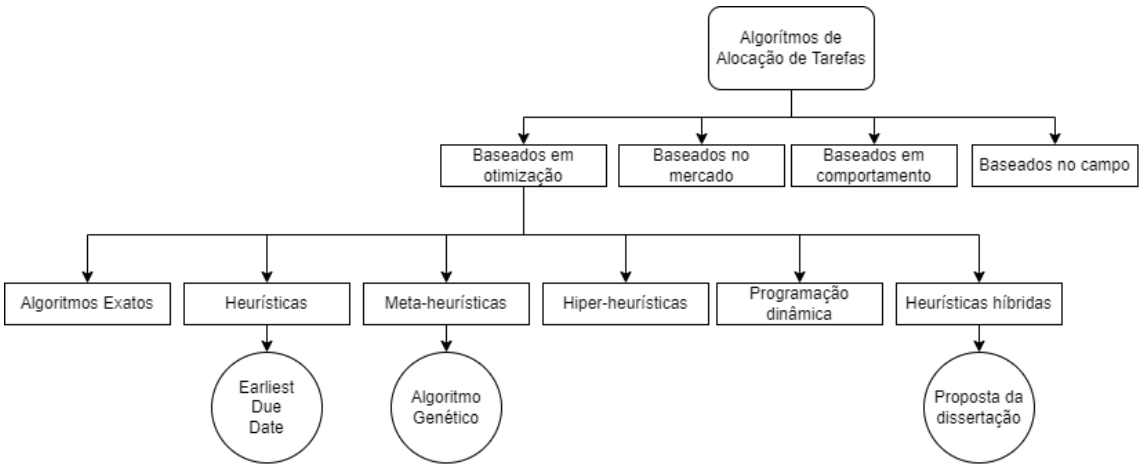


Figura 13: Categorização dos algoritmos utilizados

Fonte: Autor (2025)

De acordo com aquilo dito no início deste capítulo, as redes foram desenvolvidas na linguagem Python [68], utilizando a biblioteca Pytorch [53] para a construção das redes. Diferente do treinamento epocal, onde uma RNA é treinada por n épocas, com cada época representando uma exploração total do *dataset* de treino, neste estudo o autor treinou as redes não por épocas, mas sim por tempo total de treino, pois o autor encontrou um grande obstáculo no momento da definição das épocas, dada a natureza do problema, como definir estas de uma forma que garanta um aprendizado homogêneo durante os trei-

nos. Se definida uma passagem completa pela fila de tarefas como uma época, é necessário estabelecer também um conjunto de tarefas que serão adicionadas conforme acontece no ambiente real, e a exploração necessária para encontrar um número que garanta um aprendizado satisfatório, que explore uma população realmente representativa das tarefas existentes, acabaria por consumir uma parcela muito grande de tempo, a qual o autor não possuía disponível dado limite de tempo para a conclusão dos estudos desta dissertação.

Caso as épocas fossem definidas como a exploração completa de todas as possíveis tarefas, como as tarefas são adicionadas de forma aleatória ao final da fila, juntamente com a característica de que certas tarefas são ativadas com uma frequência expressivamente maior que outras, é completamente possível que uma rede nunca consiga encerrar uma época de treino, ou então, a RNA passe apenas uma vez por cada tarefa, aprendendo um cenário que não representa a realidade. Dessa forma, com o intuito de escapar desses problemas, o autor optou por utilizar tempo real de treino, permitindo dessa forma que o sistema escape destas restrições, garantindo que todas as RNAs serão capazes de explorar o ambiente de tarefas de uma maneira mais próxima daquilo que ocorre em ambientes reais.

Com o intuito de verificar a viabilidade dos estudos, a RNA criada no início destes foi bastante simples, sendo uma única RNA, responsável pela reorganização da fila de tarefas oriunda do EDD. Estes testes foram realizados no cenário inicial, apresentado na seção 4.2.1 , buscando determinar os parâmetros da rede com os quais ela alcança o melhor desempenho, a partir dos quais o autor desenvolveu as redes subsequentes, dessa forma agilizando o processo de desenvolvimento final. Durante esta parte dos estudos, o autor, a partir dos testes realizados, determinou que o melhor desempenho ocorria com uma taxa de aprendizado de $0,001$, isto é $0,1\%$, um gamma de $0,9$ e uma capacidade de 100.000 estados na memória da LSTM.

Durante as fases iniciais da segunda etapa da pesquisa, a rede desenvolvida, com a arquitetura que obteve os melhores resultados no cenário inicial, foi inserida no cenário real, obtendo resultados bastante negativos que são apresentados na seção 4.3. Com isso, o autor observou, então, que a arquitetura anterior não possuía capacidade de lidar com o aumento exponencial de complexidade do problema, havendo a necessidade de explorar as soluções propostas por outros autores. Durante esta nova fase de pesquisa, foi encontrado o trabalho de Bello et al. [6] o qual explora um cenário próximo ao estudado nesta dissertação.

Conforme explorado nas seções anteriores, em seu trabalho, Bello et al. [6] utiliza um par de RNAs como uma possível ferramenta para encontrar uma solução, o que inspirou o autor a realizar a mudança da solução proposta de uma única RNA para um par destas, agindo de forma cooperativa, onde uma das redes é responsável por dividir a fila entre os agentes e a outra é responsável por organizar a ordenação das subtarefas de cada agente.

Também foi desenvolvida uma 2ª arquitetura para a rede mestra, que ao invés de gerar

diferentes vieses para cada tarefa, ela diretamente atribuiria cada tarefa a um agente, mas infelizmente, devido a problemas na implementação desta no código que já havia sido desenvolvido, necessitando uma reescrita completa do código para solucionar os problemas, juntamente com o fato de que, na visão do autor, uma atribuição direta das tarefas vai de encontro com o proposto do estudo, pois a rede mestra passaria a assumir um controle maior sobre a fila que o EDD, dessa forma deixando de ser uma ferramenta para suporte e tornando-se uma ferramenta para encontrar a solução diretamente, assim, esta arquitetura foi descartada pelo autor.

Durante a realização dos experimentos, a rede mestra é alimentada o vetor gerado a partir da solução obtida pelo EDD, ressalta-se que este vetor é composto pela lista de tarefas de todos os agentes, gerando como saída uma lista de vieses, juntamente com esses vieses também é alocada uma pontuação para cada tarefa, conforme a posição na fila de tarefas. Com a pontuação e a lista de vieses gerados, seus valores são somados e a lista de tarefas é reorganizada a partir desta nova pontuação, concluindo assim a alocação das tarefas para seus respectivos agentes. Abaixo são listadas as informações inseridas nos vetores alimentados.

- Id do nodo atual do agente;
- Id do nodo objetivo da tarefa;
- Tempo levado para se deslocar até o objetivo normalizado;
- Tempo para realizar a tarefa normalizado;
- Início da janela da tarefa normalizado;
- Encerramento da janela da tarefa normalizado;
- Tipo de tarefa (Carga ou Descarga), esta informação é passada apenas para a rede agente;
- Tipagem ABC da tarefa;
- Viés atual da tarefa.

A nova fila é então dividida de acordo com o número de agentes, e subsequentemente as filas dos agentes têm suas tarefas separadas em carga e descarga, e então é gerado um vetor de entrada a partir da nova sequência das tarefas. Neste caso, é criado um vetor para cada agente, que é então alimentado para a rede Agente, que, assim como a rede mestra, retorna um vetor de vieses. Com esses novos vieses, repete-se o processo da etapa anterior, somando a pontuação de cada subtarefa com seu viés, e reorganizando a fila conforme as pontuações finais.

Com a fila reorganizada, é passada pela função avaliadora, que conforme previamente citado é bastante simples, utilizando comparações para estabelecer os resultados a partir da fila alimentada e uma variável que faz o controle do tempo passado, simulando a execução das tarefas, e conforme o desempenho da proposta de solução, retorna uma pontuação que é enviada para ambas as RNAs para que elas possam realizar seus reajustes. Após o rebalanceamento das redes, este processo de treinamento é repetido até que as durações estabelecidas para os testes sejam alcançadas. As figuras 14 e 15 trazem fluxogramas para ajudar no entendimento. O Anexo A apresenta um modelo matemático da função avaliadora e o Anexo B traz uma breve simulação utilizando o modelo apresentado.

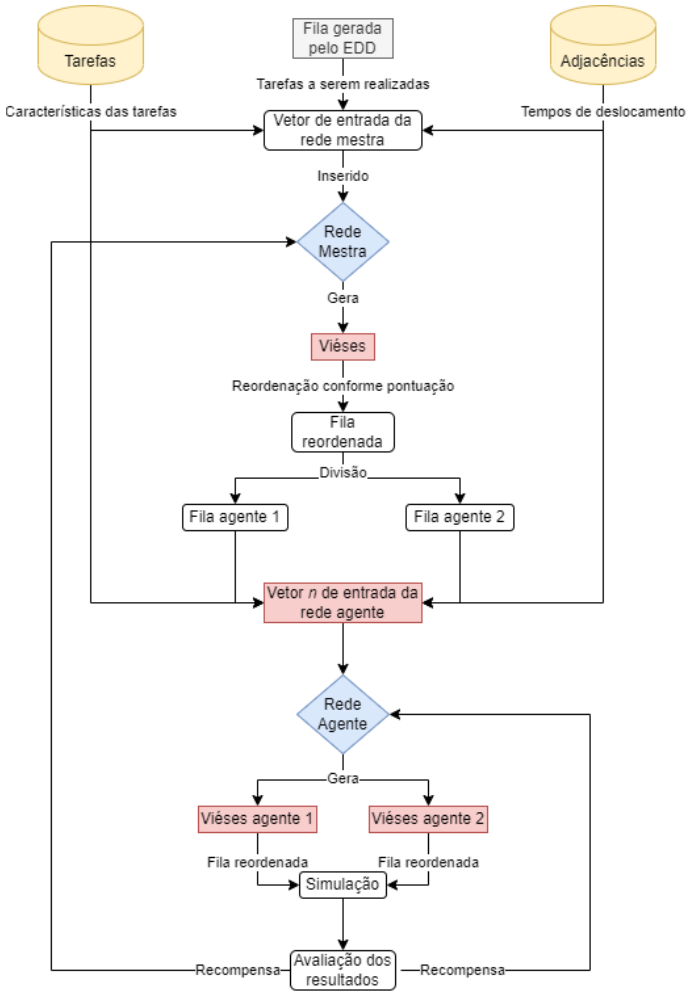


Figura 14: Fluxograma das RNAs desenvolvidas

Fonte: Autor (2025)

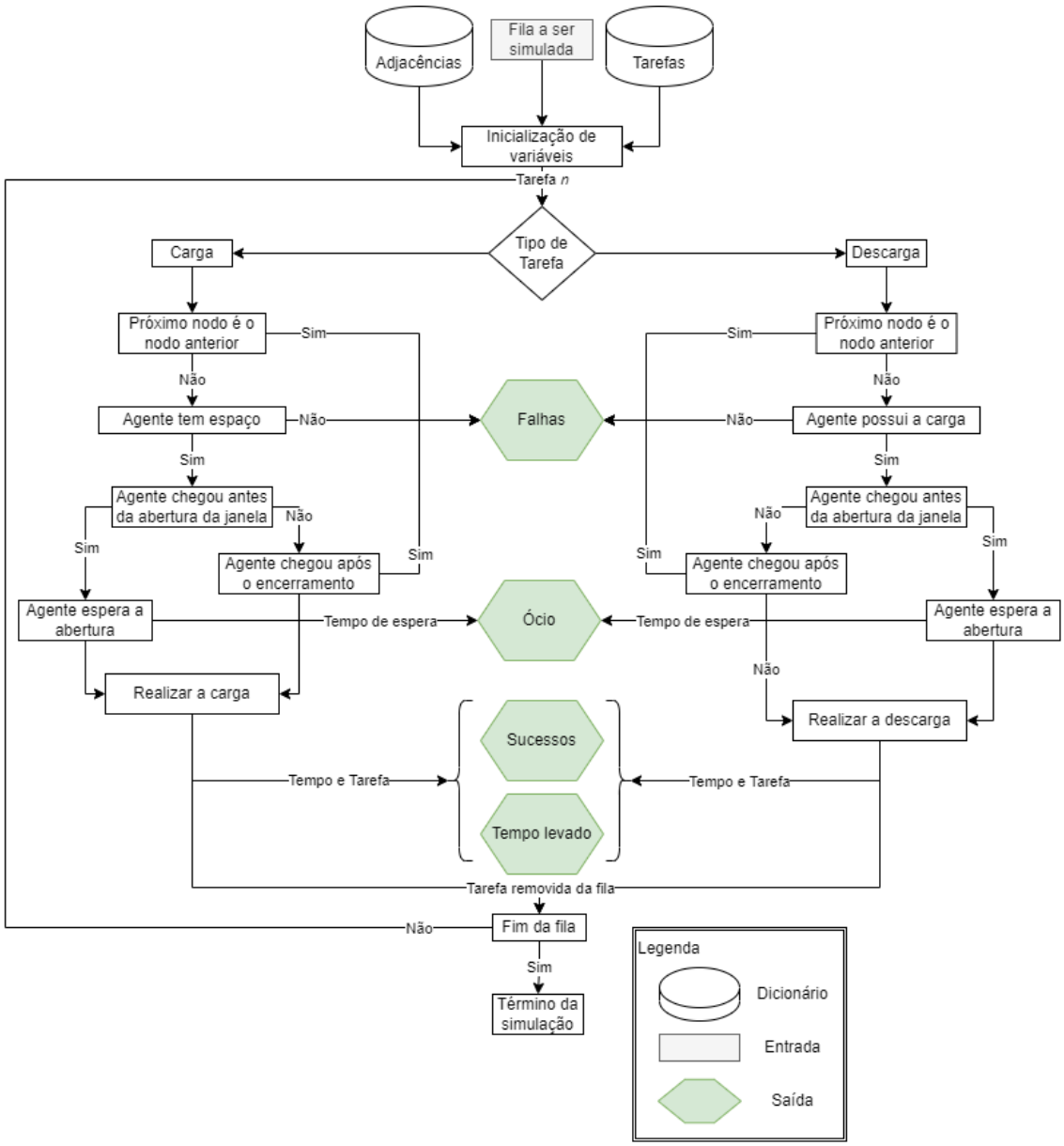


Figura 15: Fluxograma da função avaliadora

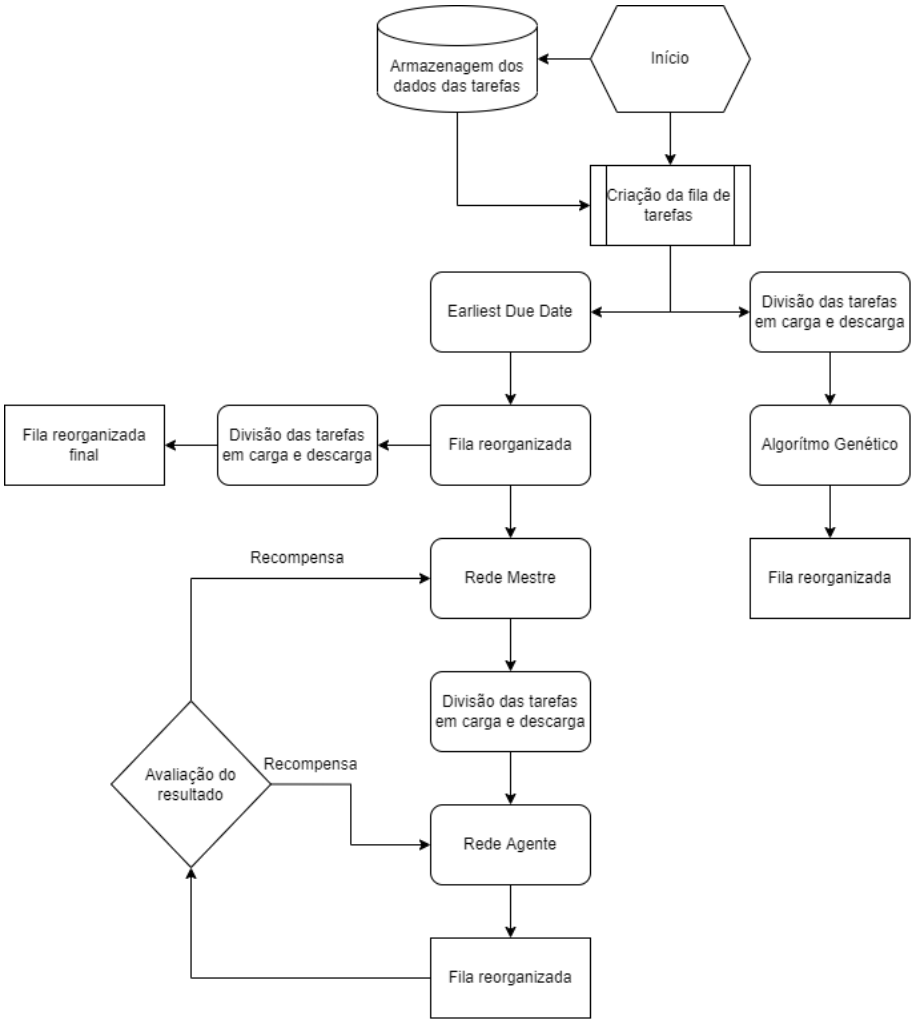
Fonte: Autor (2025)

4.2 Métodos experimentais

Para a realização dos experimentos, o autor seguiu 5 fases distintas, sendo elas a criação dos dados, a solução do problema via EDD, a solução do problema via AG, o treinamento das redes e a execução dos testes das redes treinadas. A Figura 16 mostra um fluxograma da lógica do sistema.

Durante sua primeira fase, ocorre a geração dos dados a partir dos valores definidos para os experimentos. Durante a fase exploratória, o autor optou por utilizar o 3º cenário experimental, apresentado na seção 4.2.1 a seguir, pois é o cenário representativo do problema real explorado nesta dissertação. O autor também definiu os valores de 8 pontos

Figura 16: Fluxograma do simulador desenvolvido



Fonte: Autor (2025)

de coleta e 10 pontos de entrega, que são utilizados para definir o número de tarefas que serão criadas, estas têm os valores de cada componente definidos de forma aleatória. A Tabela 5 lista essas características, bem como os valores mínimos e máximos possíveis para cada uma.

O mapa dos nodos também é gerado neste momento, sendo que ele conta com um nodo adicional, chamado de nó neutro, onde todos os agentes se encontram no momento em que os experimentos são iniciados, este nodo neutro conecta-se somente aos pontos de coleta, já o restante dos nodos são interconectados, com o tempo para viagem entre eles também sendo gerado aleatoriamente, dentro de um intervalo de 1 a 5 minutos. Destaca-se que os tempos de fechamento de janelas são somados aos de abertura, ou seja, se uma tarefa tem uma janela de abertura com duração de 3 minutos e uma janela de fechamento de 8 minutos, o tempo real de encerramento da janela é de 11 minutos a partir do momento em que a tarefa foi aberta. Essa abordagem visa representar de maneira fiel os desafios

temporais envolvidos na execução das tarefas pelo AGV.

Componente da tarefa	Intervalo de Aleatorização (minutos)	Justificativa
Abertura Janela de carga	0 a 10	Estes valores foram estabelecidos sem uma justificativa específica em mente, apenas que este intervalo permitiria uma maior distribuição de tempos na criação do dicionário de tarefas.
Encerramento Janela de carga	5 a 10	Este intervalo foi estabelecido levando em mente os tempos máximos de locomoção e carga
Tempo de carga	1 a 5	Valores criados a fim de adicionar uma certa variedade nas tarefas, a fim de representar os diferentes tipos de cargas possíveis
Abertura Janela de Descarga	5 a 10	Estabelecido levando em mente os tempos máximos de locomoção
Encerramento Janela de Descarga	5 a 10	Estabelecido levando em mente os tempos máximos de locomoção e carga
Tempo de Descarga	1 a 5	Valores criados a fim de adicionar uma certa variedade nas tarefas, a fim de representar os diferentes tipos de cargas possíveis

Tabela 5: Valores e Justificativas dos intervalos de tempos das tarefas

Após a geração dos dados, o sistema gera uma fila de tarefas, obedecendo a uma simples regra, a duração total da fila, considerando que as tarefas sejam realizadas de forma consecutiva, não deve ultrapassar 4 horas por agente, garantindo assim que existe uma solução em que os agentes conseguem realizar 100% das tarefas. Para facilitar a adaptação dos algoritmos, o sistema gera uma única fila, composta pelas filas de cada agente concatenadas. A fim de introduzir variabilidade e realismo aos experimentos, o simulador gera um vetor gatilho, composto por valores aleatórios de 0 a 1, que determina o momento em as tarefas terão suas janelas de tempo iniciadas durante os experimentos. Com a fila e gatilho gerados, o sistema aplica seu primeiro algoritmo, o Earliest Due Date (EDD).

Com os experimentos do EDD finalizados, o sistema passa para sua terceira fase, onde é utilizado o algoritmo genético para encontrar uma solução para o problema. Durante essa fase ocorre pela primeira vez a separação das tarefas em subtarefas de carga e descarga, com essa nova fila sendo então alimentada ao algoritmo.

Com a finalização do Algoritmo Genético, é dado início à fase de treino das redes neurais, o autor seguiu o exemplo de Bello et al. [6], conforme previamente citado. Após os treinos da redes é iniciada a última etapa do sistema onde ocorrem os testes finais,

com as redes sendo alimentadas com a fila inicial e seu desempenho avaliado pela função avaliativa, retornando a quantidade de tarefas completas, falhas, os tempos levados para completar cada tarefa e o tempo que o agente ficou em ócio. Os dados retornados pela função são então salvos, e os dados de todos os algoritmos utilizados são processados e salvos em um dicionário. Após esse processo, os experimentos são repetidos 100 vezes, este número foi estabelecido pelo autor a fim de gerar uma população suficiente para o comportamento das redes ser avaliado. Após isso os dados processados são exportados para diferentes arquivos que contêm os valores obtidos.

4.2.1 Cenários experimentais

Dada a complexidade do problema, o autor estabeleceu 4 cenários distintos para a avaliação do desempenho das RNAs desenvolvidas e um cenário inicial para o estabelecimento dos parâmetros iniciais. Estes 4 cenários vão de algo bastante simples até o cenário completo com todas as restrições previamente discutidas. A seguir, estes cenários são brevemente apresentados.

O Cenário inicial, utilizado durante os primeiros momentos da pesquisa, possuía a finalidade de estabelecer uma "prova de conceito". Neste cenário, o armazém conta com apenas uma unidade autônoma capaz de realizar apenas uma tarefa por vez. Este cenário foi utilizado a fim de encontrar os parâmetros iniciais nos quais a RNA utilizada seria capaz de alcançar resultados considerados satisfatórios, isto é, superando os resultados dos algoritmos simples que são utilizados para encontrar uma solução ao problema.

O primeiro cenário experimental, tem apenas duas diferenças quando comparado ao inicial, estas diferenças se encontram no aumento da capacidade de carga do agente para 5 cargas diferentes, permitindo que este faça até 5 tarefas de forma concomitante. A outra diferença em relação ao primeiro cenário é a inserção da restrição de movimento, conforme citado anteriormente, quando o agente passa a ser um comboio com 4 vagões, este perde a capacidade de realizar movimentos em marcha ré, caso este movimento seja realizado, os vagões passam a possuir um alto risco de descarrilhamento. Já no segundo cenário experimental, passam a ser utilizados 2 agentes, mas com um número menor de vagões, apenas 2 vagões. Assim, os agentes contam com uma capacidade de realizar 3 tarefas simultaneamente.

O Terceiro cenário experimental representa o problema real explorado neste estudo, contando com 2 AGVs, cada um com 4 vagões de carga, capazes de realizar até 5 tarefas ao mesmo tempo e com a movimentação em marcha ré sendo penalizada. Já o quarto cenário é quase idêntico ao anterior, a diferença sendo a exclusão da restrição de movimento. Estes cenários foram criados com o intuito de estudar o comportamento de algumas redes selecionadas, a partir da mudança de fatores externos a ela, e o quanto estes impactam nos resultados alcançados ao final dos treinos. A tabela 6 traz uma comparação destas distinções.

Cenário	Nº de AGVs	Restrição de movimento	Capacidade de carga
0	1	Não	1
1	1	Sim	5
2	2	Sim	3
3	2	Sim	5
4	2	Não	5

Tabela 6: Os diferentes cenários experimentais

4.3 Experimentos e resultados obtidos

Durante o primeiro momento da pesquisa, a solução proposta, aplicada ao cenário inicial, gerou resultados bastante positivos, obtendo uma média de $14,8$ min para completar cada tarefa, uma média de tempo em ócio de $8,792$ min e um percentual de tarefas concluídas de $66,7\%$, quando comparados com os resultados obtidos pelos outros dois algoritmos de $14,1$ min para completar cada tarefa para o algoritmo EDD e 14 minutos para o algoritmo genético, $7,82$ minutos em ócio para o EDD e $7,79$ minutos em ócio para o AG, com o EDD obtendo 63% de tarefas completas e o AG obtendo 64%.

Com os resultados da primeira fase dos estudos, observou-se uma possibilidade positiva para dar continuidade aos estudos com o cenário mais realista, onde a solução foi aplicada com todas as devidas restrições. Conforme previamente discutido, neste segundo momento, foi explorada a solução num cenário com dois agentes capazes de carregar até cinco materiais diferentes, permitindo então que o agente realize até 5 tarefas de forma concomitante, obedecendo à restrição de movimentação em que o agente não é capaz de andar de marcha ré, havendo uma perda grande de tempo caso a solução envolva retornar ao nodo anterior ao atual e a geração da fila de tarefas também foi ajustada a fim de obedecer à regra de pareto, também conhecida como regra 80-20, que diz que um número pequeno de itens (produtos, tarefas, compras, etc.) são muito mais utilizados que o restante [65], assim um número pequeno de tarefas representa a grande maioria das tarefas geradas durante a duração da simulação.

Assim como no momento inicial desta pesquisa, foram realizados diversos experimentos e simulações com diferentes tamanhos para ambas as redes, as tabelas 7 a 9 apresentam os resultados e a arquitetura das redes selecionadas para aplicação nos cenários experimentais. Durante esta exploração, o autor experimentou com aproximadamente 30 variações das RNAs desenvolvidas, com 4 delas sendo escolhidas para serem aplicadas no cenário experimental, com as redes com 13, 10 e 15 camadas em ambas as redes obtendo os melhores resultados e a arquitetura com 9 camadas para a rede mestra e 16 para a rede agente sendo escolhida pois é representativa do comportamento do restante da população.

Nº de camadas		T_{tar} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	$9,13$	$9,00$	$10,71$	$7,58$	$0,43$	$0,66$
10	10	$8,72$	$8,60$	$10,90$	$7,25$	$0,42$	$0,66$
13	13	$6,92$	$6,80$	$8,23$	$5,13$	$0,43$	$0,66$
15	15	$6,87$	$6,79$	$8,67$	$5,08$	$0,43$	$0,66$

Tabela 7: Tempo para completar uma tarefa das redes selecionadas para os experimentos

Nº de camadas		T_{ocio} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	$180,94$	$147,38$	$502,78$	$77,81$	$8045,94$	$89,70$
10	10	$150,66$	$141,23$	$377,14$	$71,00$	$3327,39$	$57,68$
13	13	$60,26$	$58,52$	$134,43$	$38,94$	$143,16$	$11,97$
15	15	$61,90$	$59,95$	$109,18$	$42,40$	$151,78$	$12,32$

Tabela 8: Tempo em ócio das redes selecionadas para os experimentos

Nº de camadas		$P_{con}\%$					
Mestra	Agente	$\overline{\%}$	M	max	min	σ^2	σ
9	16	8	7	15	1	0,06	2
10	10	9	9	16	0	0,10	3
13	13	10	10	17	3	0,12	4
15	15	10	10	17	3	0,09	3

Tabela 9: Percentuais de tarefas concluídas das redes selecionadas para os experimentos

Onde,

$\overline{T_{tar}}$ é o valor médio dos tempos;

$\overline{\%}$ é o valor médio dos percentuais;

M é a mediana da população;

σ^2 é a variância da população;

σ é o desvio padrão da população.

Devido ao fato deste trabalho ter por objetivo ser inserido em meio ao ambiente empresarial, o autor elencou como fatores determinantes para a avaliação dos resultados seus valores, os limites superior e inferior e sua homogeneidade, representada nas tabelas pela variância e desvio padrão. A importância dos valores é clara, como o objetivo dos algoritmos estudados é otimizar a distribuição de tarefas, quanto menos tempo perdido e mais tarefas forem concluídas com sucesso, melhor o desempenho alcançado. Os limites superiores e inferiores têm sua importância na existência de outliers, representando a existência de incertezas nos resultados obtidos. A homogeneidade possui uma grande im-

portância quando analisada através de um viés econômico, representando uma segurança quanto às possíveis variações nos resultados, o que aumenta a confiabilidade da solução.

Quando analisado do ponto de vista de uma empresa, percebe-se que a homogeneidade possui um impacto bastante significativo. Um exemplo desta importância se torna aparente quando colocamos o sistema no âmbito de uma empresa varejista, onde pedidos de clientes vêm a todos os momentos e estes clientes esperam receber seus produtos no menor tempo possível, com qualquer atraso sendo algo negativo. A partir disso, o tempo que um agente leva para realizar a busca e entrega do pedido do cliente é extremamente importante para o cálculo do período de entrega para o requisitante. Dessa forma, os agentes possuírem uma baixa variação no tempo levado para realizar a manipulação do estoque torna-se um fator importante para o sistema.

Outro fator que demonstra a importância da homogeneidade é em caso em que os agentes forem responsáveis por realizar tarefas de manutenção, vistoria, instalações, entre outras tarefas. Neste cenário, tanto o agente chegando antes do horário previsto para o início dessas atividades, tendo de ficar em ócio até que elas se iniciem, quanto chegando após o horário previsto, gerando atrasos nas atividades subsequentes, o que causa impactos significativos nas atividades realizadas.

Conforme pode-se observar tanto nas tabelas 7 a 9 quanto na figura 17, o conjunto de redes com 13 camadas nas redes mestra e agente obteve o melhor resultado de um percentual de tarefas concluídas de 10%, um tempo médio para completar tarefas de $6,92$ minutos e um tempo em ócio médio de $60,26$ minutos. Os resultados obtidos pelo EDD foram de um percentual médio de $85,5\%$ das tarefas concluídas com sucesso, tempo médio para concluir uma tarefa de $9,43$ minutos e um tempo em ócio médio de $169,52$ minutos. O GA alcançou valores médios de 33% de tarefas concluídas com sucesso, $123,96$ minutos em ócio e $8,9$ minutos para completar uma tarefa.

Durante a realização dos experimentos, o autor também se atentou aos tempos de processamento de cada um dos algoritmos utilizados, bem como do sistema como um todo. O algoritmo genético, em média, necessitou de aproximadamente $10,5$ segundos para realizar suas iterações e encontrar sua melhor proposta de solução. O algoritmo EDD, por ser uma heurística direta, possui um baixo custo computacional, e com isso, tempos de processamento bastante baixos, alcançando uma média de $1,3 * 10^{-4}$ segundos para realizar a reordenação da fila de tarefas. Entretanto, as RNAs possuem uma maior variação nos tempos de processamento, sendo que, conforme o treino progride, o tempo de processamento necessário para realizar uma iteração aumenta, obtendo uma média de $0,28$ segundos por iteração e uma média de 94340 iterações de treino durante as 2h estabelecidas. Já o sistema como um todo obteve um tempo médio de execução, desconsiderando o tempo de treinamento, de $14,53$ segundos, com sua maior parte sendo referente ao processamento do AG.

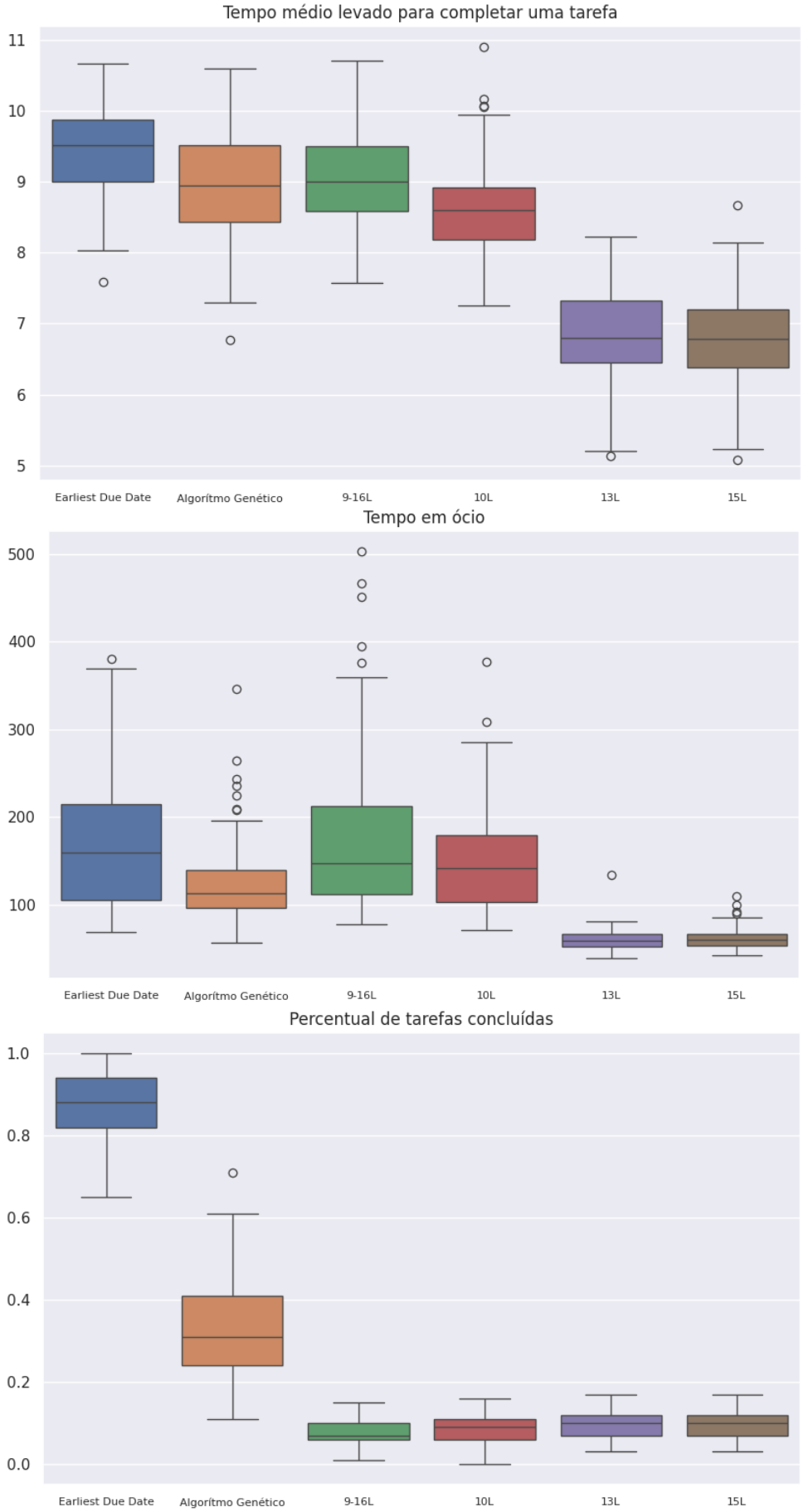


Figura 17: Visualização dos resultados das redes selecionadas

Fonte: Autor (2025)

4.3.1 Resultados obtidos nos cenários experimentais

Após os resultados obtidos durante a fase exploratória, o autor procurou explorar diferentes cenários a fim de, conforme dito na seção 4.2.1, analisar o impacto que as diferentes características do problema possuem nos resultados finais. A fim de controlar a aleatoriedade oriunda da geração das tarefas e do mapa de adjacências, os experimentos utilizaram valores idênticos em todos os cenários. Os resultados obtidos são apresentados numericamente nas tabelas 10 a 21, e gráficos de caixa, apresentados nas figuras 18 a 21, são utilizados com o intuito de facilitar a compreensão destes através de uma representação visual dos parâmetros analisados. Ressalta-se que, diferente do momento exploratório, os cenários experimentais utilizaram o mesmo dicionário de tarefa e a mesma matriz de adjacência. Assim, os resultados obtidos no cenário 3 são diferentes daqueles obtidos durante a exploração dos parâmetros internos das redes (Nº de camadas e tamanho de camadas).

Nº de camadas		T_{tar} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	8,16	8,12	10,20	6,20	0,86	0,93
10	10	8,50	8,42	10,36	6,22	0,72	0,85
13	13	8,42	8,35	11,12	6,14	0,87	0,93
15	15	8,77	8,62	11,14	6,23	0,82	0,90

Tabela 10: Tempo para completar uma tarefa no cenário 1

Nº de camadas		T_{ocio} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	115,33	74,92	400,25	27,50	7006,51	83,70
10	10	65,13	60,04	163,29	37,00	350,93	18,73
13	13	86,76	63,89	225,17	28,55	2702,42	51,98
15	15	79,16	61,86	213,14	37,40	1540,71	39,25

Tabela 11: Tempo em ócio das redes no cenário 1

Nº de camadas		$P_{con}\%$					
Mestra	Agente	$\overline{\%}$	M	max	min	σ^2	σ
9	16	12	12	31	0	0,63	8
10	10	14	12	38	0	0,61	8
13	13	15	14	36	0	0,76	9
15	15	17	19	38	0	0,84	9

Tabela 12: Percentuais de tarefas completas no cenário 1

Nº de camadas		T_{tar} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	8,75	8,61	10,64	6,92	0,35	0,60
10	10	8,89	8,81	10,88	7,62	0,33	0,58
13	13	7,86	7,67	10,90	5,71	0,91	0,95
15	15	8,58	8,43	10,38	6,27	0,51	0,72

Tabela 13: Tempo para completar uma tarefa no cenário 2

Nº de camadas		T_{ocio} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	123,46	115,97	293,06	58,36	2458,94	49,59
10	10	128,01	92,02	319,06	49,82	5268,95	72,59
13	13	287,33	280,43	725,57	46,53	23051,96	151,83
15	15	150,43	137,02	342,00	52,81	5096,75	71,39

Tabela 14: Tempo em ócio das redes no cenário 2

Nº de camadas		$P_{con}\%$					
Mestra	Agente	$\overline{\%}$	M	max	min	σ^2	σ
9	16	12	11	25	0	0,29	5
10	10	17	18	36	4	0,38	6
13	13	13	12	25	3	0,20	4
15	15	12	12	25	0	0,25	5

Tabela 15: Percentuais de tarefas completas no cenário 2

Nº de camadas		T_{tar} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	8,95	8,87	10,41	7,65	0,33	0,57
10	10	8,77	8,58	10,31	7,37	0,35	0,59
13	13	7,57	7,50	8,93	6,22	0,25	0,50
15	15	8,31	8,23	9,80	6,50	0,43	0,65

Tabela 16: Tempo para completar uma tarefa no cenário 3

Nº de camadas		T_{ocio} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	185,27	168,13	392,80	64,17	7556,11	86,93
10	10	97,73	90,34	220,25	64,85	685,79	26,19
13	13	127,20	100,70	366,65	55,53	4547,51	67,44
15	15	188,69	169,89	471,17	65,18	7899,72	88,88

Tabela 17: Tempo em ócio das redes no cenário 3

Nº de camadas		$P_{con}\%$					
Mestra	Agente	$\overline{\%}$	M	max	min	σ^2	σ
9	16	17	16	38	6	0,42	7
10	10	17	16	34	6	0,37	6
13	13	13	12	28	3	0,37	6
15	15	14	12	28	3	0,32	6

Tabela 18: Percentuais de tarefas completas no cenário 3

Nº de camadas		T_{tar} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	8,95	8,87	10,41	7,65	0,33	0,57
10	10	8,74	8,63	10,06	7,17	0,27	0,52
13	13	7,57	7,50	8,93	6,22	0,25	0,50
15	15	8,49	8,40	10,14	7,00	0,47	0,68

Tabela 19: Tempo para completar uma tarefa no cenário 4

Nº de camadas		T_{ocio} minutos					
Mestra	Agente	$\overline{T}$	M	max	min	σ^2	σ
9	16	185,27	168,13	392,80	64,17	7556,11	86,93
10	10	98,31	87,58	217,44	66,95	771,52	27,78
13	13	127,20	100,70	366,65	55,53	4547,51	67,44
15	15	177,54	147,76	438,54	62,57	8206,50	90,59

Tabela 20: Tempo em ócio das redes no cenário 4

Nº de camadas		$P_{con}\%$					
Mestra	Agente	$\overline{\%}$	M	max	min	σ^2	σ
9	16	18	19	34	0	0,40	6
10	10	18	19	38	6	0,36	6
13	13	14	12	34	3	0,43	7
15	15	18	18	38	3	0,40	6

Tabela 21: Percentuais de tarefas completas no cenário 4

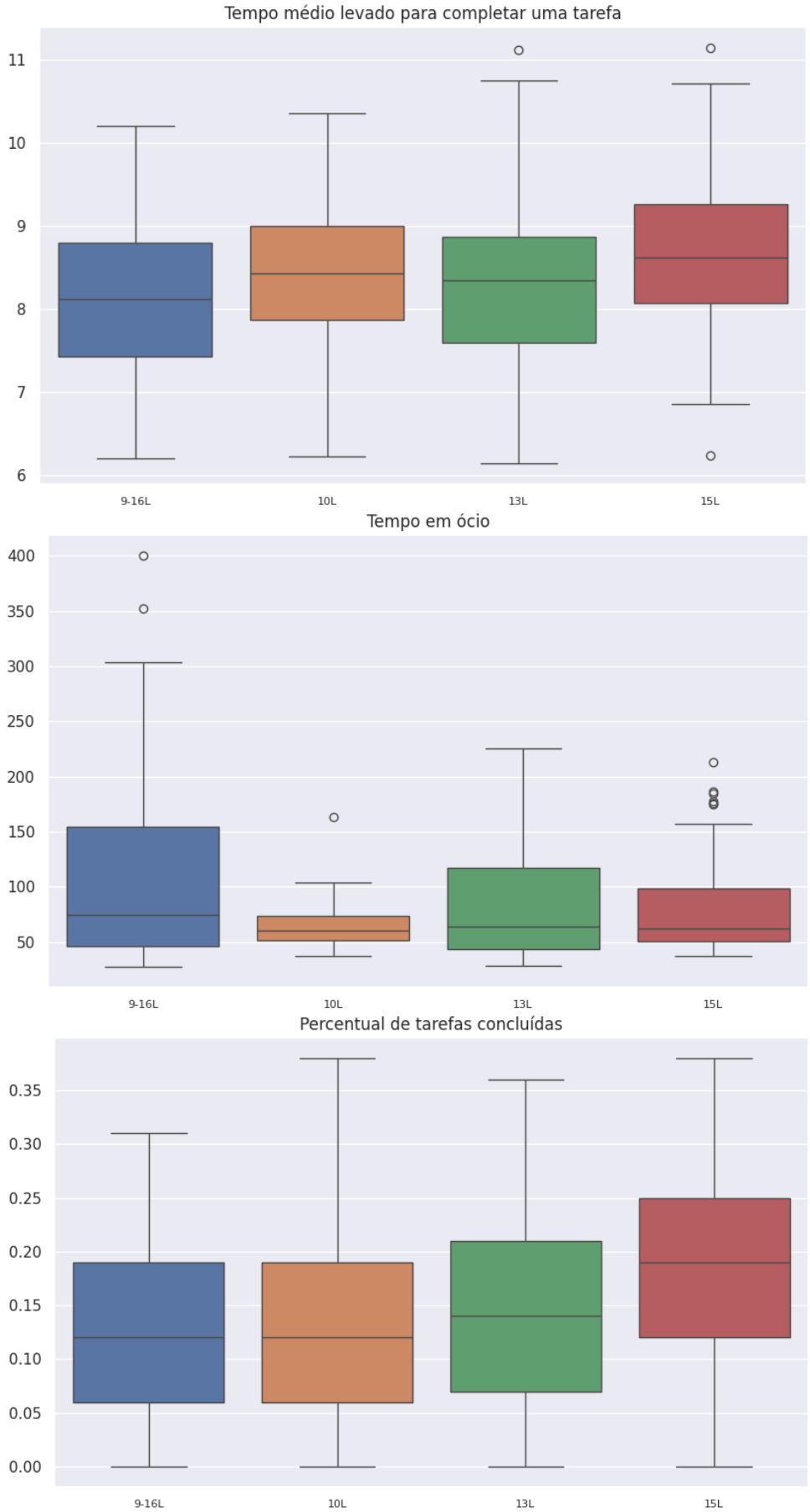


Figura 18: Visualização dos resultados obtidos no cenário 1

Fonte: Autor (2025)

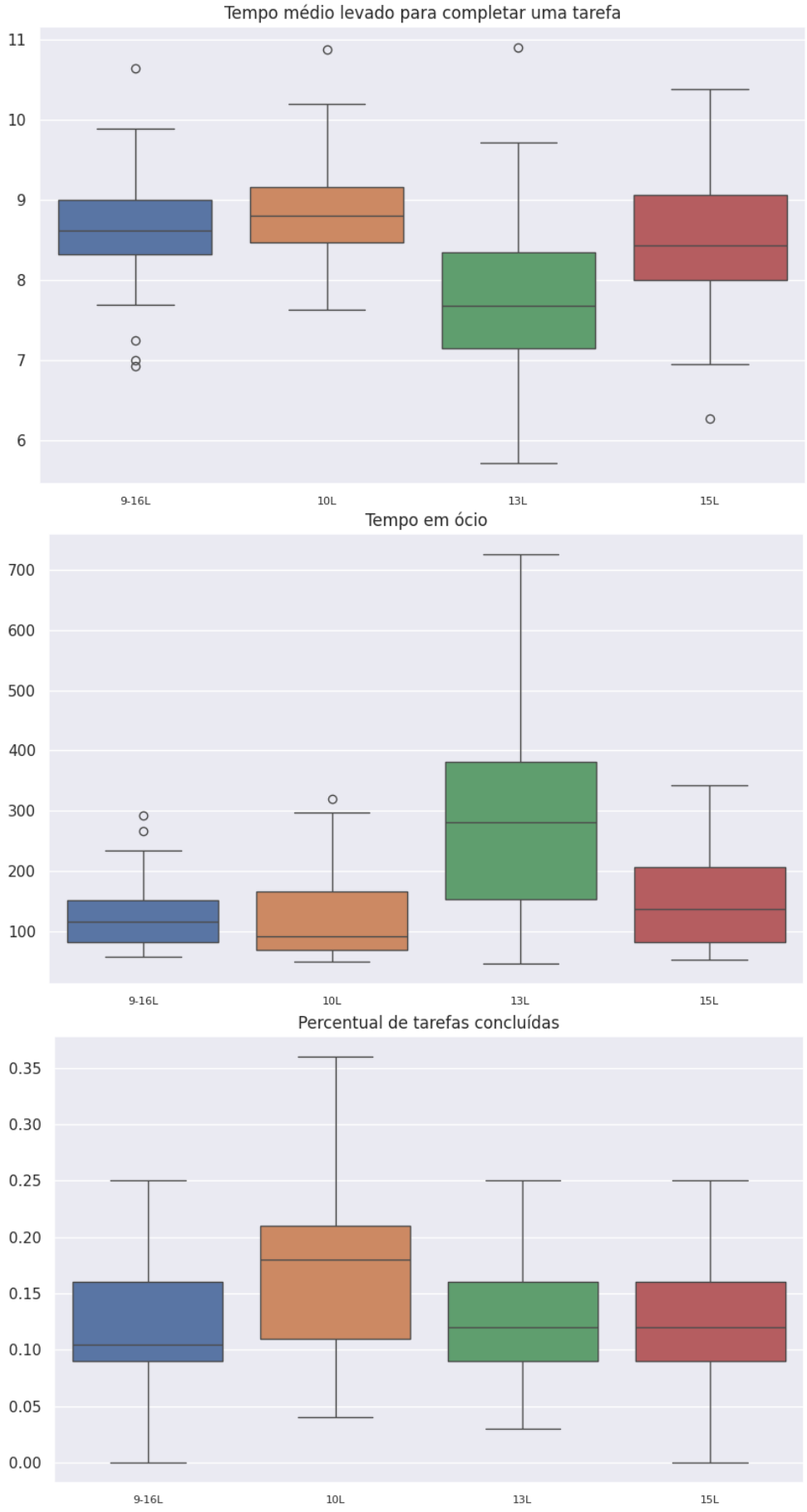


Figura 19: Visualização dos resultados obtidos no cenário 2

Fonte: Autor (2025)

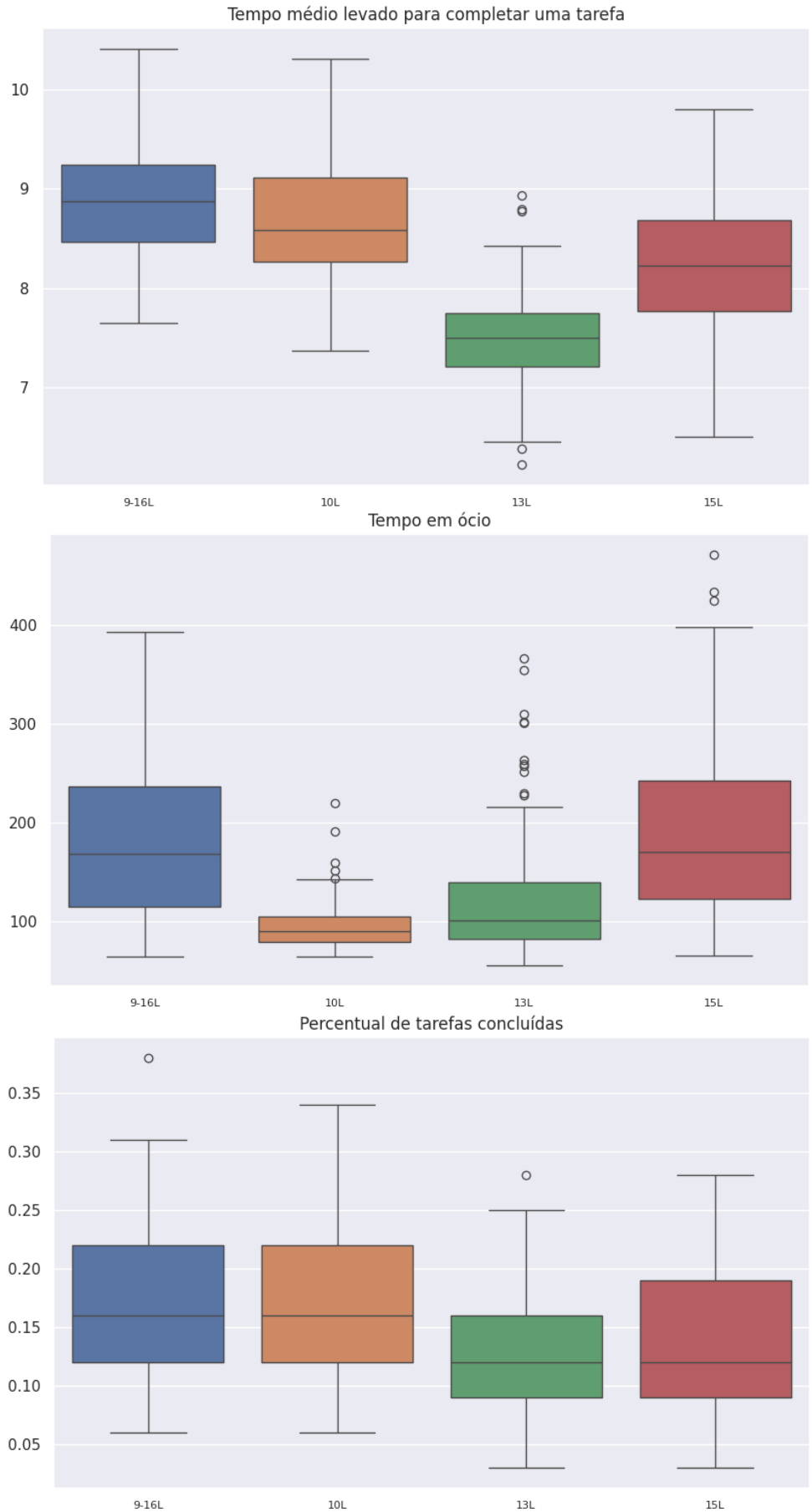


Figura 20: Visualização dos resultados obtidos no cenário 3

Fonte: Autor (2025)

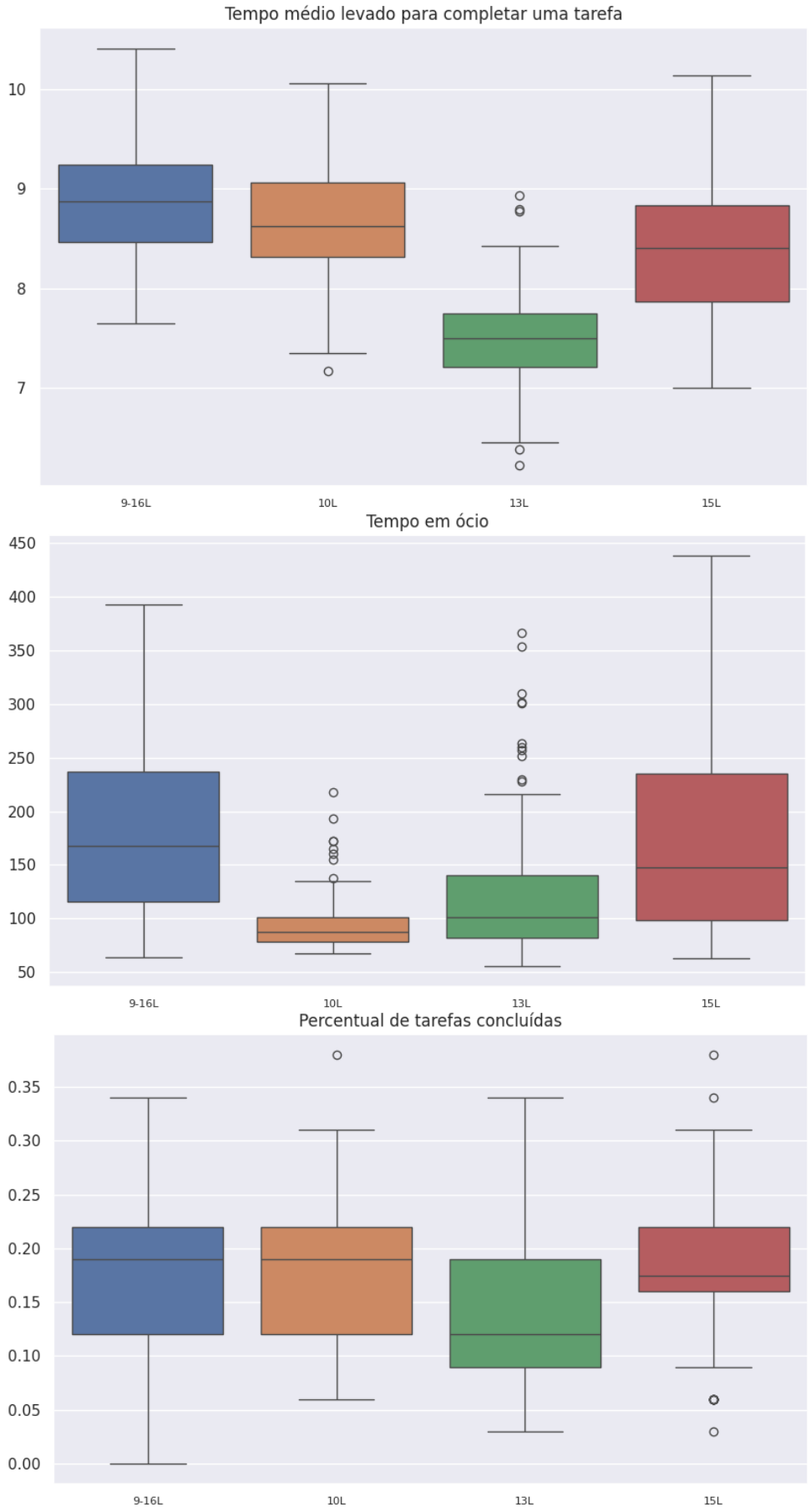


Figura 21: Visualização dos resultados obtidos no cenário 4

Fonte: Autor (2025)

4.3.2 Análise dos resultados exploratórios e experimentais

Ao analisar os resultados obtidos durante as duas fases exploratórias da pesquisa, observa-se que a divisão das tarefas, em entrega e coleta, teve um impacto significativo no desempenho das redes desenvolvidas, demonstrando claramente um aumento na complexidade do problema, ressaltando que nem sempre as soluções são compatíveis com versões mais complexas do mesmo problema. Conforme apresentado, o desenvolvido nesta dissertação obteve resultados bastante positivos para o PCV, mas quando aplicado ao PCEJT, não somente deixou muito a desejar, como também afetou negativamente o algoritmo EDD. O autor levantou algumas possíveis causas para o desempenho insatisfatório do proposto neste estudo, elas são apresentadas e discutidas a seguir.

Percebe-se que o AG obteve um desempenho inferior ao EDD em vários cenários, essa diferença acontece devido às características intrínsecas de cada abordagem. O EDD, sendo uma heurística determinística, que utiliza informações fornecidas pelo sistema, neste caso os tempos de encerramento das janelas de entrega, para realizar a ordenação das tarefas. Essa estrutura permite respostas mais rápidas e consistentes, especialmente em sistemas de baixa complexidade e restrições temporais bem definidas [59].

Já o AG, uma meta-heurística estocástica, realiza avaliações iterativas de uma população de soluções, não possuindo acesso direto às informações do sistema, precisando então, inferir padrões a partir dos resultados de sua função avaliativa [26]. Além disso, é amplamente reconhecido que AGs possuêm certa tendência em convergir para ótimos locais, especialmente em espaços de busca complexos, comprometendo sua capacidade de encontrar soluções globais [23, 5].

Durante o desenvolvimento deste trabalho, observou-se que, apesar da proposta apresentar resultados satisfatórios em cenários simplificados, seu desempenho foi bastante impactado quando aplicada a cenários mais complexos. Dessa forma, foram elencados alguns fatores que podem ter contribuído para estes, os quais são discutidos a seguir.

Uma das possíveis causas é a geração aleatória dos dados das tarefas, essa aleatoriedade, embora eficaz na simulação de situações diversas, pode resultar em distribuições desbalanceadas, com uma baixa representatividade dos casos críticos, ou então apresentar padrões inconsistentes. Essa limitação pode assim, ter comprometido a capacidade das redes em aprender os padrões relevantes ao sistema e generalizar para novas possíveis instâncias. Além disso, a escolha dos hiperparâmetros foi realizada tendo como base os testes pontuais realizados durante o início deste estudo, podendo assim ter restringido o desempenho das redes.

A forma como foi feito o treinamento das redes, diferente daquilo realizado por Bello et al. [6] em seu trabalho, o autor não realiza uma avaliação do resultado da rede mestra, utilizando o resultado final para realizar o aprendizado de ambas as redes. Por isso é muito provável que a rede mestra tenha sofrido uma deficiência durante seu processo de aprendizado, não atingindo o potencial completo da arquitetura e técnicas utilizadas.

Entretanto, idealizar e implantar o avaliador intermediário, realizar seus testes e seu refino, acabaria levando muito mais tempo do que aquele disponível para a realização deste estudo.

Ao comparar esta dissertação com o trabalho de Bello et al. [6] ficam evidentes algumas diferenças fundamentais que explicam a grande divergência entre os resultados. Bello foca em problemas clássicos como o PCV e o Problema da Mochila simplificados e controlados, já neste estudo é abordado um cenário mais verossímil e complexo, tornando o espaço de soluções exponencialmente maior e de mais difícil exploração. No que diz respeito à arquitetura, Bello utiliza *Pointer Networks*, uma estrutura altamente especializada para lidar com saídas permutáveis, como as rotas em um PCV, permitindo que o modelo aprenda diretamente uma sequência ótima de visitas às cidades. Já o autor desta dissertação tomou como inspiração apenas a dupla de redes agindo em conjunto, optando pela abordagem com as redes mestra e agente.

Outro ponto crucial é a estratégia de treino, Bello aplica técnicas avançadas de aprendizado por reforço, como *policy gradient* com busca ativa (*Active Search*) e o uso de uma rede intermediária, chamada de rede crítica, que avalia e estabiliza o aprendizado reduzindo sua variância. Em contrapartida, neste trabalho foi utilizado apenas o QL, sem mecanismos de avaliação intermediária, o que pode ter comprometido o aprendizado da rede mestra e, por consequência, a qualidade das soluções geradas. Além disso, em seu artigo, Bello explorou mecanismos de busca durante a inferência, enquanto neste estudo o autor usou uma abordagem mais direta.

Por fim, a infraestrutura disponível para a realização dos estudos teve papel bastante relevante, com o trabalho de Bello sendo desenvolvido no Google Brain, tendo acesso a recursos computacionais de alto desempenho e uma equipe especializada, permitindo uma maior exploração de arquiteturas e melhor refinamento dos hiperparâmetros. Já esta dissertação foi conduzida integralmente no equipamento pessoal do autor, com tempo limite para conclusão e recursos bastante escassos, limitando a exploração e experimentação.

Durante a exploração inicial, o autor obteve os melhores resultados normalizando todos os valores temporais, isto é, transformando os valores de cada categoria em uma escala de 0 a 1, com 0 sendo o menor dos valores e 1 o maior dos valores, mantendo os índices dos locais de coleta e entrega em sua forma normal, fazendo com que a rede dê importância maior aos locais do que às restrições de tempo. Esta escolha foi feita devido à alta variação nos valores atribuídos aos tempos, acabando por gerar uma falsa importância para valores que ocorrem naturalmente mais elevados, que neste estudo acontece quando comparamos os valores da abertura da carga ao encerramento da descarga. Enquanto os valores daquela são fixos num intervalo de 0 a 10 minutos, os valores desta se encontram em um intervalo de 15 a 30 minutos. Caso estas informações fossem passadas em sua forma natural para a rede, provavelmente haveria um viés interno que daria uma maior importância para os valores do encerramento das janelas de descarga, o que impactaria de

uma forma ainda mais negativa os resultados, pois no cenário estudado nenhuma destas informações temporais é mais importante que as outras.

A partir dos resultados obtidos durante os experimentos nos diferentes cenários, observa-se que o desempenho da rede com 10 camadas para ambas as redes obteve um maior sucesso na reordenação das tarefas na maioria dos cenários, com a rede com 15 camadas obtendo melhor desempenho no primeiro cenário. Observa-se também que, de uma forma geral, conforme é aumentada a complexidade dos cenários, o desempenho das redes piora, reforçando novamente que o proposto nesta dissertação não é viável da forma como foi concebida.

O autor também observou o percentual das subtarefas que foram concluídas com sucesso, e descobriu que um valor significativo destas foi completo. A partir disto, viu-se que as redes conseguiram em todos os momentos, realizar a grande parte das cargas, falhando ao realizar as entregas, sugerindo que, apesar de as redes conseguirem organizar a fila a ponto de realizar um número elevado de subtarefas, estas falham em realizar a conexão entre ambos tipos destas, resultando nos valores de tarefas concluídas observados. Demonstrando uma incapacidade, por parte das redes desenvolvidas, em realizar a visão do autor, onde as redes utilizariam da principal característica das LSTMs, sua memória interna, para aprender e agir sobre esta ligação entre os tipos de subtarefas.

4.4 Sumário do capítulo

Apesar de possuir resultados negativos quando aplicada em cenários mais realistas e complexos, as redes treinadas obtiveram resultados satisfatórios em sua aplicação no PCV simples. Com os resultados obtidos durante a primeira fase exploratória demonstrando uma real possibilidade da aplicação do proposto nesta dissertação. Assim, o autor ressalta a necessidade de uma maior exploração e refino, buscando outras estruturas de RNAs ou então uma reformulação e um maior refino da rede apresentada neste trabalho.

Neste capítulo foi expandido sobre a solução desenvolvida, explicando o funcionamento dos algoritmos utilizados durante os experimentos e apresentando os resultados obtidos, bem como as dificuldades enfrentadas durante o percurso deste estudo. O próximo capítulo apresenta as conclusões do autor sobre o estudo realizado bem como sugestões para trabalhos futuros.

5 CONCLUSÃO

A partir dos resultados obtidos, percebe-se que no cenário mais simples a solução proposta obteve resultados bastante promissores, elevando os resultados obtidos pelo algoritmo base, até mesmo superando os resultados obtidos pelo AG, um dos algoritmos mais utilizados para solucionar problemas como o retratado. Entretanto, quando a proposta foi aplicada no cenário real, houve uma perda considerável na eficácia do EDD, o que aponta ao fato de que, enquanto viável para solucionar um PCV, quando aplicada ao PCEJT, a arquitetura proposta e utilizada pelo autor não possui capacidade de solucionar problemas de alta complexidade. Apesar dos resultados obtidos pelo autor, diversos trabalhos, como os de Bello et al. [6], Ferretti e Marchi [24] e Wang, Jiang e Yue [70], demonstram que é sim viável utilizar RNAs como solução para problemas logísticos como o estudado.

A partir desses estudos, percebe-se que, enquanto existe um futuro promissor para a solução apresentada quando aplicada em cenários de baixa complexidade, como o PCV de agente único explorado neste estudo, no momento em que a proposta é aplicada em um cenário de alta complexidade, emulando um ambiente mais realista, ela falha em obter resultados positivos, causando a perda de desempenho do algoritmo base utilizado. No cenário simples, sem muitas restrições, mesmo com a simplicidade do algoritmo utilizado, já foi possível observar melhorias consideráveis em seu desempenho, tornando-o uma abordagem viável para lidar com a complexidade inerente à distribuição de tarefas, a partir do fato de que o EDD superou algoritmos genéticos, reconhecidos como alguns dos melhores solucionadores para esse problema no mercado atual.

Entretanto, quando a solução foi aplicada no cenário mais realista deste estudo, não somente a proposta de solução original falhou em obter resultados positivos, mas teve um desempenho suficientemente ineficiente para justificar uma pivotagem da maneira como a solução foi desenvolvida. Com a nova forma da solução, resultados superiores aos originais foram obtidos, mas mesmo com estes resultados superiores, a solução falhou em alcançar resultados significativos, demonstrando-se incapaz de atuar como alternativa viável para solucionar o problema estudado. Dessa forma, como pode ser observado através dos diferentes artigos que tratam sobre o tema da utilização de RNAs para encontrar

soluções para o PCV e suas variações, esta é uma área que vem sendo explorada, com resultados bastante promissores.

Assim, para futuros trabalhos, o autor sugere a união de um algoritmo Solver com um modelo de linguagem de grande escala (LLM), modificar o AG para receber a fila pré-ordenada a fim de diminuir as diferenças entre este e o EDD, modificar o sistema para utilizar uma meta-heurística durante os treinos das redes. Além destas propostas, é necessária também uma exploração de diferentes arquiteturas a fim de contornar os problemas apresentados na seção anterior. O código utilizado neste trabalho pode ser encontrado integralmente em github.com/clevertonbsj/dissertation-code.

BIBLIOGRAFIA

[1] Reza Javanmard Alitappeh e Kossar Jeddisaravi. “Multi-robot exploration in task allocation problem”. en. Em: *Applied Intelligence* 52.2 (jan. de 2022), pp. 2189–2211. ISSN: 0924-669X, 1573-7497. DOI: 10.1007/s10489-021-02483-3. URL: https://link.springer.com/10.1007/s10489-021-02483-3 (acesso em 04/09/2023).

[2] Ivan Antoniuk et al. “Methodology of design and optimization of internal logistics in the concept of Industry 4.0”. en. Em: *Transportation Research Procedia* 55 (2021), pp. 503–509. ISSN: 23521465. DOI: 10.1016/j.trpro.2021.07.093. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352146521005068 (acesso em 04/09/2023).

[3] J. Ashayeri e L.F. Gelders. “Warehouse design optimization”. en. Em: *European Journal of Operational Research* 21.3 (set. de 1985), pp. 285–294. ISSN: 03772217. DOI: 10.1016/0377-2217(85)90149-3. URL: https://linkinghub.elsevier.com/retrieve/pii/0377221785901493 (acesso em 26/07/2024).

[4] N. Atay e B. Bayazit. “Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation ProblemAllocation Problem”. Em: (2006), p. 11. URL: https://openscholarship.wustl.edu/cgi/viewcontent.cgi?article=1204&context=cse_research.

[5] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[6] Irwan Bello et al. *Neural Combinatorial Optimization with Reinforcement Learning*. arXiv:1611.09940 [cs]. Jan. de 2017. DOI: 10.48550/arXiv.1611.09940. URL: http://arxiv.org/abs/1611.09940 (acesso em 28/01/2025).

[7] D. Britz. *Recurrent Neural Networks Tutorial, Part 1 – Introduction to RNNs*. en. Set. de 2015. URL: https://dennybritz.com/posts/wildml/

recurrent-neural-networks-tutorial-part-1/ (acesso em 23/10/2023).

[8] Giulia Bruno e Dario Antonelli. "Dynamic task classification and assignment for the management of human-robot collaborative teams in workcells". en. Em: *The International Journal of Advanced Manufacturing Technology* 98.9-12 (out. de 2018), pp. 2415–2427. ISSN: 0268-3768, 1433-3015. DOI: 10.1007/s00170-018-2400-4. URL: http://link.springer.com/10.1007/s00170-018-2400-4 (acesso em 04/09/2023).

[9] Natalia Burganova et al. "Optimalisation of Internal Logistics Transport Time Through Warehouse Management: Case Study". en. Em: *Transportation Research Procedia* 55 (2021), pp. 553–560. ISSN: 23521465. DOI: 10.1016/j.trpro.2021.07.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S2352146521004178 (acesso em 04/09/2023).

[10] E. K. Burke e Bykov. "A late acceptance strategy in hill-climbing for examination timetabling problems". Em: 2008. URL: https://www.patatconference.org/patat2008/proceedings/Bykov-HC2a.pdf (acesso em 28/10/2023).

[11] Edmund K. Burke et al. "A Classification of Hyper-Heuristic Approaches: Revisited". en. Em: *Handbook of Metaheuristics*. Ed. por Michel Gendreau e Jean-Yves Potvin. International Series in Operations Research & Management Science. Cham: Springer International Publishing, 2019, pp. 453–477. ISBN: 978-3-319-91086-4. DOI: 10.1007/978-3-319-91086-4_14. URL: https://doi.org/10.1007/978-3-319-91086-4_14 (acesso em 28/10/2023).

[12] Timo Bänziger, Andreas Kunz e Konrad Wegener. "Optimizing human–robot task allocation using a simulation tool based on standardized work descriptions". en. Em: *Journal of Intelligent Manufacturing* 31.7 (out. de 2020), pp. 1635–1648. ISSN: 0956-5515, 1572-8145. DOI: 10.1007/s10845-018-1411-1. URL: http://link.springer.com/10.1007/s10845-018-1411-1 (acesso em 04/09/2023).

[13] Grace Cai, Noble Harasha e Nancy Lynch. *A Comparison of New Swarm Task Allocation Algorithms in Unknown Environments with Varying Task Density*. Version Number: 3. 2022. DOI: 10.48550/ARXIV.2212.00844. URL: https://arxiv.org/abs/2212.00844 (acesso em 30/10/2023).

[14] L. Cavecchia et al. "Mixed Integer Linear Programming Models for Logistics Optimization". Em: *Journal of Operations Research* 45.2 (2017), pp. 123–135.

[15] Shushman Choudhury et al. *Dynamic Multi-Robot Task Allocation under Uncertainty and Temporal Constraints*. en. arXiv:2005.13109 [cs]. Jul. de 2020. URL: http://arxiv.org/abs/2005.13109 (acesso em 04/09/2023).

[16] Rodrigo Furlan De Assis et al. "Machine Learning in Warehouse Management: A Survey". en. Em: *Procedia Computer Science* 232 (2024), pp. 2790–2799. ISSN: 18770509. DOI: 10.1016/j.procs.2024.02.096. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050924002734 (acesso em 07/06/2024).

[17] M. De Ryck, M. Versteyhe e F. Debrouwere. "Automated guided vehicle systems, state-of-the-art control algorithms and techniques". en. Em: *Journal of Manufacturing Systems* 54 (jan. de 2020), pp. 152–173. ISSN: 02786125. DOI: 10.1016/j.jmsy.2019.12.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S0278612519301177 (acesso em 04/09/2023).

[18] Li Deng e Dong Yu. "Deep Learning: Methods and Applications". en-US. Em: (mai. de 2014). URL: https://www.microsoft.com/en-us/research/publication/deep-learning-methods-and-applications/ (acesso em 23/10/2023).

[19] Xiaoheng Deng et al. "Task allocation algorithm and optimization model on edge collaboration". en. Em: *Journal of Systems Architecture* 110 (nov. de 2020), p. 101778. ISSN: 13837621. DOI: 10.1016/j.sysarc.2020.101778. URL: https://linkinghub.elsevier.com/retrieve/pii/S1383762120300722 (acesso em 04/09/2023).

[20] Dinesh Dhoka e Choudary Dr. Y. Lokeswara. "ABC Classification for Inventory Optimization". Em: 15.1 (2013), pp. 38–41. ISSN: 2319-7668.

[21] Mariagrazia Dotoli et al. "An integrated approach for warehouse analysis and optimization: A case study". en. Em: *Computers in Industry* 70 (jun. de 2015), pp. 56–69. ISSN: 01663615. DOI: 10.1016/j.compind.2014.12.004. URL: https://linkinghub.elsevier.com/retrieve/pii/S0166361514002097 (acesso em 26/07/2024).

[22] George Dunn et al. *Deep Reinforcement Learning for Picker Routing Problem in Warehousing*. en. arXiv:2402.03525 [cs]. Fev. de 2024. URL: http://arxiv.org/abs/2402.03525 (acesso em 07/06/2024).

[23] A.E. Eiben e J.E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[24] Ivan Ferretti e Beatrice Marchi. "Q-Learning for Inventory Management: an application case". en. Em: *Procedia Computer Science* 232 (2024), pp. 2431–2439. ISSN: 18770509. DOI: 10.1016/j.procs.2024.02.062. URL: https://

linkinghub.elsevier.com/retrieve/pii/S1877050924002394 (acesso em 07/06/2024).

[25] Bezalel Gavish e Stephen C. Graves. “The Travelling Salesman Problem and Related Problems”. Em: 1978. URL: https://api.semanticscholar.org/CorpusID:14353795.

[26] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[27] Sepp Hochreiter e Jürgen Schmidhuber. “Long Short-Term Memory”. en. Em: *Neural Computation* 9.8 (nov. de 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: 10.1162/neco.1997.9.8.1735. URL: https://direct.mit.edu/neco/article/9/8/1735-1780/6109 (acesso em 04/09/2023).

[28] Aamal Abbas Hussain, Francesco Belardinelli e Georgios Piliouras. *Asymptotic Convergence and Performance of Multi-Agent Q-Learning Dynamics*. Version Number: 1. 2023. DOI: 10.48550/ARXIV.2301.09619. URL: https://arxiv.org/abs/2301.09619 (acesso em 25/10/2023).

[29] Dmitry Ivanov, Ilya Zisman e Kirill Chernyshev. “Mediated Multi-Agent Reinforcement Learning”. Em: (2023). Publisher: arXiv Version Number: 1. DOI: 10.48550/ARXIV.2306.08419. URL: https://arxiv.org/abs/2306.08419 (acesso em 20/10/2023).

[30] Xiao Jia e M. Meng. “A survey and analysis of task allocation algorithms in multi-robot systems”. en. Em: *IEEE International Conference on Robotics and Biomimetics* (2013). URL: https://www.semanticscholar.org/paper/A-survey-and-analysis-of-task-allocation-algorithms-Jia-Meng/03a853a66bbe0ed115a6df54ea5e6bed1a8c4214 (acesso em 28/10/2023).

[31] Cleverton Bueno dos Santos Júnior. “Path-planning utilizando uma rede neural artificial de aprendizado profundo”. por. Trabalho de Conclusão de Curso. Santa Maria, RS: Universidade Federal de Santa Maria, 2021. URL: http://repositorio.ufsm.br/handle/1/21381 (acesso em 23/10/2023).

[32] Alaa Khamis, Ahmed Hussein e Ahmed Elmogy. “Multi-robot Task Allocation: A Review of the State-of-the-Art”. en. Em: *Cooperative Robots and Sensor Networks 2015*. Ed. por Anis Koubâa e J.Ramiro Martínez-de Dios. Studies in Computational Intelligence. Cham: Springer International Publishing, 2015, pp. 31–51. ISBN: 978-3-319-18299-5. DOI: 10.1007/978-3-319-18299-5_2. URL: https://doi.org/10.1007/978-3-319-18299-5_2 (acesso em 28/10/2023).

[33] Kap Hwan Kim e Jong Wook Bae. “A Look-Ahead Dispatching Method for Automated Guided Vehicles in Automated Port Container Terminals”. en. Em: *Transportation Science* 38.2 (mai. de 2004), pp. 224–234. ISSN: 0041-1655, 1526-5447. DOI: 10.1287/trsc.1030.0082. URL: https://pubsonline.informs.org/doi/10.1287/trsc.1030.0082 (acesso em 04/09/2023).

[34] Adam Klyne e Kathryn Merrick. “Task Allocation Using Particle Swarm Optimisation and Anomaly Detection to Generate a Dynamic Fitness Function”. en. Em: *AI 2015: Advances in Artificial Intelligence*. Ed. por Bernhard Pfahringer e Jochen Renz. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2015, pp. 317–329. ISBN: 978-3-319-26350-2. DOI: 10.1007/978-3-319-26350-2_28.

[35] Wojciech Kmiecik et al. “Task Allocation in Mesh Connected Processors with Local Search Meta-heuristic Algorithms”. en. Em: *Intelligent Information and Database Systems*. Ed. por Ngoc Thanh Nguyen, Manh Thanh Le e Jerzy Świątek. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 215–224. ISBN: 978-3-642-12101-2. DOI: 10.1007/978-3-642-12101-2_23.

[36] N. Kokash. “An introduction to heuristic algorithms”. Em: 2005. URL: https://www.semanticscholar.org/paper/An-introduction-to-heuristic-algorithms-Kokash/8314bf30780871868076775ba62759f1faf8c9f0 (acesso em 28/10/2023).

[37] J. Kratika et al. “Solving the task assignment problem with a variable neighborhood search”. Em: *Serdica Journal of Computing* 4.4 (2010), 435p–446p. URL: http://sci-gems.math.bas.bg:8080/jspui/handle/10525/1604.

[38] Xu Li, Zhengyan Liu e Fuxiao Tan. “Multi-Robot Task Allocation Based on Cloud Ant Colony Algorithm”. en. Em: *Neural Information Processing*. Ed. por Derong Liu et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 3–10. ISBN: 978-3-319-70093-9. DOI: 10.1007/978-3-319-70093-9_1.

[39] Hongtao Liang e Fengju Kang. “A novel task optimal allocation approach based on Contract Net Protocol for Agent-oriented UUV swarm system modeling”. Em: *Optik* 127.8 (abr. de 2016), pp. 3928–3933. ISSN: 0030-4026. DOI: 10.1016/j.ijleo.2016.01.071. URL: https://www.sciencedirect.com/science/article/pii/S0030402616001200 (acesso em 28/10/2023).

[40] Seppo Linnainmaa. “Taylor expansion of the accumulated rounding error”. en. Em: *BIT Numerical Mathematics* 16.2 (jun. de 1976), pp. 146–160. ISSN: 1572-9125.

DOI: 10.1007/BF01931367. URL: https://doi.org/10.1007/BF01931367 (acesso em 30/10/2023).

[41] Chun Liu e Andreas Kroll. “Memetic algorithms for optimal task allocation in multi-robot systems for inspection problems with cooperative tasks”. en. Em: *Soft Computing* 19.3 (mar. de 2015), pp. 567–584. ISSN: 1433-7479. DOI: 10.1007/s00500-014-1274-0. URL: https://doi.org/10.1007/s00500-014-1274-0 (acesso em 28/10/2023).

[42] Yiming Liu, Mengxia Chen e Hejiao Huang. “Multi-agent Pathfinding Based on Improved Cooperative A* in Kiva System”. Em: *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*. ISSN: 2251-2446. Abr. de 2019, pp. 633–638. DOI: 10.1109/ICCAR.2019.8813319. URL: https://ieeexplore.ieee.org/abstract/document/8813319 (acesso em 23/10/2023).

[43] C. A. Mendez e J. Cerda. “An MILP framework for batch reactive scheduling with limited intermediate storage”. Em: *Computers & Chemical Engineering* 30.4 (2006), pp. 614–634.

[44] Li Minglei, Wang Hongwei e Qi Chao. “A novel HTN planning approach for handling disruption during plan execution”. en. Em: *Applied Intelligence* 46.4 (jun. de 2017), pp. 800–809. ISSN: 1573-7497. DOI: 10.1007/s10489-016-0865-0. URL: https://doi.org/10.1007/s10489-016-0865-0 (acesso em 28/10/2023).

[45] Tom M. Mitchell. *Machine Learning*. McGraw-Hill series in computer science. New York: McGraw-Hill, 1997. ISBN: 978-0-07-042807-2.

[46] Ababneh Mohammad, Oqeili Saleh e Rawan A. Abdeen. “Occurrences Algorithm for String Searching Based on Brute-force Algorithm”. Em: *Journal of Computer Science* 2.1 (jan. de 2006), pp. 82–85. ISSN: 15493636. DOI: 10.3844/jcssp.2006.82.85. URL: http://www.thescipub.com/abstract/?doi=jcssp.2006.82.85 (acesso em 28/10/2023).

[47] Alejandro R Mosteo e Luis Montano. “A survey of multi-robot task allocation A survey of multi-robot task allocation”. en. Em: (2010). DOI: 10.13140/2.1.2442.3688. URL: http://rgdoi.net/10.13140/2.1.2442.3688 (acesso em 28/10/2023).

[48] Abdulrahman Nahhas, Andrey Kharitonov e Klaus Turowski. “Deep Reinforcement Learning for Solving Allocation Problems in Supply Chain: An Image-Based Observation Space”. en. Em: *Procedia Computer Science* 232 (2024), pp. 2570–2579. ISSN: 18770509. DOI: 10.1016/j.procs.2024.02.

075. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050924002527 (acesso em 07/06/2024).

[49] Vinod Nair e Geoffrey E. Hinton. "Rectified linear units improve restricted boltzmann machines". Em: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Madison, WI, USA: Omnipress, jun. de 2010, pp. 807–814. ISBN: 978-1-60558-907-7. (Acesso em 23/10/2023).

[50] Sergio Nesmachnow. "An overview of metaheuristics: accurate and efficient methods for optimisation". en. Em: *International Journal of Metaheuristics* 3.4 (2014), p. 320. ISSN: 1755-2176, 1755-2184. DOI: 10.1504/IJMHEUR.2014.068914. URL: http://www.inderscience.com/link.php?id=68914 (acesso em 28/10/2023).

[51] Ernesto Nunes e Maria Gini. "Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints". en. Em: *Proceedings of the AAAI Conference on Artificial Intelligence* 29.1 (fev. de 2015). ISSN: 2374-3468. DOI: 10.1609/aaai.v29i1.9440. URL: https://ojs.aaai.org/index.php/AAAI/article/view/9440 (acesso em 28/10/2023).

[52] C OLAH. *Understanding LSTM Networks*. Ago. de 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[53] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". Em: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019. URL: https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (acesso em 23/10/2023).

[54] Jennifer A. Pazour e Héctor J. Carlo. "Warehouse reshuffling: Insights and optimization". en. Em: *Transportation Research Part E: Logistics and Transportation Review* 73 (jan. de 2015), pp. 207–226. ISSN: 13665545. DOI: 10.1016/j.tre.2014.11.002. URL: https://linkinghub.elsevier.com/retrieve/pii/S1366554514001914 (acesso em 26/07/2024).

[55] J. Pearl. "Heuristics: Intelligent search strategies for computer problem solving". English. Em: (jan. de 1984). URL: https://www.osti.gov/biblio/5127296 (acesso em 27/10/2023).

[56] Matheus de Mattos Pereira. "Aprendizado profundo: redes LSTM". por. Tese de dout. Dourados, MS: Universidade Federal da Grande Dourados, mar. de 2017. URL: http://repositorio.ufgd.edu.br/jspui/handle/prefix/2887 (acesso em 23/10/2023).

[57] Laurent Perron e Vincent Furnon. *OR-Tools*. Versão v9.10. Google, 7 de mai. de 2024. URL: https://developers.google.com/optimization/.

[58] Saravana Perumaal Subramanian e Selva Kumar Chandrasekar. "Simultaneous allocation and sequencing of orders for robotic mobile fulfillment system using reinforcement learning algorithm". en. Em: *Expert Systems with Applications* 239 (abr. de 2024), p. 122262. ISSN: 09574174. DOI: 10.1016/j.eswa.2023.122262. URL: https://linkinghub.elsevier.com/retrieve/pii/S0957417423027641 (acesso em 07/06/2024).

[59] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer, 2016.

[60] A. Q. R. S. Queiroz. "CONTROLE DE UM SISTEMA VARIANTE NO TEMPO USANDO REDES NEURAIS ARTIFICIAIS E APRENDIZADO POR REFORÇO". por. TCC. Brasília, DF: Universidade de Brasília, 2018. URL: http://www.ene.unb.br/adolfo/Monographs/Graduation/TG2018%20Alvaro%20Queiroz.pdf (acesso em 23/10/2023).

[61] Humyun Fuad Rahman e Izabela Nielsen. "Scheduling automated transport vehicles for material distribution systems". en. Em: *Applied Soft Computing* 82 (set. de 2019), p. 105552. ISSN: 15684946. DOI: 10.1016/j.asoc.2019.105552. URL: https://linkinghub.elsevier.com/retrieve/pii/S1568494619303321 (acesso em 04/09/2023).

[62] P. Ralevic et al. "A DYNAMIC PROGRAMMING APPROACH ALGORITHM FOR THE ALLOCATION OF LIMITED RESOURCES". Em: *Metalurgia International* 17 (2012), 91p–93p. URL: https://www.researchgate.net/publication/295634930_A_DYNAMIC_PROGRAMMING_APPROACH_ALGORITHM_FOR_THE_ALLOCATION_OF_LIMITED_RESOURCES (acesso em 28/10/2023).

[63] David E. Rumelhart, Geoffrey E. Hinton e Ronald J. Williams. "Learning representations by back-propagating errors". en. Em: *Nature* 323.6088 (out. de 1986), pp. 533–536. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/323533a0. URL: https://www.nature.com/articles/323533a0 (acesso em 30/10/2023).

[64] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". Em: *IBM Journal of Research and Development* 3.3 (jul. de 1959), pp. 210–229. ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: https://ieeexplore.ieee.org/abstract/document/5392560/authors#authors (acesso em 23/10/2023).

[65] Robert Sanders. "THE PARETO PRINCIPLE: ITS USE AND ABUSE". Em: 1.2 (1987), pp. 37–40. DOI: 10.1108/eb024706.

[66] A. Santiago et al. “An Iterative Local Search Algorithm for Scheduling Precedence-Constrained Applications on Heterogeneous Machines”. Em: 2013. URL: https://www.researchgate.net/publication/256487129_An_Iterative_Local_Search_Algorithm_for_Scheduling_Precedence-Constrained_Applications_on_Heterogeneous_Machines (acesso em 28/10/2023).

[67] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. Em: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958. ISSN: 1533-7928. URL: http://jmlr.org/papers/v15/srivastava14a.html (acesso em 23/10/2023).

[68] Guido Van Rossum e Fred L. Drake. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN: 1441412697.

[69] T. Vidal e J. Bidot. “Dynamic task sequencing in temporal problems with uncertainty”. Em: *Proceedings of CP’01 workshop on Constraints and Uncertainty*. Springer Verlag, 2001, p. 8.

[70] Yunrui Wang, Ziqiang Jiang e Wu Yue. *Model Construction of Material Distribution System Based on Digital Twin*. en. preprint. In Review, nov. de 2021. DOI: 10.21203/rs.3.rs-1018310/v1. URL: https://www.researchsquare.com/article/rs-1018310/v1 (acesso em 04/09/2023).

[71] Christopher J.C.H. Watkins e Peter Dayan. “Technical Note: Q-Learning”. Em: *Machine Learning* 8.3/4 (1992), pp. 279–292. ISSN: 08856125. DOI: 10.1023/A:1022676722315. URL: http://link.springer.com/10.1023/A:1022676722315 (acesso em 04/09/2023).

[72] Łukasz Wojciechowski, Tadeusz Cisowski e Arkadiusz Małek. “Route optimization for city cleaning vehicle”. en. Em: *Open Engineering* 11.1 (jan. de 2021), pp. 483–498. ISSN: 2391-5439. DOI: 10.1515/eng-2021-0049. URL: https://www.degruyter.com/document/doi/10.1515/eng-2021-0049/html (acesso em 04/09/2023).

[73] Peter R. Wurman, Raffaello D’Andrea e Mick Mountz. “Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses”. en. Em: *AI Magazine* 29.1 (mar. de 2008), pp. 9–9. ISSN: 2371-9621. DOI: 10.1609/aimag.v29i1.2082. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2082 (acesso em 23/10/2023).

[74] Dong Yang, Yaohua Wu e Wenkai Ma. “RETRACTED: Optimization of storage location assignment in automated warehouse”. en. Em: *Microprocessors and Microsystems* 80 (fev. de 2021), p. 103356. ISSN: 01419331. DOI: 10.1016/j.micpro.2020.103356. URL: https://linkinghub.elsevier.com/retrieve/pii/S0141933120305159 (acesso em 26/07/2024).

A MODELO MATEMÁTICO PARA EXECUÇÃO DE TAREFAS POR AGVS

A.1 Estado Inicial do Agente

Para cada agente $a \in \{1, 2, \ldots, n\}$, definimos:

- Tamanho da fila de tarefas: N
- Tempo decorrido: $T_a(0)$
- Posição atual: $P_a(0)$
- Tempo ocioso acumulado: $O_a(0) = 0$
- Vetor de carga:

$$\text{Carga}_a(0) = [0, 0, \ldots, 0]$$

Cada agente possui uma sequência de tarefas:

$$Q_a = \{q_{a,1}, q_{a,2}, \ldots, q_{a,m}\}, \quad m = \frac{N}{n}$$

Cada tarefa q tem tipo $X \in \{C, D\}$, onde:

- C: Carregamento
- D: Descarregamento

A.2 Parâmetros da Tarefa

Para cada tarefa q do tipo X, definimos:

- Local de execução:

$$\text{Local}_X(q) = \begin{cases} \text{Local}_C(q), & \text{se } X = C \\ \text{Local}_D(q), & \text{se } X = D \end{cases}$$

- Janela de operação:

$$\text{Início}_X(q), \quad \text{Fim}_X(q)$$

- Tempo de operação:

$$\text{Oper}_X(q)$$

- Custo de deslocamento:

$$d_{i,j}$$

A.3 Restrição de movimento

Se:

$$\text{Local}_X(q_{a,i}) = P_a(i-1) \quad \text{e} \quad q_{a,i} \neq q_{a,i-1}$$

então:

- Tempo extra: $\Delta_m = d_{a,i} + 5$
- Penalidade: π_m

A.4 Atualização do Estado

A.4.1 Tempo de chegada:

$$d = A(P_a(i-1), \text{Local}_X(q_{a,i})), \quad S = T_a(i-1) + d + \Delta_m$$

A.4.2 Atualização do tempo e ociosidade:

- **Caso 1 – Antes da janela:**

$$T_a(i) = T_a(i-1) + \text{Início}_X(q_{a,i}) + \text{Oper}_X(q_{a,i})$$

$$O_a(i) = O_a(i-1) + (\text{Início}_X(q_{a,i}) - T_a(i-1)) + \text{Oper}_X(q_{a,i})$$

- **Caso 2 – Dentro da janela:**

$$T_a(i) = T_a(i-1) + \text{Oper}_X(q_{a,i}), \quad O_a(i) = O_a(i-1) + \text{Oper}_X(q_{a,i})$$

- **Caso 3 – Após a janela:**

$$T_a(i) = T_a(i-1) + 2d + \text{Oper}_X(q_{a,i}) + \pi_m$$

$$O_a(i) = O_a(i-1)$$

A.4.3 Atualização da posição:

$$P_a(i) = \text{Local}_X(q_{a,i}),$$

A.5 Gerenciamento da Carga

- **Se** $X = C$: inserir $q_{a,i}$ se houver espaço (0) no vetor de carga.
- **Se** $X = D$: remover $q_{a,i}$ se estiver presente no vetor de carga.

A.6 Sincronização via gatilhos

A.6.1 Definição

Seja gatilho$[k]$ um vetor de disparadores indexado por:

$$k = \left\lfloor \frac{i}{2} \right\rfloor$$

Para cada ação do agente $X \in \{C, D\}$, define-se um limiar θ_X.

A.6.2 Condição de ativação

A sincronização é ativada quando:

$$\text{gatilho}[k] \geq \theta_X$$

A.6.3 Efeitos da ativação

- Atualiza o tempo de referência da tarefa:

$$\text{Início}_x(q) = \text{Início}_x(q) + T_a(i-1)$$

- Sinaliza mudança de estado:

$$\text{disparo} \leftarrow \text{Verdadeiro}$$

A.7 Resumo Final

Para cada tarefa $q_{a,i}$, o agente atualiza:

- Tempo decorrido $T_a(i)$
- Tempo ocioso $O_a(i)$

- Posição atual $P_a(i)$
- Vetor de carga $\mathrm{Carga}_a(i)$

Considerando:

- Deslocamento via matriz A
- Reposicionamento se necessário
- Janelas de tempo
- Penalidades por falhas
- Sincronização por gatilhos

B SIMULAÇÃO DE AGENTE

B.1 Estado Inicial do Agente

Para o agente a:

$$T_0 = 0, \quad P_0 = 0, \quad O_0 = 0, \quad \text{Carga}_0 = [0, 0, 0, 0, 0]$$

B.2 Resultados da Simulação

- **Subtarefa 1.C (Carregamento, Local 3, Janela [5, 15], Oper = 4)**: Sucesso

$$d = A(0, 3) = 4, \quad S = 0 + 4 = 4 < 5 \Rightarrow \text{espera até } 5$$

$$T_1 = 5 + 2 = 6, \quad O_1 = 1, \quad P_1 = 2 \quad \text{Carga}_1 = [1, 0, 0, 0, 0]$$

- **Subtarefa 1.D (Descarregamento, Local 5, Janela [20, 30], Oper = 2)**: Sucesso

$$d = A(3, 5) = 6, \quad S = 6 + 6 = 12 < 20 \Rightarrow \text{espera até } 20$$

$$T_2 = 20 + 2 = 22, \quad O_2 = 1 + 8 = 9, \quad P_2 = 4 \quad \text{Carga}_2 = [0, 0, 0, 0, 0]$$

- **Subtarefa 2.C (Carregamento, Local 2, Janela [30, 40], Oper = 6)**: Sucesso

$$d = A(5, 2) = 5, \quad S = 22 + 5 = 27 < 30 \Rightarrow \text{espera até} 30$$

$$T_3 = 30 + 6 = 36, \quad O_3 = 9 + 3 = 12, \quad P_3 = 5 \quad \text{Carga}_3 = [2, 0, 0, 0, 0]$$

- **Subtarefa 2.D (Descarregamento, Local 6, Janela [35, 45], Oper = 9)**: Sucesso

$$d = A(5, 6) = 6, \quad S = 36 + 6 = 42 < 45$$

$$T_4 = 42 + 6 = 48, \quad O_4 = 12, \quad P_4 = 5 \quad \text{Carga}_4 = [0, 0, 0, 0, 0]$$

- **Subtarefa 3.C (Carregamento, Local 4, Janela [35, 45], Oper = 1)**: Sucesso

$$d = A(5,4) = 3, \quad S = 42 + 3 = 45 \leq 45$$

$$T_5 = 45 + 1 = 46, \quad O_5 = 12, \quad P_5 = 4 \quad \text{Carga}_5 = [3,0,0,0,0]$$

- **Subtarefa 3.D (Descarregamento, Local 7, Janela [50, 60], Oper = 6)**: Sucesso

$$d = A(5,7) = 7, \quad S = 46 + 7 = 53 < 60$$

$$T_6 = 53 + 6 = 59, \quad O_6 = 12, \quad P_) = 7 \quad \text{Carga}_6 = [0,0,0,0,0]$$

- **Subtarefa 4.C (Carregamento, Local 1, Janela [45, 55], Oper = 8)**: Falha (chegada fora da janela)

$$d = A(5,1) = 6, \quad S = 59 + 6 = 65 > 55 \Rightarrow \text{falha}$$

$$T_7 = 65, \quad O_7 = 12, \quad P_7 = 1 \quad \text{Carga}_7 = [0,0,0,0,0]$$

- **Subtarefa 4.D (Descarregamento, Local 8, Janela [60, 75], Oper = 3)**: Falha (Sem carga presente)

$$d = A(1,8) = 8, \quad S = 65 + 8 = 73 < 75, \quad 4 \notin \text{Carga}_7 \Rightarrow \text{falha}$$

$$T_8 = 73, \quad O_8 = 12, \quad P_8 = 8 \quad \text{Carga}_8 = [0,0,0,0,0]$$

- **Subtarefa 5.C (Carregamento, Local 3, Janela [75, 85], Oper = 5)**: Sucesso

$$d = A(8,3) = 6, \quad S = 73 + 6 = 79 < 85$$

$$T_9 = 79 + 5 = 84, \quad O_9 = 12, \quad P_9 = 8 \quad \text{Carga}_9 = [5,0,0,0,0]$$

- **Subtarefa 5.D (Descarregamento, Local 9, Janela [75, 85], Oper = 5)**: Falha (chegada fora da janela)

$$d = A(5,9) = 9, \quad S = 84 + 9 = 93 > 85 \Rightarrow \text{falha}$$

$$T_{10} = 84, \quad O_{10} = 12, \quad P_{10} = 9 \quad \text{Carga}_{10} = [5,0,0,0,0]$$

Estado Final do Sistema:

$$T_f = 84, \quad O_f = 12, \quad P_f = 9, \quad \text{Carga}_f = [5,0,0,0,0] \quad \text{Sucessos} = [1,2,3] \quad \text{Falhas} = [4,5]$$