UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Ciências Aplicadas

Vothan Salomão Dos Santos

Otimização da Operação de Abastecimento de Linhas de Produção e Processos Industriais Através da Integração de Técnicas de Aprendizado por Reforço e Simulação: Um Estudo com o Proximal Policy Optimization (PPO) no Ambiente FlexSim

Limeira
2023

Vothan Salomão Dos Santos

Otimização da Operação de Abastecimento de Linhas de Produção e Processos Industriais Através da Integração de Técnicas de Aprendizado por Reforço e Simulação: Um Estudo com o Proximal Policy Optimization (PPO) no Ambiente FlexSim

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de Bacharel em Engenharia de Produção à Faculdade de Ciências Aplicadas da Universidade Estadual de Campinas.

Orientador: Prof. Dr. Anibal Tavares de Azevedo

Este trabalho corresponde à versão final da Trabalho de Conclusão de Curso defendida por Vothan Salomão Dos Santos e orientada pelo Prof. Dr. Anibal Tavares de Azevedo.

Limeira
2023

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Faculdade de Ciências Aplicadas
Ana Luiza Clemente de Abreu Valério - CRB 8/10669

Sa59o

Santos, Vothan Salomão dos, 1997-

Otimização da operação de abastecimento de linhas de produção em processos industriais através da integração de técnicas de aprendizado por reforço e simulação : um estudo com o Proximal Policy Optimization (PPO) no ambiente *FlexSim* / Vothan Salomão dos Santos. – Limeira, SP : [s.n.], 2023.

Orientador: Anibal Tavares de Azevedo.

Trabalho de Conclusão de Curso (graduação) – Universidade Estadual de Campinas, Faculdade de Ciências Aplicadas.

1. Aprendizagem por reforço. 2. Automação industrial. 3. Otimização. 4. Simulação. I. Azevedo, Anibal Tavares de,1977-. II. Universidade Estadual de Campinas. Faculdade de Ciências Aplicadas. III. Título.

Informações adicionais, complementares

Título em outro idioma: Optimization of supply operation for production lines and industrial processes through the integration of reinforcement learning techniques and simulation: a study with Proximal Policy Optimization (PPO) in the FlexSim environment
Palavras-chave em inglês:
Reinforcement learning
Industrial automation
Optimization
Simulation
Titulação: Bacharel em Engenharia de Produção
Banca examinadora:
Carla Taviane Lucke da Silva Ghidini
Priscila Cristina Berbert Rampazzo
Data de entrega do trabalho definitivo: 01-12-2023

UNIVERSIDADE ESTADUAL DE CAMPINAS

FCA

Comissão de Graduação do Curso de Engenharia de Produção

Dados do Aluno	
RA 247548	**Nome do(a) Aluno(a)** Vothan Salomão Dos Santos
Nível Graduação	**Curso** 102G - Engenharia de Produção
Habilitação / Enfase -	

Dados do Trabalho	
Data/Hora do Exame 01/12/2023 - 08:00	**Local** https://meet.google.com/cpb-drbd-msp
Título “Otimização da operação de abastecimento de linhas de produção e Processos Industriais através da Integração de Técnicas de Aprendizado por Reforço e Simulação: Um Estudo com o Proximal Policy Optimization (PPO) no Ambiente FlexSim ”	
Código – Nome Disciplina EU010A - Trabalho de Graduação	
Orientador Prof. Dr. Anibal Tavares de Azevedo	

A Comissão Examinadora foi assim constituída:

Presidente		Nota
Nome Prof. Dr. Anibal Tavares de Azevedo / FCA/ Unicamp Participação por videoconferência Assinatura: Anibal Tavares de Azevedo __	**Matrícula** 301335	Dez
Membros		
Nome Profa. Dra. Carla Taviane Lucke da Silva Ghidini / FCA/ Unicamp Participação por videoconferência Assinatura: Carla Taviane Lucke da Silva Guidini __	**Matrícula** 306394	Dez
Nome Profa. Dra. Priscila Cristina Berbert Rampazzo / FCA/ Unicamp Participação por videoconferência Assinatura: Priscila Cristina Berbert Rampazzo __	**Matrícula** 310418	Dez

Coordenador	
Nome Prof. Dr. Paulo Sergio de Arruda Ignácio Assinatura: _	**Matrícula** 305221

CÓDIGO DE AUTENTICIDADE

Verifique a autenticidade deste documento na página www.dac.unicamp.br
Código: ee25469fb16b95c8a8a5e74e8e775b58e51277c4

Dedico este trabalho aos meus pais e avós, cujo amor, sabedoria e apoio incondicional foram as luzes guias em minha jornada. A vocês, que me ensinaram o valor da dedicação e da resiliência, ofereço este fruto do meu esforço e aprendizado.
Com amor e gratidão,
Vothan Salomão.

Agradecimentos

À medida que encerro esta etapa da minha jornada acadêmica, meu coração transborda de gratidão. As palavras, embora poderosas, parecem pequenas diante da imensidão do agradecimento que desejo expressar.

Aos meus amados pais e avós, verdadeiros pilares de força e amor, agradeço por terem plantado em mim as sementes da determinação e do sonho. Vocês são as raízes que me mantêm firme e a copa que me oferece sombra e proteção.

Minha irmã, companheira nas aventuras da vida, sua presença tem sido um farol de alegria e companheirismo. Em cada etapa, seu apoio foi tão essencial quanto as estrelas são para o céu noturno.

Aos meus amigos e colegas, especialmente Beatriz Spaggiari, Lucas Matheus Rodrigues, Luis Gustavo Freitas, Matheus de Araujo Coelho, Flávio Oliveira de Brito e Michael Machado, vocês são mais do que companheiros de jornada. Juntos, provamos que a soma dos nossos esforços pode tecer o tecido do impossível. Cada dúvida que compartilhamos e cada solução que encontramos juntos, foram os blocos de construção da nossa incrível realização coletiva.

Aos mestres que guiaram meus passos nesta estrada do conhecimento, vocês foram mais do que educadores, foram faróis de sabedoria em um mar de incertezas. A paciência e a dedicação de vocês não apenas iluminaram meu caminho, mas também me ensinaram a valorizar cada curva da aprendizagem.

Este trabalho não é apenas um reflexo do meu esforço, mas também um mosaico formado por cada um de vocês, cujas cores e texturas enriqueceram minha vida e esta jornada. A todos vocês, meu sincero obrigado.

Resumo

O projeto de Trabalho de Conclusão de Curso (TCC) tem como objetivo explorar a aplicação do algoritmo de Aprendizado por Reforço Proximal Policy Optimization (PPO) para otimizar o desempenho de Veículos Autônomos Guiados (AGVs, do inglês Automated Guided Vehicles) em ambientes de simulação utilizando o software Flexsim. Com o decorrer do tempo, tornou-se claro que os AGVs desempenham um papel fundamental na logística como uma tecnologia altamente valorizada em centros de distribuição, fábricas e armazéns. Desse modo, o seu papel é crucial na promoção da eficiência logística e na prevenção de acidentes na indústria. O emprego do aprendizado por reforço nos AGVs tem potencial para trazer múltiplos benefícios, tais como aprimoramento da eficiência operacional e diminuição dos gastos relacionados à produção nesses ambientes industriais. Para o aumento dessa eficiência, foi realizado um estudo utilizando o algoritmo PPO de aprendizado por reforço, para treinar nosso modelo, fornecendo ao agente, punições e recompensas ao longo do aprendizado, que corroborem para a sua eficiência. O treinamento aconteceu utilizando Python integrado com o software FlexSim, onde serviu de ambiente de treinamento para o nosso algoritmo. Uma das metas fundamentais deste projeto, consistiu em oferecer um método de otimização-simulação, empregando simulação multi-agentes e aprendizado por reforço. Além disso, realizou-se uma análise acerca do desempenho do uso de métodos de aprendizagem por reforço que podem ser expandida para diversos outros contextos industriais nos quais sejam necessária a coordenação otimizada dos AGVs. Observou-se que os resultados gerados, impactaram positivamente os principais indicadores, como lead time e número de movimentações de paletesm assim havendo uma aplicabilidade tangível em cenários do cotidiano, com o intuito de otimizar as operações em espaços industriais e minimizar o investimento logístico utilizando essa nova tecnologia.

Palavras-chave: Aprendizado por Reforço, Proximal Policy Optimization, AGVs, Flexsim, Otimização, Simulação, Automação Industrial.

Abstract

The final paper aims to explore the application of the Proximal Policy Optimization (PPO) Reinforcement Learning algorithm to optimize the performance of Automated Guided Vehicles (AGVs) in simulation environments using Flexsim software.

Over time, it has become clear that AGVs play a fundamental role in logistics as a highly valued technology in distribution centers, factories, and warehouses. Thus, their role is crucial in promoting logistical efficiency and preventing accidents in the industry. The use of reinforcement learning in AGVs has the potential to bring multiple benefits, such as improving operational efficiency and reducing production-related expenses in these industrial environments. To increase this efficiency, we will conduct a study using the PPO reinforcement learning algorithm to train our model, providing the agent with punishments and rewards throughout the learning process, which corroborate its efficiency. The training will take place using Python integrated with FlexSim software, which will serve as the training environment for our algorithm.

One of the fundamental goals of this project is to offer an optimization-simulation method, employing multi-agent simulation and reinforcement learning. In addition, an analysis was made about the performance of using reinforcement learning methods that can be expanded to various other industrial contexts where optimized coordination of AGVs is necessary. We hope that the generated results will have tangible applicability in everyday scenarios, with the aim of optimizing operations in industrial spaces and minimizing the logistical investment of those using this new technology.

Keywords: Reinforcement Learning, PPO, AGVs, Flexsim, Optimization, Simulation, Industrial Automation.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

$V^*(s)$	Valor ótimo do estado s.
π	Política de decisão.
G_t	Retorno total a partir do tempo t.
S_t	Estado no tempo t.
a	Ação tomada pelo agente.
s'	Estado sucessor.
$T(s, a, s')$	Função de transição do estado.
$R(s, a, s')$	Recompensa recebida após transição.
γ	Fator de desconto para o retorno futuro.
θ	Parâmetros da rede neural.
$J(\theta)$	Função objetivo para a política π.
∇_θ	Gradiente em relação aos parâmetros θ.
$\pi_\theta(a_t \mid s_t)$	Probabilidade de escolher a ação a no estado s segundo π.
α	Taxa de aprendizado no algoritmo.

Sumário

Capítulo 1

Introdução

A integração de veículos Autônomos Guiados (AGVs) no contexto industrial, desempenha um papel muito importante não só para aumentar ainda mais a produtividade, como também a eficiência na Indústria 4.0. (ZHONG et al., 2017), um respeitado estudioso na área, defende que a Indústria 4.0 pode oferecer múltiplas vantagens, contendo maior flexibilidade no processo fabril, capacidade para produção em grande escala com personalização e garantia reforçada da excelência dos produtos. Além do mais, proporciona uma produção expandida de empregos, um elemento vital para a implementação de tecnologias de ponta, como os AGVs, no setor industrial.

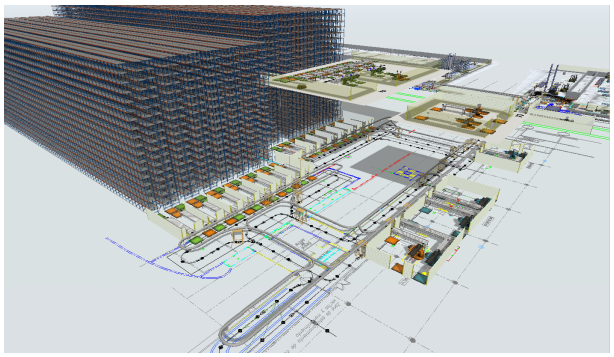
Contudo, as estratégias avançadas de controle são indispensáveis para garantir que os AGVs alcancem todo o seu potencial máximo enquanto se adaptam eficientemente a diversos cenários complexos. Dentro dessa perspectiva, o Aprendizado por Reforço (AR) desponta como uma estratégia promissora. O algoritmo de aprendizado por reforço que será utilizado é o Proximal Policy Optimization (PPO), o qual foi introduzido por pesquisadores da OpenAI. Ele é considerado uma melhoria sobre métodos mais antigos de aprendizado por reforço, como o Actor-Critic e o Trust Region Policy Optimization (TRPO). O PPO é projetado para ser mais simples de implementar e ajustar, enquanto mantém um bom desempenho, eficiência e robustez. Um ponto forte do PPO é sua aplicabilidade para enfrentar problemas complexos e melhorar a eficiência dos AGVs em ambientes virtuais(SCHULMAN; WOLSKI et al., 2017).

Este projeto de TCC visa explorar a aplicação do algoritmo PPO para otimizar o desempenho dos AGVs, utilizando o Software Flexsim como ambiente de treinamento. Através da modelagem do ambiente, implementação do algoritmo, treinamento dos AGVs e análise dos

resultados, buscamos contribuir para o avanço do conhecimento na interseção da automação industrial e do Aprendizado por Reforço.

(WANG et al., 2016) destacam a importância de sistemas auto-organizáveis e multiagentes com feedback baseado em big data e coordenação para a criação de fábricas inteligentes na Indústria 4.0. Desenvolvendo estratégias de controle para AGVs, essa abordagem se torna especialmente importante. A importância do big data e da inteligência artificial na manufatura automatizada também é salientada por (TAO et al., 2018). Apresentam uma visão importante sobre como integrar com sucesso as tecnologias avançadas dos AGVs em ambientes industriais. Com o intuito de otimizar as tarefas realizadas pelos AGVs, será conduzida uma avaliação minuciosa dos obstáculos e vantagens associados à introdução da Realidade Aumentada nas operações industriais. É esperado que os resultados obtidos, forneçam novos conhecimentos importantes e tragam contribuições práticas relevantes para a área da indústria e automação. Começar por esta fase significa embarcar em uma jornada de pesquisa e exploração em que todo o potencial sinérgico entre inteligência artificial e automação industrial se desdobra, abrindo caminho para futuras oportunidades interessantes na indústria.

Figura 1.1: AGVs em ambiente de fábrica dentro da simulação



Fonte: O autor

1.1 Abordagem do Problema

Impulsionada pelos avanços tecnológicos, a automação industrial assume um papel fundamental na procura constante de eficiência e competitividade nas fábricas modernas. Os Veículos Autônomos Guiados (AGVs) têm se tornado ativos valiosíssimos nesse contexto, pois oferecem um transporte eficiente e versátil de materiais em ambientes fabris complexos (ZHONG

et al., 2017). No entanto, a otimização das operações desses AGVs em ambientes de fábrica é um desafio significativo que requer abordagens avançadas.

Em um ambiente de fábrica, os AGVs frequentemente enfrentam uma série de desafios operacionais intrincados. Estes incluem a dinâmica do ambiente, como a presença de trabalhadores, maquinaria em movimento e estoques variados. Além disso, existem restrições temporais rigorosas, como prazos de produção e entregas just-in-time, que demandam alta eficiência e precisão nas operações de transporte (STOCK; SELIGER, 2016).

Para lidar com a complexidade dos cenários fabris, os AGVs precisam ter habilidades como tomada de decisão em tempo real e adaptação rápida diante das ocorrências não previstas. Além disso, é comum que os sistemas tradicionais de AGVs operem por meio de algoritmos pré-determinados e essa abordagem pode não oferecer flexibilidade o bastante para enfrentar as variações típicas do ambiente industrial(VAN BRUSSEL et al., 1998).

Portanto, o problema fundamental que este projeto de TCC busca abordar é a otimização das operações de AGVs em um ambiente de fábrica. Como podemos capacitar esses veículos autônomos a tomar decisões inteligentes e adaptativas em tempo real, levando em consideração a dinâmica do ambiente e as restrições operacionais?

1.2 Abordagem Proposta

Considera-se que a utilização da técnica de Proximal Policy Optimization (PPO) no campo de estudo conhecido como Aprendizado por Reforço seja uma estratégia promissora para enfrentar essa dificuldade. Por meio do uso inteligente das informações obtidas pelas experiências dos AGVs e pelo feedback das recompensas, o AR possibilita um aprendizado e flexibilidade contínua diante das constantes mudanças na fábrica.

Essa pesquisa busca examinar a utilização do PPO com o intuito de aprimorar as capacidades dos AGVs no contexto de uma fábrica simulada, contribuindo para melhorias nas operações automáticas fabris. Por meio dessa iniciativa, almejamos disponibilizar informações relevantes à indústria acerca da otimização do transporte de materiais em ambientes desafiadores. Com isso pretendemos impulsionar tanto a produtividades como também o investimento tecnológico entre as fábricas modernas.

Capítulo 2

Fundamentação Teórica

Antes de nos aprofundarmos na fundamentação teórica do aprendizado por reforço, é importante entender alguns conceitos-chave:

- **Agente:** Conforme definido por Sutton e Barto (2018), um agente é uma entidade, que pode ser um software ou uma combinação de software e hardware, responsável por tomar decisões em um ambiente. Ele interage com o ambiente, executando ações e recebendo recompensas correspondentes, sendo a entidade central que aprende no Reinforcement Learning (SUTTON; BARTO, 2018).

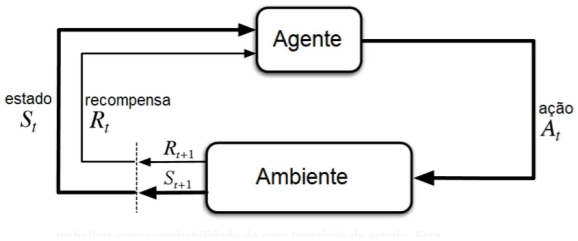
- **Ambiente:** O ambiente, como descrito por Mnih et al. (2015), é a materialização ou simulação do problema a ser resolvido. Pode ser um ambiente real ou virtual, onde o agente executa suas ações (MNIH et al., 2015).

- **Estado (s):** O estado representa a configuração atual do sistema, agente e ambiente em um determinado momento. Conforme o agente executa ações, o ambiente responde com um novo estado e uma recompensa correspondente, como ilustrado em trabalhos de Pan e Yang (2010) sobre aprendizado de transferência (PAN, S. J.; YANG, 2010).

- **Política (π):** A política é a estratégia que o agente utiliza para decidir suas próximas ações com base no estado atual. Essa política é continuamente atualizada para alcançar um comportamento ideal, conforme discutido por Kaelbling et al. (1996) (KAELBLING; LITTMAN; MOORE, 1996).

- **Recompensa (r):** A recompensa orienta as ações do agente. Uma ação desejável em um determinado estado resulta em uma recompensa positiva, enquanto uma ação indesejá-

Figura 2.1: Estrutura de um MDP



Fonte: O autor

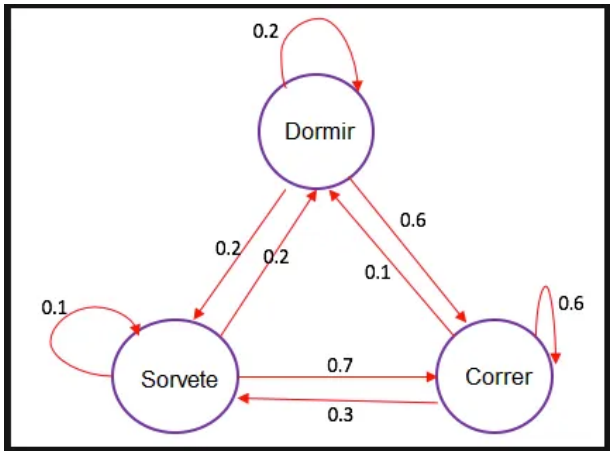
vel pode resultar em uma recompensa negativa ou nula, como explicado por Russell e Norvig (2016) (RUSSELL; NORVIG, 2016).

O aprendizado por reforço, objeto de estudo intenso na área da inteligência artificial sob os olhares atentos dos pesquisadores Sutton e Barto (2018), se dedica à compreensão do processo pelo qual um agente consegue adquirir conhecimento ao interagir com seu ambiente, buscando otimizar as possíveis recompenas acumuladas nesse contexto. Não há uma pessoa designada como supervisora do agente, responsável por indicar as medidas corretas para cada situação; no entanto, o desempenho do agente se manifesta através de sinais que representam recompensas. A finalidade reside em obter conhecimento acerca de como definir e aplicar corretamente um modelo político que seja capaz de associar cada situação ou circunstância encontrada no ambiente às melhores decisões possíveis. Essa associação visa alcançar e otimizar os retornos futuros aguardados por meio da soma calculada e ponderação adequada das eventuais recompensas subsequentes (SUTTON; BARTO, 2018). O foco dos próximos capítulos estará na exploração da construção do algoritmo, a qual é fundamentada neste capítulo.

2.1 Processo de Decisão de Markov(MDP)

Para modelar ambientes estocásticos e totalmente observáveis, nos quais agentes tomam decisões sequenciais, o conceito de Processo de Decisão de Markov (MDP) é fundamental. Um MDP, conforme descrito por Whittle e Puterman (WHITTLE; PUTERMAN, 1994), é composto por um conjunto de estados S, um conjunto de ações A, uma função de transição $T(s, a, s') = P(s'|s, a)$ que define a probabilidade de transitar para o estado s' após executar a ação a no estado s, e uma função de recompensa $R(s, a, s')$ que define a recompensa imediata recebida pelo agente após essa transição.

Figura 2.2: Probabilidade de Transição



Fonte:Fukushima, 2020

Puterman (PUTERMAN, 1995) enfatiza a importância de entender o comportamento desses sistemas de forma probabilística, trabalhando com a probabilidade de uma transição de estado. Essa probabilidade é função do estado e ação imediatamente anteriores, sem considerar todo o histórico anterior.

Papadimitriou e Tsitsiklis (PAPADIMITRIOU; TSITSIKLIS, 1987) discutem a complexidade inerente aos MDPs, destacando que a análise e solução desses problemas requerem uma compreensão profunda das dinâmicas estocásticas e das estratégias de otimização.

Considere um exemplo ilustrativo fornecido na figura 2.2 para entender as probabilidades de transição em um Processo de Decisão de Markov (MDP). Imagine um indivíduo hipotético que só pense em três coisas: dormir, correr ou tomar sorvete. Se ele estiver correndo, há 60% de chance de continuar correndo, 30% de chance de ir tomar sorvete, ou 10% de chance de ir dormir. Para cada um dos estados que esse indivíduo consegue almejar e atingir, existem chances correspondentes de migração para outro estado dentro desse grupo. Essa é uma ilustração das probabilidades de transição (PUTERMAN, 1995).

$$p(s', r|s, a) = P[S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a] \tag{2.1}$$

Um MDP também possui um fator de desconto $\gamma \in [0, 1]$ que determina a importância relativa das recompensas futuras em relação às presentes (BELLMAN, 1957).

A política ótima do agente é aquela que maximiza o valor esperado de cada estado, definido como:

$$V^*(s) = \max_{\pi} \mathbb{E}_{\pi}[G_t|S_t = s] \tag{2.2}$$

onde π é uma política, G_t é o retorno obtido a partir do tempo t, e S_t é o estado no tempo t. O valor esperado de cada estado pode ser obtido pela equação de Bellman:

$$V^*(s) = \max_{a} \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*(s')] \tag{2.3}$$

Essa equação expressa o princípio da otimalidade: o valor ótimo de um estado é igual ao valor máximo da soma da recompensa imediata com o valor ótimo do próximo estado. A equação de Bellman pode ser resolvida por algoritmos iterativos, como o algoritmo de iteração de valor ou o algoritmo de iteração de política (TIOMKIN; TISHBY, 2017).

2.2 Policy Gradient

Uma abordagem eficaz para encontrar diretamente a política ótima de um agente, sem a necessidade de uma função de valor auxiliar, é o uso de algoritmos de gradiente de política (Policy Gradient). Estes algoritmos, que empregam redes neurais para representar e atualizar a política do agente, baseiam-se no gradiente da função objetivo em relação aos parâmetros da rede (SUTTON; MCALLESTER et al., 2000).

A função objetivo em algoritmos de gradiente de política é o retorno esperado da política π_θ, onde θ representa os parâmetros da rede neural:

$$J(\theta) = \mathbb{E}_{\pi_\theta}[G_t] \tag{2.4}$$

O gradiente da função objetivo é expresso como:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t) G_t] \tag{2.5}$$

Aqui, $\pi_\theta(a_t|s_t)$ é a probabilidade de escolher a ação a_t no estado s_t conforme a política π_θ. Este gradiente aponta para a direção de maior aumento do retorno esperado e pode ser estimado por meio da amostragem de episódios de interação do agente com o ambiente. A atualização dos parâmetros da rede neural é realizada utilizando algoritmos de otimização, como o gradiente descendente estocástico (SGD), Momentum ou Adam (KINGMA; BA, 2014).

Um exemplo notável de algoritmo de gradiente de política é o REINFORCE, que opera da seguinte maneira (WILLIAMS, 1992):

Algorithm 1 Algoritmo REINFORCE

Require: Uma política $\pi(a|s,\theta)$ parametrizada por θ
Ensure: Melhor política $\pi^*(a|s)$
1: Inicialize os parâmetros da política θ
2: **for** cada episódio **do**
3: Gere um episódio seguindo a política $\pi(a|s,\theta)$: $s_0, a_0, r_1, ..., s_{T-1}, a_{T-1}, r_T$
4: **for** $t = 0$ até $T-1$ **do**
5: $G \leftarrow \sum_{k=t+1}^{T} \gamma^{k-t-1} r_k$ ▷ Calcula o retorno descontado
6: $\theta \leftarrow \theta + \alpha\gamma^t G \nabla_\theta \log \pi(a_t|s_t,\theta)$ ▷ Atualiza os parâmetros da política
7: **end for**

Um desafio com os algoritmos de gradiente de política é a alta variância nas estimativas do gradiente, o que pode resultar em aprendizado instável e lento (SUTTON; MCALLESTER et al., 2000).

2.3 Policy Gradient Baseline

Uma estratégia eficaz para reduzir a variância em algoritmos de gradiente de política é a utilização de uma função de valor como linha de base. Esta função serve como um ponto de referência para avaliar a eficácia de uma ação em um estado específico. A função de valor pode ser estimada por outra rede neural, que recebe o estado como entrada e retorna o valor esperado como saída. A função objetivo dos algoritmos de gradiente de política com linha de base é expressa como (KONDA; TSITSIKLIS, 2000):

$$J(\theta) = \mathbb{E}_{\pi_\theta}[\nabla_\theta \log \pi_\theta(a_t|s_t)(G_t - V(s_t))] \tag{2.6}$$

onde $V(s_t)$ é a estimativa da função de valor no estado s_t. A subtração da linha de base não muda o valor esperado do gradiente, mas ajuda a reduzir sua variância.

Um exemplo de algoritmo de gradiente de política com linha de base é o Actor-Critic, que opera da seguinte maneira (MNIH et al., 2016):

Um desafio nos algoritmos de gradiente de política é lidar com ambientes contínuos, onde as ações são valores reais e não discretos. Nestes casos, a política deve ser uma função que gera uma distribuição de probabilidade sobre as ações possíveis, como uma distribuição gaussiana

Algorithm 2 Algoritmo Actor-Critic

1: Inicializar os parâmetros θ da rede neural da política (ator) e os parâmetros w da rede neural da função de valor (crítico) aleatoriamente.
2: **repeat**
3: Inicializar o estado s.
4: **repeat**
5: Escolher uma ação a seguindo a política π_θ.
6: Executar a ação a e observar o próximo estado s' e a recompensa r.
7: Calcular o erro TD: $\delta = r + \gamma V(s') - V(s)$.
8: Atualizar os parâmetros w na direção do gradiente do erro quadrático médio: $w \leftarrow w + \beta\delta\nabla_w V(s)$, onde β é a taxa de aprendizado do crítico.
9: Atualizar os parâmetros θ na direção do gradiente da função objetivo: $\theta \leftarrow \theta + \alpha\nabla_\theta \log \pi_\theta(a|s)\delta$, onde α é a taxa de aprendizado do ator.
10: Atualizar o estado: $s \leftarrow s'$.
11: **until** fim do episódio
12: **until** convergência =0

ou beta. A rede neural da política deve produzir os parâmetros dessa distribuição, como média e desvio padrão, e a ação é amostrada dessa distribuição (SILVER et al., 2014).

2.4 Trust Tegion Policy Optmization(TRPO)

Um desafio comum em algoritmos de gradiente de política para ambientes contínuos é a ineficiência na exploração, frequentemente devido à dependência de ruído adicionado à política para gerar ações variadas. Uma abordagem para aprimorar a exploração é adotar uma política estocástica, que cria uma distribuição de probabilidade sobre as ações possíveis, e aprender os parâmetros dessa distribuição através de máxima verossimilhança (SCHULMAN; LEVINE et al., 2015).

Um exemplo de algoritmo de gradiente de política para ambientes contínuos com política estocástica é o Trust Region Policy Optimization (TRPO), que opera da seguinte forma (SCHULMAN; LEVINE et al., 2015):

Um desafio dos algoritmos de gradiente de política com política estocástica é a alta sensibilidade aos hiperparâmetros, como o tamanho do passo ou o grau de proximidade entre as políticas. Uma estratégia para tornar o algoritmo mais robusto e eficiente é a utilização de uma função de perda modificada que limita mudanças significativas na política do agente, mantendo-a próxima da política anterior (SCHULMAN; WOLSKI et al., 2017).

Algorithm 3 Algoritmo TRPO

1: Inicializar os parâmetros θ da rede neural da política estocástica (ator) e os parâmetros w da rede neural da função de valor (crítico) aleatoriamente.
2: **repeat**
3: Gerar um conjunto de trajetórias seguindo a política π_θ e armazenar os estados, as ações, as recompensas e as probabilidades.
4: Para cada trajetória:
   1. Calcular o retorno G_t e a vantagem A_t de cada passo usando o estimador de vantagem generalizada (GAE).
   2. Calcular a razão entre as probabilidades da ação a_t sob as políticas atual e anterior: $r_t(\theta) = \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$.
5: Atualizar os parâmetros θ na direção do gradiente da função objetivo: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$, onde α é a taxa de aprendizado e $J(\theta) = \mathbb{E}_{\pi_\theta}[r_t(\theta)A_t]$, sendo ϵ um hiperparâmetro que controla o tamanho da região de confiança.
6: Atualizar os parâmetros w na direção do gradiente do erro quadrático médio: $w \leftarrow w + \beta(G_t - V(s_t))\nabla_w V(s_t)$, onde β é a taxa de aprendizado do crítico.
7: **until** convergência

2.5 Algoritmo de Otimização de Política Proximal (PPO)

O Proximal Policy Optimization (PPO) é um método de Aprendizado por Reforço que utiliza gradiente de política, introduzido pela equipe OpenAI em 2017. Este método rapidamente ganhou popularidade devido ao seu equilíbrio entre facilidade de implementação, complexidade de amostra e facilidade de ajuste (SCHULMAN; WOLSKI et al., 2017).

O PPO busca calcular uma atualização a cada etapa que minimiza a função de valor, mantendo um desvio relativamente pequeno da política anterior. Isso é feito para assegurar um treinamento mais estável e prevenir que o agente siga um caminho irrecuperável de ações sem sentido (SCHULMAN; WOLSKI et al., 2017).

Na implementação do PPO, duas políticas são mantidas: a política atual π_θ(at|st), que se busca aperfeiçoar, e a política π_{θ_k}(at|st), utilizada no último treinamento. A proporção $r_t(\theta) = \frac{\pi_\theta(\text{at}|\text{st})}{\pi_{\theta_k}(\text{at}|\text{st})}$ mede a diferença entre as duas políticas. A função objetivo é então definida como:

$$L_{\text{CLIP}}(\theta) = \mathbb{E}_t[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$$

A função CLIP limita a probabilidade quando há variações significativas, utilizando o hiperparâmetro ϵ, com valores comuns de 0,1 ou 0,2. Se a taxa de proporção entre a nova política

e a antiga estiver fora do intervalo $(1-\epsilon, 1+\epsilon)$, a função objetivo é limitada (SCHULMAN; WOLSKI et al., 2017).

Além disso, o PPO pode ser combinado com métodos Actor-Critic para melhorar o desempenho. No Actor-Critic, o Ator decide a ação com base na política atual, enquanto o Crítico avalia a ação usando uma função de valor. O Ator atualiza os parâmetros da política com base nas informações fornecidas pelo Crítico, permitindo um aprendizado mais eficiente (SCHULMAN; WOLSKI et al., 2017).

Algorithm 4 Algoritmo PPO

1: Inicializar os parâmetros θ da rede neural da política estocástica (ator) e os parâmetros w da rede neural da função de valor (crítico) aleatoriamente.
2: **repeat**
3: Gerar um conjunto de trajetórias seguindo a política π_θ e armazenar os estados, as ações, as recompensas e as probabilidades.
4: Para cada trajetória:
   1. Calcular o retorno G_t e a vantagem A_t de cada passo usando o estimador de vantagem generalizada (GAE).
   2. Calcular a razão entre as probabilidades da ação a_t sob as políticas atual e anterior: $r_t(\theta) = \frac{\pi_{\theta_{\text{old}}}(a_t|s_t)}{\pi_\theta(a_t|s_t)}$.
5: Atualizar os parâmetros θ na direção do gradiente da função objetivo: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$, onde α é a taxa de aprendizado e $J(\theta) = \mathbb{E}_{\pi_\theta}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t)]$, sendo ϵ um hiperparâmetro que controla o tamanho da região de confiança.
6: Atualizar os parâmetros w na direção do gradiente do erro quadrático médio: $w \leftarrow w + \beta(G_t - V(s_t))\nabla_w V(s_t)$, onde β é a taxa de aprendizado do crítico.
7: **until** convergência

Existem seis situações possíveis que podem ocorrer durante o treinamento segundo(PAN, S. J.; YANG, 2010):

1. A proporção está dentro do intervalo e a vantagem é positiva: Nesse caso, a ação é melhor do que a média de todas as ações naquele estado. Portanto, a política atual é incentivada a aumentar a probabilidade de tomar essa ação naquele estado.

2. A proporção está dentro do intervalo e a vantagem é negativa: Nesse caso, a ação é pior do que a média de todas as ações naquele estado. Portanto, a política atual é desencorajada de tomar essa ação naquele estado.

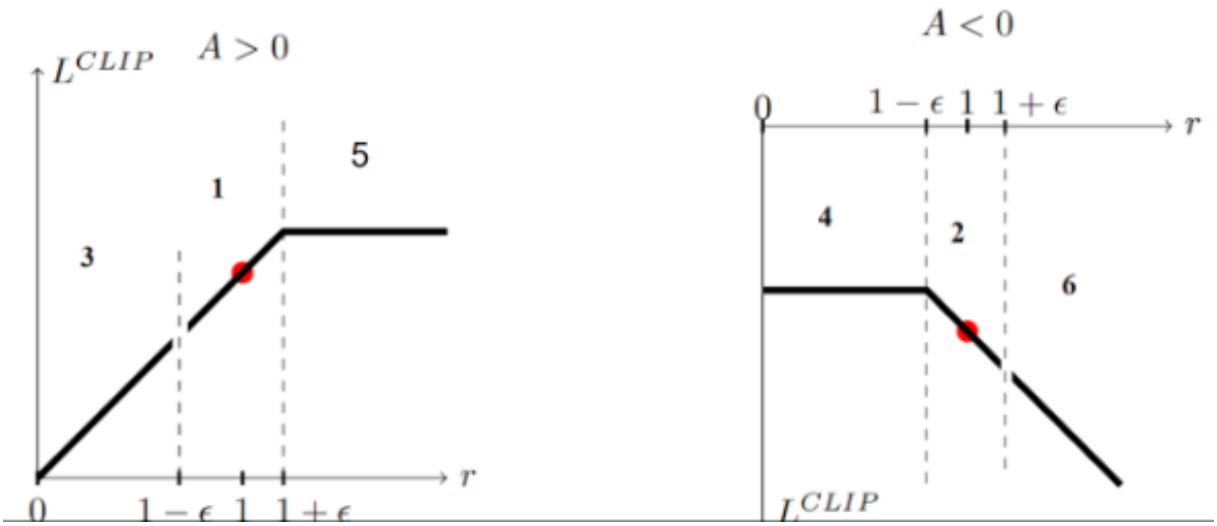
3. A proporção está abaixo do intervalo e a vantagem é positiva: Nesse caso, a probabilidade de tomar essa ação naquele estado é muito menor do que com a política antiga.

$pt(\theta) > 0$	At	Return Value of min	Objective is Clipped	Sign of Objective	Gradient
$pt(\theta)\epsilon[1-\epsilon, 1+\epsilon]$	+	$pt(\theta)At$	no	+	∨
$pt(\theta)\epsilon[1-\epsilon, 1+\epsilon]$	-	$pt(\theta)At$	no	-	∨
$pt(\theta) < 1-\epsilon$	+	$pt(\theta)At$	no	+	∨
$pt(\theta) < 1-\epsilon$	-	$(1-\epsilon)At$	yes	-	0
$pt(\theta) > 1+\epsilon$	+	$(1+\epsilon)At$	yes	+	0
$pt(\theta) > 1+\epsilon$	-	$pt(\epsilon)At$	no	-	∨

Fonte: Bick, D. (2021)

Tabela 2.1: Tabela resumindo o comportamento da função objetivo L_{CLIP} do PPO para todos os casos não triviais, onde ambos $pt(\theta)$ e At são diferentes de zero. A primeira coluna indica o valor da proporção de probabilidade $pt(\theta)$, enquanto a segunda coluna indica se a estimativa de vantagem At é positiva (+) ou negativa (-) para um dado exemplo de treinamento (indexado pelo subscrito t) retirado de um minibatch de exemplos de treinamento. A terceira coluna indica a saída de L_{CLIP}, ou seja, o valor de retorno do operador mínimo de L_{CLIP} para o exemplo de minibatch indexado pelo subscrito t. A quarta coluna indica se este termo, ou seja, a saída de L_{CLIP}, é um termo recortado (sim) ou não (não). A quinta coluna indica se o sinal do valor retornado por L_{CLIP} é positivo (+) ou negativo (-). A última coluna indica se o gradiente resultante da retropropagação de L_{CLIP} visa maximizar o valor retornado por L_{CLIP} (∨) ou se apenas o gradiente zero trivial (0) resulta.

Figura 2.3: Gráfico da atualização da política conforme a L CLIP



Fonte: Bick, D. (2021)

No entanto, como a vantagem é positiva, a política atual é incentivada a aumentar a probabilidade de tomar essa ação naquele estado.

4. A proporção está abaixo do intervalo e a vantagem é negativa: Nesse caso, a probabilidade de tomar essa ação naquele estado é muito menor do que com a política antiga. No entanto, como a vantagem é negativa, a política atual não é incentivada a diminuir ainda mais a probabilidade de tomar essa ação naquele estado.

5. A proporção está acima do intervalo e a vantagem é positiva: Nesse caso, a probabilidade de tomar essa ação naquele estado na política atual é muito maior do que na política anterior. No entanto, como a vantagem é positiva, a política atual não é incentivada a ser muito gananciosa e aumentar ainda mais a probabilidade de tomar essa ação naquele estado.

6. A proporção está acima do intervalo e a vantagem é negativa: Nesse caso, a probabilidade de tomar essa ação naquele estado na política atual é muito maior do que na política anterior. No entanto, como a vantagem é negativa, a política atual é incentivada a diminuir a probabilidade de tomar essa ação naquele estado.

Em resumo, a política é atualizada apenas se a proporção estiver dentro do intervalo $[1 - \epsilon, 1 + \epsilon]$, ou se a proporção estiver fora do intervalo, mas a vantagem leva a se aproximar do intervalo. Isso restringe o intervalo em que a política atual pode variar em relação à antiga, removendo o incentivo para que a razão de probabilidade se mova para fora do intervalo, pois o clipe tem o efeito de gradiente. Se a proporção for $> 1 + \epsilon$ ou $< 1 - \epsilon$, o gradiente será igual a 0.

Capítulo 3

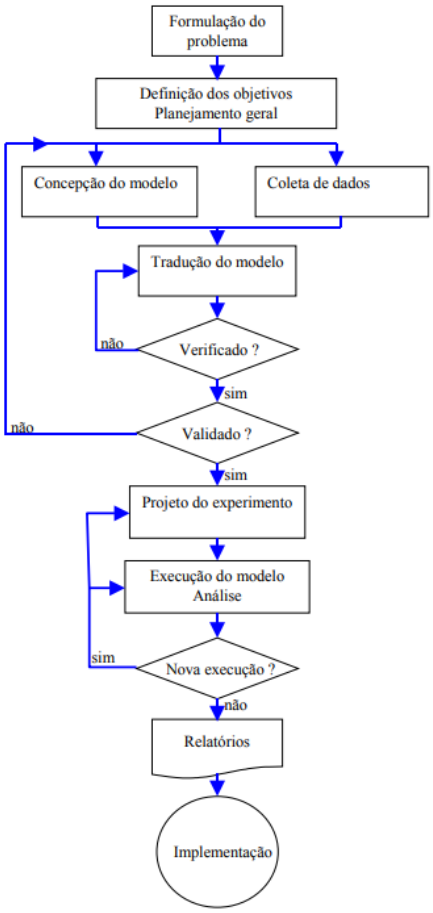
Introdução à Simulação

A simulação é uma ferramenta essencial na Engenharia de Produção e áreas relacionadas, desempenhando um papel crucial na modelagem, análise e otimização de sistemas complexos (BANKS et al., 2007). Como técnica, a simulação permite a criação de modelos computacionais de sistemas reais para compreender seu comportamento, fazer previsões e tomar decisões informadas (LAW, 2007). Sua relevância na Engenharia de Produção é destacada pela complexidade, multifacetamento e variabilidade dos sistemas encontrados nessa área (ROBINSON, 2004).

As aplicações da simulação são vastas, abrangendo manufatura, logística, cadeia de suprimentos, saúde, serviços e transporte (NEGAHBAN; SMITH, J. S., 2014). Essa técnica é inestimável para modelar sistemas complexos e testar diferentes estratégias operacionais (JUN; JACOBSON; SWISHER, 2006). Além disso, a simulação é fundamental para apoiar a tomada de decisões, permitindo a avaliação do impacto de diversas escolhas em termos de custos, qualidade, eficiência e desempenho (BANKS et al., 2007).

Este trabalho de conclusão de curso (TCC) explorará aplicações significativas da simulação na Engenharia de Produção, incluindo a otimização de layouts de fábricas, dimensionamento de capacidades de produção, melhoria da logística e aprimoramento de processos de atendimento ao cliente (LAW, 2007; NEGAHBAN; SMITH, J. S., 2014). Também serão examinadas as tecnologias e ferramentas para implementação de simulações, bem como as melhores práticas para assegurar resultados precisos e confiáveis (ROBINSON, 2004).

Figura 3.1: Fluxograma das etapas para uma simulação



Fonte: O autor

Com o avanço da inteligência artificial e do aprendizado de máquina, o aprendizado por reforço emerge como uma técnica promissora, envolvendo agentes autônomos que aprendem a otimizar seu desempenho em ambientes complexos através da interação e experiência (KAELBLING; LITTMAN; MOORE, 1996). O aprendizado por reforço tem ganhado destaque na Engenharia de Produção, possibilitando a otimização autônoma de processos de fabricação e logística (SUTTON; BARTO, 2018).

A medida que nossa sociedade enfrenta desafios cada vez mais complexos, a simulação e o aprendizado por reforço emergem como ferramentas indispensáveis para a inovação e o aprimoramento contínuo. Neste TCC, pretendemos demonstrar como essas abordagens podem capacitar os engenheiros de produção a enfrentar esses desafios de frente, tomando decisões

informadas com base em modelos computacionais precisos e representativos da realidade e permitindo que sistemas autônomos aprendam a operar de maneira eficaz.

Ao explorar o vasto mundo da simulação, suas aplicações práticas e as possibilidades oferecidas pelo aprendizado por reforço, esperamos fornecer uma visão abrangente e inspiradora deste campo dinâmico e em constante evolução. Através deste estudo, os leitores poderão apreciar o potencial da simulação e do aprendizado por reforço como ferramentas vitais para aprimorar processos, sistemas e operações, impulsionando a inovação e a excelência em engenharia.

3.1 O que é Simulação?

A simulação é um processo poderoso e essencial na engenharia de produção, que permite a análise e experimentação de sistemas complexos. De acordo com Seila [1995], um sistema é definido como um conjunto de componentes ou entidades que interagem para atingir um objetivo. Um modelo, por outro lado, é uma representação dos componentes mais importantes do sistema e a forma como eles interagem.

A simulação é particularmente útil em sistemas de grande complexidade, que não admitem falhas, de difícil entendimento, que ainda não existem, onde alterações acarretam grandes custos, e onde há variabilidade e interdependência. A simulação permite a representação da dinâmica de um objeto ou sistema para análise de alguma atividade deste, considerada muito grande e/ou complexa.

O "Simulation Modeling Handbook" define a simulação como o processo de criação e experimentação com um modelo matemático informatizado de um sistema físico. A simulação tradicional é usada para analisar sistemas e tomar decisões operacionais ou políticas de recursos.

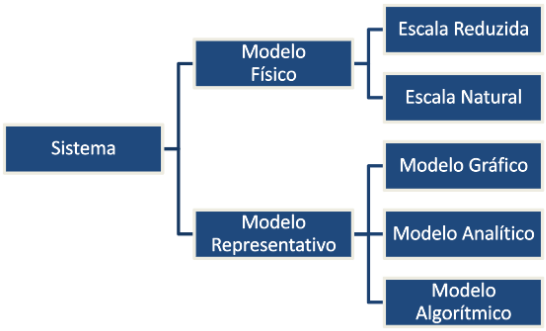
A simulação é classificada quanto à continuidade em Simulação Contínua e Simulação Discreta. Quanto ao tipo de variáveis, a simulação pode ser Determinística ou Estocástica.

Ela pode ser classificada de acordo com a continuidade e o tipo de variáveis envolvidas:

Quanto à continuidade:

Simulação Contínua: Neste tipo de simulação, o tempo é um fator muito importante. São feitos cálculos sobre as mudanças nas variáveis de cada estado. A Simulação Contínua depende

Figura 3.2: Modelo na Simulação



Fonte: O autor

de variáveis que assumem valores contínuos, isto é, em um domínio de valores contínuos tais como o conjunto de números reais. Simulação Discreta: A Simulação Discreta modela a operação de um sistema como uma sequência de eventos discretos no tempo. Cada evento ocorre em um determinado instante e marca uma mudança de estado no sistema. A Simulação Discreta depende de variáveis que assumem valores discretos, isto é, em um domínio de valores finitos ou enumeráveis tais como o conjunto de números inteiros. Quanto ao tipo de variáveis:

Simulação Determinística: Neste tipo de simulação, dada uma certa entrada, ela produzirá sempre a mesma saída, com a máquina responsável sempre passando pela mesma sequência de estados6. O método determinístico projeta o fluxo de caixa, geralmente, com base nos dados históricos da empresa e em previsões econômicas. Simulação Estocástica: A Simulação Estocástica envolve a geração de números aleatórios (pseudo-aleatórios) com o objetivo de explorar o espaço de incerteza ou campo de possibilidades de um dado fenômeno físico ou qualquer outro tipo de variável de estudo cujo comportamento possa ser quantificado matematicamente. Essas classificações ajudam a entender melhor o comportamento do sistema que está sendo simulado e escolher a abordagem mais adequada para cada situação. Em resumo, a simulação é empregada quando o valor gasto com seu desenvolvimento é muito menor do que o desenvolvimento de um protótipo ou testes reais. Portanto, a simulação desempenha um papel crucial na engenharia de produção, permitindo a análise eficiente e econômica de sistemas complexos.

Figura 3.3: Fluxo da representação de um modelo real na simulação

Fonte: FlexSim Brasil

3.2 Importância das Réplcas na Simulação

As Réplicas em simulações são execuções repetidas de um modelo sob as mesmas condições iniciais. Elas são cruciais para avaliar a variabilidade e a confiabilidade dos resultados, especialmente em simulações estocásticas que incluem componentes de aprendizado de máquina (SMITH, J. A.; OTHER, 2015).

3.3 Cálculo de Réplicas

O número de réplicas necessárias n para alcançar um intervalo de confiança desejado para uma estimativa com um nível de confiança de $1 - \alpha$ e uma variância estimada S^2 é dado pela fórmula:

$$n \geq \left(\frac{z_{\alpha/2} \cdot S}{\varepsilon} \right)^2 \tag{3.1}$$

onde $z_{\alpha/2}$ é o valor crítico da distribuição normal padrão para um nível de confiança $1 - \alpha$, S é o desvio padrão das métricas de desempenho de interesse, e ε é a largura máxima desejada do intervalo de confiança (JOHNSON, 2018).

3.4 Número de réplicas no projteo

Para garantir a confiabilidade estatística dos resultados obtidos na simulação dos AGVs, adotou-se a prática de realizar múltiplas réplicas. No contexto deste trabalho, uma 'réplica' é definida como uma execução independente da simulação completa, começando com as mesmas condições iniciais e parâmetros de configuração. Cada réplica pode ser vista como uma amostra independente do espaço de todos os possíveis resultados do modelo de simulação.

A escolha de realizar 5 réplicas foi baseada em considerações estatísticas que equilibram a necessidade de uma análise de dados robusta com a viabilidade em termos de recursos computacionais e tempo. Este número de réplicas permite a avaliação da variabilidade dos resultados e a construção de intervalos de confiança com um nível aceitável de precisão, sem impor uma carga excessiva nos recursos de computação.

A aplicação de réplicas em simulações de AGVs com aprendizado de máquina é fundamental para assegurar a validade dos resultados obtidos. O cálculo do número de réplicas é essencial para a construção de intervalos de confiança precisos e para a verificação da consistência das estratégias de aprendizado (LOPEZ; KHAN, 2019).

Capítulo 4

Simulação do Ambiente Industrial

Caracterizada pela habilidade em responder às variações na demanda e no mercado, a agilidade na cadeia de suprimentos desempenha um papel crucial(JHARKHARIA; SHANKAR, 2007; AZIZ; SAMAN, 2018). A agilidade implica na redução do tempo de ciclo, a flexibilização da produção e uma sincronização eficiente dos fluxos de materiais e informações (JHARKHARIA; SHANKAR, 2007). Vantagens competitivas consideráveis podem ser conquistadas pelas empresas ao optarem por uma cadeia de suprimentos ágil. Isso inclui satisfazer melhor os clientes, aumentar a participação no mercado e melhorar a rentabilidade.(SAJAD; LONG, 2018).

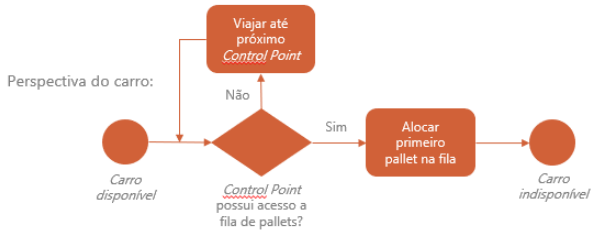
4.1 Funcionamento da Eletromonovia:

A eletromonovia que constituem de uma via única onde os AGVs circulam, contribui para a agilidade da sua cadeia de suprimentos, pois permite que os paletes sejam transportados de forma rápida e contínua, reduzindo o tempo de espera e de entrega dos produtos. A eletromonovia é um sistema de transporte automatizado que usa trilhos elétricos para movimentar paletes entre diferentes pontos de uma fábrica ou de um armazém. A eletromonovia é composta

Figura 4.1: Fluxograma da movimentação

Fonte: O autor

Figura 4.2: Fluxograma do carro



Fonte: O autor

Figura 4.3: processo de movimentação dos Paletes na Eletromonovia

Fonte: O autor

por veículos motorizados que se deslocam sobre os trilhos, seguindo um circuito pré-definido. Os veículos podem carregar e descarregar os paletes de forma autônoma, usando sensores e dispositivos de elevação. A eletromonovia é controlada por um software que coordena o fluxo dos veículos e dos paletes, garantindo a segurança e a eficiência do sistema. A eletromonovia interliga toda a fábrica, desde a chegada da matéria-prima até a saída do produto final. A eletromonovia realiza o reabastecimento para todos os setores, através da movimentação de 300 paletes por hora até as 50 mil posições dos transelevadores, que são equipamentos que armazenam e recuperam os paletes de forma vertical.

Reduzindo o tempo necessário para os processos de produção e distribuição, a eletrmonovia eleva tanto a produtivi Desde o momento em que os materiais são adquiridos até chegar às mãos do cliente final, o ciclo temporal abrange todas as etapas necessárias na cadeia de abastecimento. A redução do tempo do ciclo tem como consequência direta uma maior velocidade no

atendimento dos pedidos, possibilitando também um aumento na capacidade para lidar com as flutuações na demanda e resultando numa menor quantidade necessária no estoque.. Segundo a empresa, a eletromonovia reduziu o tempo de ciclo em 50%, o que representa uma economia de $ 15 milhões por ano .

A eletromonovia também flexibiliza a produção e a distribuição dos produtos para os seus principais destinos, dentro da fábrica, como os armazéns verticas(AV) que é um tipo de armazém que utiliza o espaço vertical para armazenar os produtos, usando estruturas como estantes, prateleiras ou transelevadores. O armazém vertical permite otimizar o uso do espaço, reduzir o tempo de movimentação e aumentar a capacidade de armazenagem, em torno da eletromonovia, há dois armazéns verticais, como mostrado na Figura4.4:

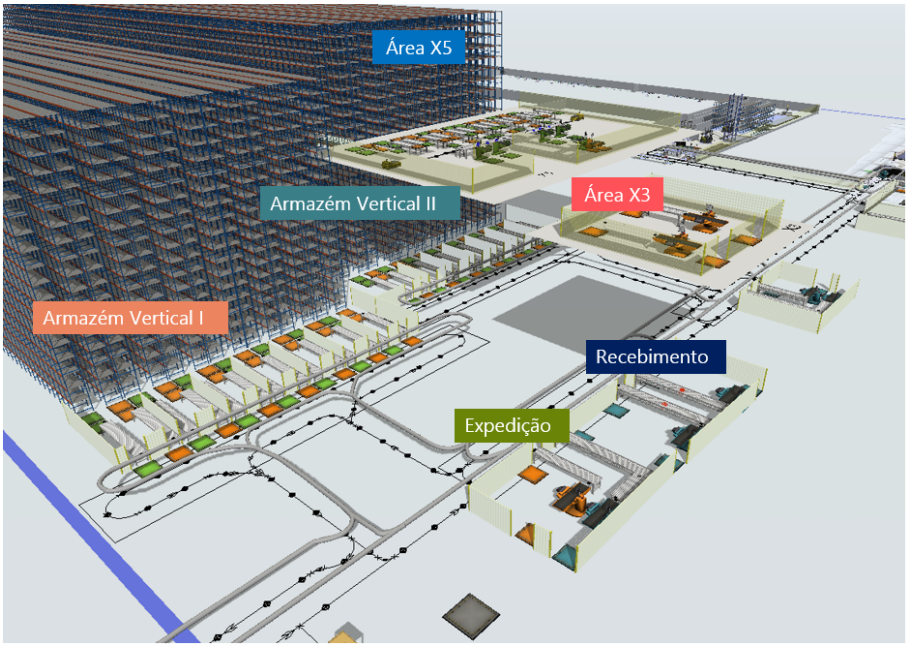
A eletromonovia permite que os paletes sejam transportados de forma contínua, sem interrupções ou paradas. Tornando o fluxo do processo fluído e sem dificuldades, sendo a quantidade, variedade e qualidade dos seus produtos para suprir as exigências tanto dos clientes quanto do mercado. É necessário empregar recursos modulares, versáteis e integrados para alcançar a flexibilidade desejada. Esses recursos devem poder ser facilmente reconfiguráveis ou redirecionáveis. Adotar estratégias flexíveis na produção e distribuição pode gerar grandes oportunidades para as empresas se destacarem no mercado. Elas terão a capacidade não apenas para oferecer uma variedade maior e mais criativa de produtos, mas também poderão fornecer atendimento

A eletromonovia sincroniza os fluxos de materiais e de informações da cadeia de suprimentos, pois utiliza um sistema de identificação e de rastreamento dos paletes, que evita erros e perdas. A sincronização é a coordenação dos processos e das atividades da cadeia de suprimentos, de forma a evitar desperdícios, atrasos ou excessos. A sincronização envolve a troca de dados e de informações entre os elos da cadeia, de forma a garantir a visibilidade e a transparência dos fluxos. Uma cadeia de suprimentos sincronizada pode oferecer vantagens competitivas para as empresas, como uma maior precisão dos dados, uma maior qualidade dos produtos e uma maior confiabilidade dos serviços.

4.2 Funcionamento das Mesas

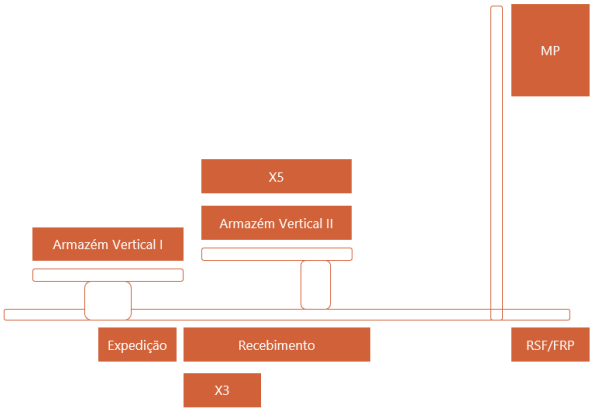
As mesas na fábrica, chamadas de mesas de entrada, saída e reversíveis, são equipamentos que permitem a transferência dos paletes entre os diferentes pontos do sistema de transporte

Figura 4.4: Ilustração interna da fábrica na simulação



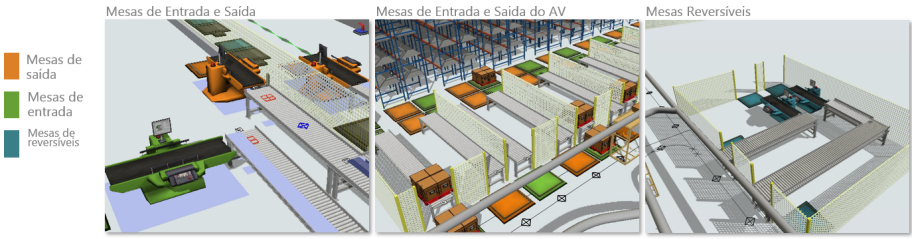
Fonte: O autor

Figura 4.5: Representação da planta da fábrica



Fonte: O autor

Figura 4.6: Diferentes tipos de Mesas



Fonte: O autor

automatizado da eletromonovia. Elas são compostas por esteiras rolantes que se movem em diferentes direções, de acordo com a necessidade de cada operação. Elas podem ser classificadas em:

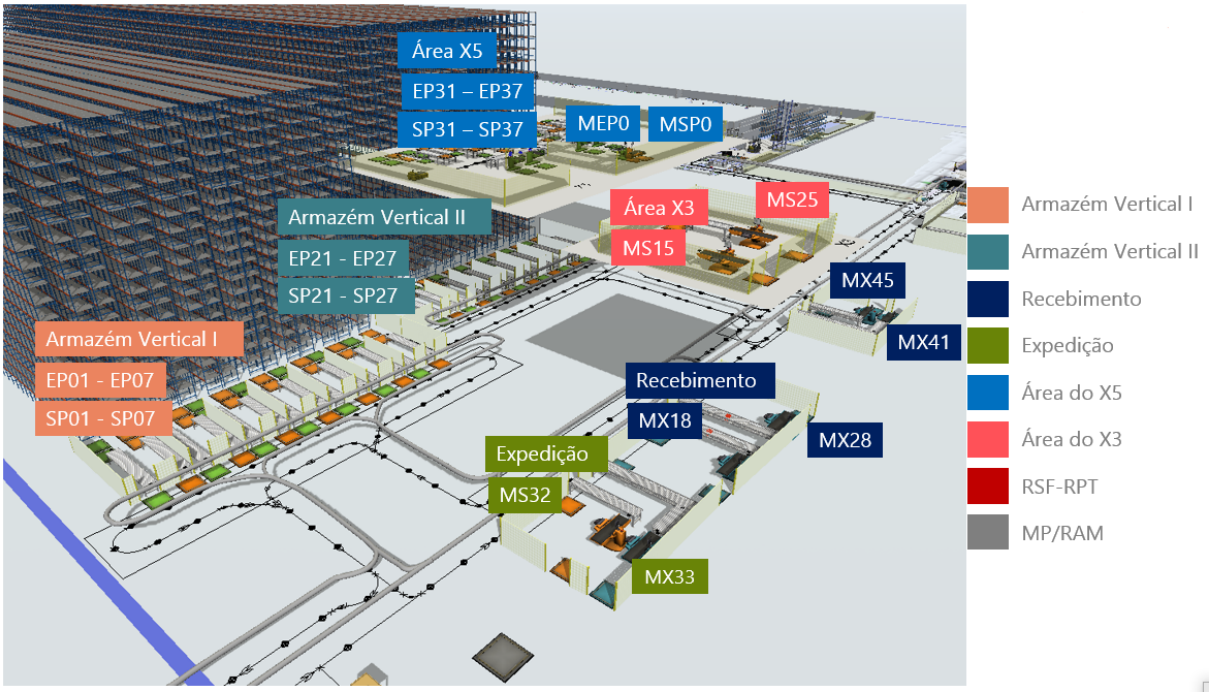
- Mesas de entrada: são as mesas que recebem os paletes que chegam dos pontos de origem, como as docas, as áreas de produção ou as áreas de armazenagem. Elas são responsáveis por alinhar os paletes com os trilhos elétricos e acionar os veículos motorizados que vão transportá-los até os pontos de destino.
- Mesas de saída: são as mesas que entregam os paletes que chegam dos pontos de destino, como as áreas de expedição, as áreas de armazenagem ou as áreas de produção. Elas são responsáveis por desalinhar os paletes dos trilhos elétricos e liberar os veículos motorizados que vão retornar aos pontos de origem.
- Mesas reversíveis: são as mesas que podem funcionar tanto como mesas de entrada quanto como mesas de saída, dependendo da demanda do sistema. Elas são responsáveis por inverter a direção dos paletes, de forma a facilitar a movimentação dos veículos motorizados e otimizar o fluxo dos paletes.

4.3 Funcionamento das Fábricas

Na fábrica, há quatro unidades produtivas, elas são baseadas em um sistema integrado de supply chain que utiliza a eletromonovia como meio de transporte sustentável e eficiente. A eletromonovia é uma ferrovia elétrica que liga as fábricas com os centros de distribuição e os armazéns verticais.

O processo começa com o abastecimento das fábricas com os insumos e matérias-primas necessários para a produção dos produtos, como, cremes, perfumes, etc. Esses insumos são

Figura 4.7: Localização das Mesas ao longo da Eletromonovia



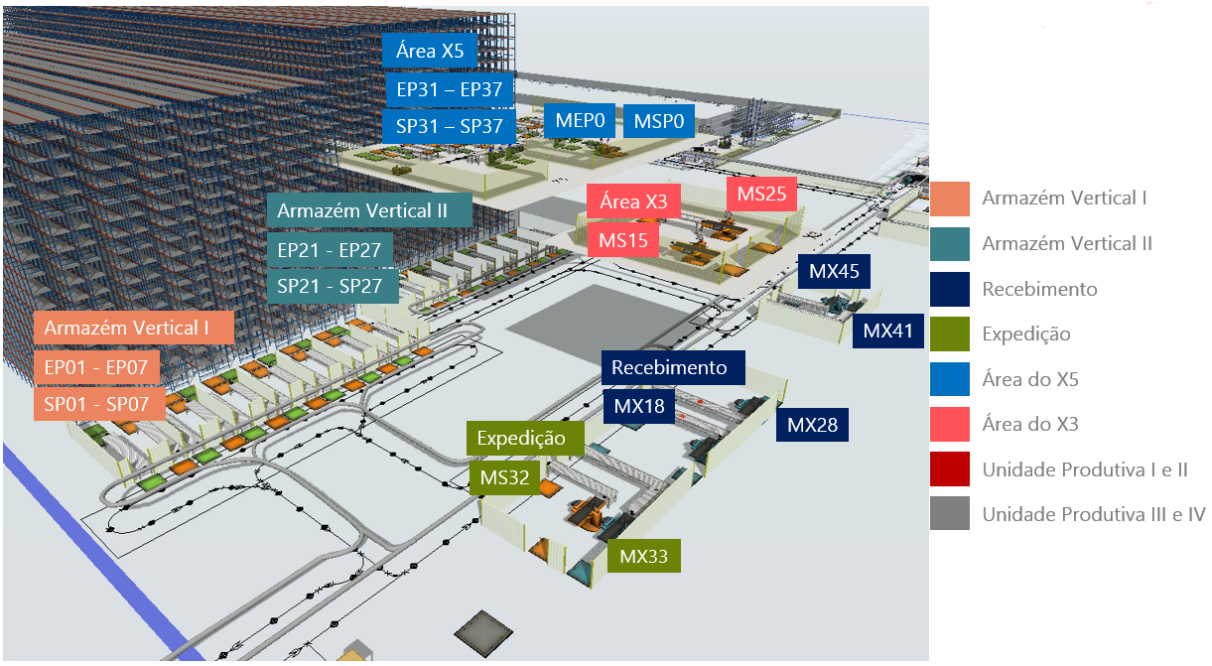
Fonte: O autor

provenientes de fornecedores externos ou de outras unidades, e são transportados pela eletromonovia até as fábricas. Cada fábrica é responsável por uma linha de produtos específica, de acordo com a sua capacidade e especialização.

Após a produção, os produtos acabados são enviados pela eletromonovia para três destinos possíveis: O recebimento, a expedição ou um dos armazéns verticais. A expedição é o local onde os produtos são separados e embalados para serem enviados aos clientes finais, seja por meio de transportadoras, lojas físicas ou consultoras. Os armazéns verticais são locais onde os produtos são armazenados por meio de esteiras e transelevadores em estruturas automatizadas e inteligentes, que otimizam o espaço e facilitam o controle de estoque. No total, são dois armazéns verticais, um para cada linha de produtos.

Dessa forma, as unidades produtivas funcionam de forma integrada e sustentável, utilizando a eletromonovia como um diferencial competitivo e ambiental.

Figura 4.8: Localização das Unidades Produtivas



Fonte: O autor

Capítulo 5

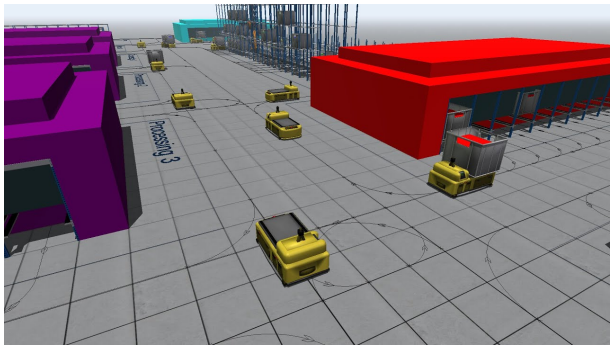
Método de Simulação-Otimização da Operação dos AGVs via Aprendizado por Reforço

A simulação é uma técnica que permite modelar e analisar sistemas complexos, reproduzindo o seu comportamento e avaliando o seu desempenho sob diferentes cenários e condições. A simulação pode ser usada para diversos fins, como planejamento, otimização, treinamento, pesquisa e entretenimento. Uma das áreas de aplicação da simulação é a engenharia de produção, que estuda e projeta sistemas produtivos, envolvendo pessoas, materiais, equipamentos, informações e energia.

Um exemplo de sistema produtivo que pode ser simulado é o uso de veículos guiados automaticamente (AGVs) para transportar materiais entre diferentes pontos de controle em uma fábrica. Os AGVs são veículos autônomos que seguem rotas pré-definidas ou dinâmicas, usando sensores e sistemas de navegação. Os AGVs podem aumentar a eficiência, a segurança e a flexibilidade do transporte de materiais, mas também apresentam desafios como o planejamento de rotas, a coordenação entre veículos, a alocação de recursos e a gestão de conflitos.

Uma forma de abordar esses desafios é usar técnicas de inteligência artificial, como o aprendizado por reforço, para treinar algoritmos que possam controlar os AGVs de forma ótima. O aprendizado por reforço é uma técnica especial no mundo da inteligência artificial, em que um agente aprende a atingir uma meta em um ambiente incerto e potencialmente complexo.

Figura 5.1: Ambiente de simulação com AGVs



Fonte: O autor

O agente aprende por meio de tentativa e erro, recebendo recompensas ou penalidades pelas ações que executa. Seu objetivo é maximizar a recompensa total.

Um dos algoritmos de aprendizado por reforço mais populares e eficazes é o PPO (Proximal Policy Optimization), que busca maximizar a recompensa esperada de uma política. Uma política é uma função que mapeia as observações do agente para as ações que ele deve tomar. O PPO é um método que atualiza a política de forma gradual e segura, evitando grandes mudanças que possam prejudicar o desempenho do agente. O PPO também é capaz de lidar com ambientes contínuos e discretos, bem como com recompensas atrasadas ou esparsas.

5.1 Automated guided vehicle(AGVs) e Aprendizado por Reforço

Para usar o PPO para controlar os AGVs em um ambiente simulado, é preciso definir os seguintes elementos:

- As observações: são as informações que o agente recebe sobre o estado do ambiente em cada momento da simulação. Por exemplo, as observações podem incluir a posição, a velocidade, o destino e a carga dos AGVs, bem como a localização e o tipo dos materiais nas estações.
- As ações: são as decisões que o agente toma para alterar o estado do ambiente em cada momento da simulação. Por exemplo, as ações podem incluir a escolha do próximo ponto de controle a ser visitado pelo AGV ou a modificação dos destinos dos AGVs.

- A recompensa: é o sinal que o agente recebe após cada ação, indicando o quão boa foi essa ação para atingir a meta. Por exemplo, a recompensa pode ser baseada no número de materiais entregues, no tempo de entrega ou no consumo de energia dos AGVs.

Uma ferramenta que permite usar um modelo FlexSim como um ambiente para treinar e avaliar algoritmos de aprendizado por reforço é o Reinforcement Learning Tool. Essa ferramenta fornece recursos necessários para um algoritmo de aprendizado por reforço se comunicar com o FlexSim usando sockets. Além disso, essa ferramenta disponibiliza scripts em Python que demonstram como se comunicar com o FlexSim usando o OpenAI Gym e o Stable-Baselines3, que são bibliotecas populares para implementar algoritmos de aprendizado por reforço.

Para usar essa ferramenta, é preciso configurar um ambiente de desenvolvimento Python e instalar as bibliotecas necessárias. Depois, é preciso preparar um modelo FlexSim com os elementos necessários para o aprendizado por reforço, como tabelas de parâmetros para as observações e as ações, triggers para enviar e receber dados do Python e uma função para calcular a recompensa. Em seguida, é preciso modificar os scripts em Python para se adaptarem ao modelo FlexSim e ao algoritmo PPO. Por fim, é possível executar os scripts para treinar, salvar e avaliar o modelo PPO usando a conexão direta com o ambiente FlexSim.

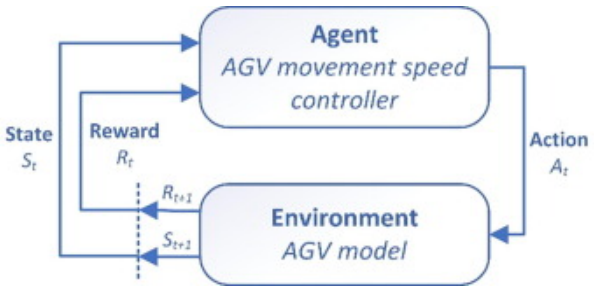


Figura 5.2: Fluxo de aprendizagem dos AGVs na Simulação

5.2 Treinamento do agente na simulação

Para o treinamento ds AGVs na simulação, utilizamos o framework de aprendizado por reforço da OpenAI Gym, ao conectarmos o FLexSim com o Pyhton, pudemos realizar o aprendizado do agente

5.2.1 Identificação dos Parâmetros de Observação

Os parâmetros de observação são cruciais para monitorar e otimizar o comportamento dos AGVs no sistema de eletromonovia. No modelo foram utilizados os seguintes parâmetros de observação por meio da criação de Tabelas de parâmetros no FlexSim

- **Último Control Point (LastCP)**: Representa a última posição conhecida do AGV.
- **Control Point Atual (CP)**: Indica a próxima posição prevista do AGV.
- **Destination**: Refere-se ao ponto final de entrega ou coleta do AGV.
- **Distância entre o AGV e o Control Point (Distance)**: Mede a distância física entre o AGV e o control point mais próximo.

Figura 5.3: Tabela Global com os Parâmetros do modelo

	agv	cp	lastCP	destination	Distance
CAR	/CAR	/CP_AV01_14	/CP_AV01_14	/FELE_MS32	48.26809
CAR_2	/CAR_2	/CP_AUX_1	/CP_AUX_1	/FELE_MX18	21.50399
CAR_3	/CAR_3	/CP_AUX_9	/CP_AUX_9	/FELE_MS52	45.47788
CAR_4	/CAR_4	/CP_AV02_11	/CP_AV02_11	/FELE_MS52	158.64165
CAR_5	/CAR_5	/CP_AUX_31	/CP_AUX_31	/FELE_SP02	77.18503
CAR_6	/CAR_6	/CP_WORKF_14	/CP_WORKF_14	/FELE_MS52	66.95592
CAR_7	/CAR_7	/CP_AUX_11	/CP_AUX_11	/FELE_MS32	38.84739
CAR_8	/CAR_8	/CP_WORKF_11	/CP_WORKF_11	/FELE_MS32	139.58937
CAR_9	/CAR_9	/CP_AUX_2	/CP_AUX_2	/FELE_MS32	17.68489
CAR_10	/CAR_10	/CP_WORKF_02	/CP_WORKF_02	/FELE_MS32	6.68229
CAR_11	/CAR_11	/CP_AUX_24	/CP_AUX_24	/FELE_SP24	140.76675
CAR_12	/CAR_12	/CP_AUX_7	/CP_AUX_7	/FELE_MS32	147.18501
CAR_13	/CAR_13	/CP_WORKF_13	/CP_WORKF_13	/FELE_MS62	108.35858
CAR_14	/CAR_14	/CP_WORKF_03	/CP_WORKF_03	/FELE_MS62	85.23682
CAR_15	/CAR_15	/CP_AV02_12	/CP_AV02_12	/FELE_SP24	156.29808
CAR_16	/CAR_16	/CP_AUX_8	/CP_AUX_8	/FELE_SP21	58.72473
CAR_17	/CAR_17	/CP_REC_03	/CP_REC_03	/FELE_SP26	90.45332
CAR_18	/CAR_18	/CP_AUX_28	/CP_AUX_28	/FELE_SP23	230.99594
CAR_19	/CAR_19	/CP_WORKF_02	/CP_WORKF_02	/FELE_SP01	6.68229
CAR_20	/CAR_20	/CP_AV01_04	/CP_AV01_04	/FELE_MS62	57.07042
CAR_21	/CAR_21	/CP_WORKF_03	/CP_WORKF_03	/FELE_MS62	85.23682
CAR_22	/CAR_22	/CP_WORKF_16	/CP_WORKF_16	/FELE_SP02	139.11467
CAR_23	/CAR_23	/CP_AUX_29	/CP_AUX_29	/FELE_SP22	136.20462
CAR_24	/CAR_24	/CP_WORKF_12	/CP_WORKF_12	/FELE_MS62	131.36304
CAR_25	/CAR_25	/CP_WORKF_14	/CP_WORKF_14	/FELE_MS62	66.95592
CAR_26	/CAR_26	/CP_FAB_05	/CP_FAB_05	/FELE_MX18	163.00854
CAR_27	/CAR_27	/CP_AUX_7	/CP_AUX_7	/FELE_MS150	147.18501
CAR_28	/CAR_28	/CP_WORKF_11	/CP_WORKF_11	/FELE_MX33	139.58937

Fonte: O autor

5.3 Ação do Agente

Para a ação do agente na mudança de rota dos AGVs, foi realizada na ferramenta de Reinforcement Learning do FLexSim, onde foram colocado os respectivos Inputs abordados na seção anterior.

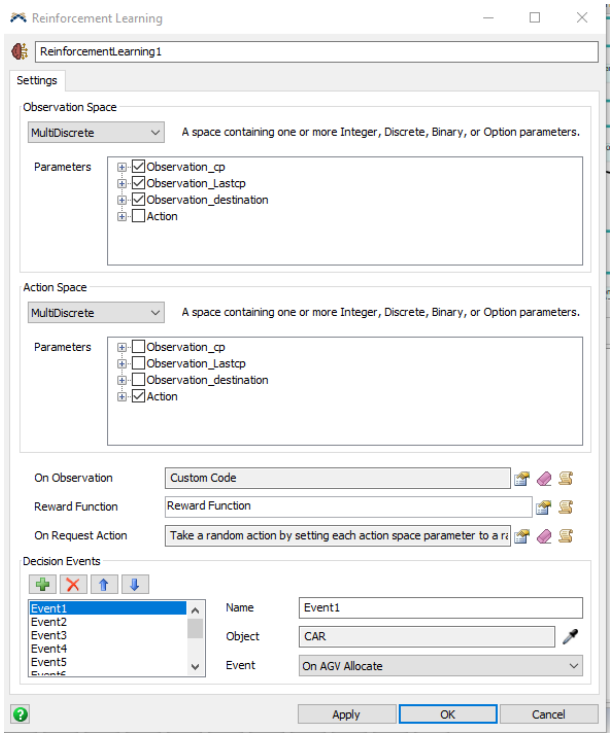


Figura 5.4: Ferramenta de Aprendizado por reforço FlexSim

5.3.1 Configuração no ProcessFlow

No ProcessFlow do FlexSim, foi configurado um fluxo de etapas para a troca automática de AGV que estejam sem itens sendo carregados, para poderem trocar e otimizar conforme as tomadas de decisão do agente, para as melhores decisões:

1. Detecção de Demanda de Item.

2. Troca de AGVs.

3. Atualização em Tempo Rea conforme os estados observadosl.

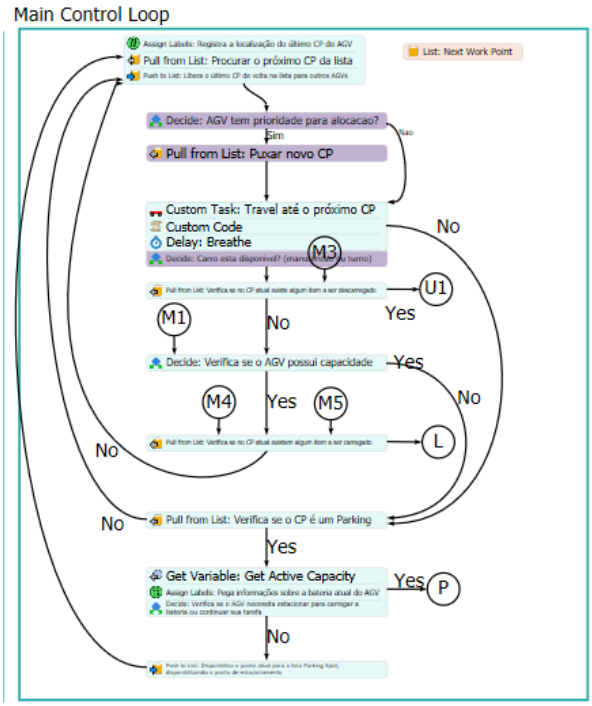


Figura 5.5: Fluxo de lógica de designação de tarefa para os AGVs na Eletromonovia

5.4 Modelo de Recompensa

A função de recompensa é crucial no aprendizado por reforço, pois guia o agente para alcançar o objetivo. A recompensa deve ser projetada de forma a incentivar comportamentos desejáveis e desencorajar comportamentos indesejáveis. Neste caso dos AGVs, os comportamentos de menor lead time e maior volume de itens levados ao destinos, serão tidos como comportamentos positivos e favoráveis, enquanto gargalos e filas, serão tidos como punição.

5.4.1 Código de Recompensa

O código utilizado para calcular a recompensa do agente é o seguinte:

```
double reward = Model.find("Tools/PerformanceMeasureTables/pm_Indicadores
    >variables/performanceMeasures/2/Value").value -
    Model.find("Tools/PerformanceMeasureTables/pm_Indicadores
    >variables/performanceMeasures/3/Value").value;

if (reward < 6) {
    reward = 1;
```

```
} else if (reward <= 7) {
    reward = 0.7;
} else if (reward <= 15) {
   reward = 0.3;
} else if (reward > 15) {
    reward = 0;
}
return reward;
```

5.5 Treinamento do Modelo

Inicie o processo de treinamento onde os AGVs (agentes) interagem com o ambiente FlexSim. Em cada episódio de treinamento, os AGVs devem realizar tarefas, como navegar até um destino ou transportar carga, baseando-se na política atual. Após cada episódio, o algoritmo PPO atualiza a política do agente com base nas recompensas coletadas, buscando melhorar o desempenho nas tarefas.

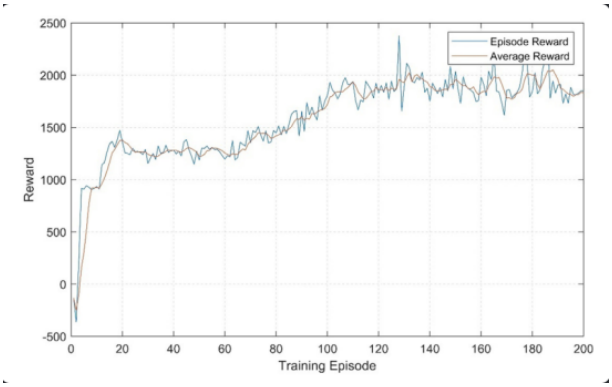
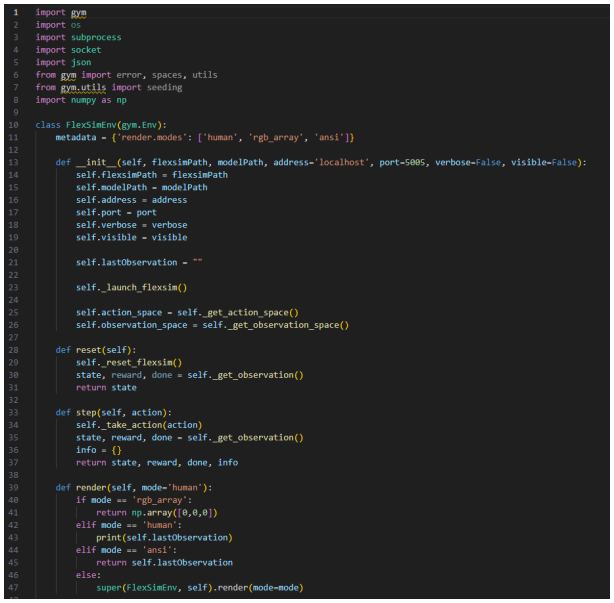


Figura 5.6: Gráfico de treinamento do Modelo

Figura 5.7: Recompensas coletadas ao longo da Simulação

5.5.1 Integração FlexSim-Python

A comunicação entre o FlexSim e o Python foi estabelecida através de sockets por meio do seguinte código:



```python
import gym
import os
import subprocess
import socket
import json
from gym import error, spaces, utils
from gym.utils import seeding
import numpy as np

class FlexSimEnv(gym.Env):
    metadata = {'render.modes': ['human', 'rgb_array', 'ansi']}

    def __init__(self, flexsimPath, modelPath, address='localhost', port=5005, verbose=False, visible=False):
        self.flexsimPath = flexsimPath
        self.modelPath = modelPath
        self.address = address
        self.port = port
        self.verbose = verbose
        self.visible = visible

        self.lastObservation = ""

        self._launch_flexsim()

        self.action_space = self._get_action_space()
        self.observation_space = self._get_observation_space()

    def reset(self):
        self._reset_flexsim()
        state, reward, done = self._get_observation()
        return state

    def step(self, action):
        self._take_action(action)
        state, reward, done = self._get_observation()
        info = {}
        return state, reward, done, info

    def render(self, mode='human'):
        if mode == 'rgb_array':
            return np.array([0,0,0])
        elif mode == 'human':
            print(self.lastObservation)
        elif mode == 'ansi':
            return self.lastObservation
        else:
            super(FlexSimEnv, self).render(mode=mode)
```

Figura 5.8: Código Python que permite conexão com o FlexSim

O script flexsimenv.py contém a definição de uma classe FlexSimEnv que podemos usar para comunicar de Python para FlexSim. Possui funções para iniciar o FlexSim com um modelo e comunicar-se com o modelo por meio de soquetes, a continuação do código se encontra nos Apendices.A.1

5.5.2 Algoritmo PPO e Stable Baselines 3

O modelo foi treinado usando o algoritmo Proximal Policy Optimization (PPO) com a biblioteca Stable Baselines 3.

5.5.3 Configuração do Algoritmo PPO:

Utilize o Stable Baselines para configurar o algoritmo PPO. Isso inclui colocar os parâmetros dos AGVs, taxa de aprendizado, número de passos de atualização O PPO é uma escolha popular para esse tipo de tarefa devido à sua eficiência e estabilidade em ambientes complexos.

5.5.4 Monitoramento e Ajustes:

Monitore o desempenho dos AGVs ao longo do treinamento no FLexSim. Isso inclui analisar métricas como número de tarefas concluídas, ocorrência de colisões, eficiência de rota, possíveis ocorrências de bugs durante a simulação Faça ajustes nos parâmetros do algoritmo PPO, na função de recompensa, ou no próprio ambiente de simulação, conforme necessário.

5.5.5 Validação e Testes:

Após um período de treinamento, valide a eficácia dos AGVs em diferentes cenários dentro do FlexSim para garantir que a aprendizagem seja generalizável e robusta. Realize testes em condições variáveis, como mudanças no layout ou na demanda de tarefas, para avaliar a adaptabilidade dos AGVs.

5.5.6 Reflexão sobre o treinamento:

Este processo é iterativo e pode exigir ajustes frequentes para otimizar o desempenho dos AGVs. A chave é a integração eficaz entre o FlexSim e o Stable Baselines, juntamente com uma função de recompensa bem projetada e um monitoramento contínuo do desempenho dos agentes.

Capítulo 6

Resultados e Discussões

6.1 Modelagem do Sistema Atual:

No sistema atual, os Veículos Guiados Automáticos (AGVs) operam dentro de um ambiente de produção complexo, onde a eficiência e a rapidez na movimentação de materiais são cruciais. Para compreender e otimizar o funcionamento deste sistema, utilizou-se o FlexSim, um software avançado de simulação 3D, que permite a modelagem detalhada de processos de produção.

A coleta de dados foi realizada simulando o ambiente operacional dos AGVs no FlexSim, considerando vários parâmetros como layout da planta, tipos de materiais movimentados (insumos, produtos acabados, etc.) e o fluxo de trabalho em diferentes horários do dia. Os dados coletados incluem o número de paletes movimentados por hora e o lead time médio, que representa o tempo desde a solicitação até a entrega do material. Esses dados foram concatenados nas tabelas 6.1 e 6.2, onde mostra-se os dados da seguinte forma. Na primeira coluna temos a origem e destino para cada um dos locais dentro da fábrica, na segunda coluna os dados coletados se referem ao número total de movimentações que aconteceram ao longo do ano dos seus respectivos locais de origem para o destino. Já a terceira coluna, a tabela nos dara a informação do lead time médio das movimentações de cada origem para cada destino dentro da simulação, assim havendo a comparação do número de movimentação e lead time médio antes e depois da utilizacão do aprendizado por reforço.

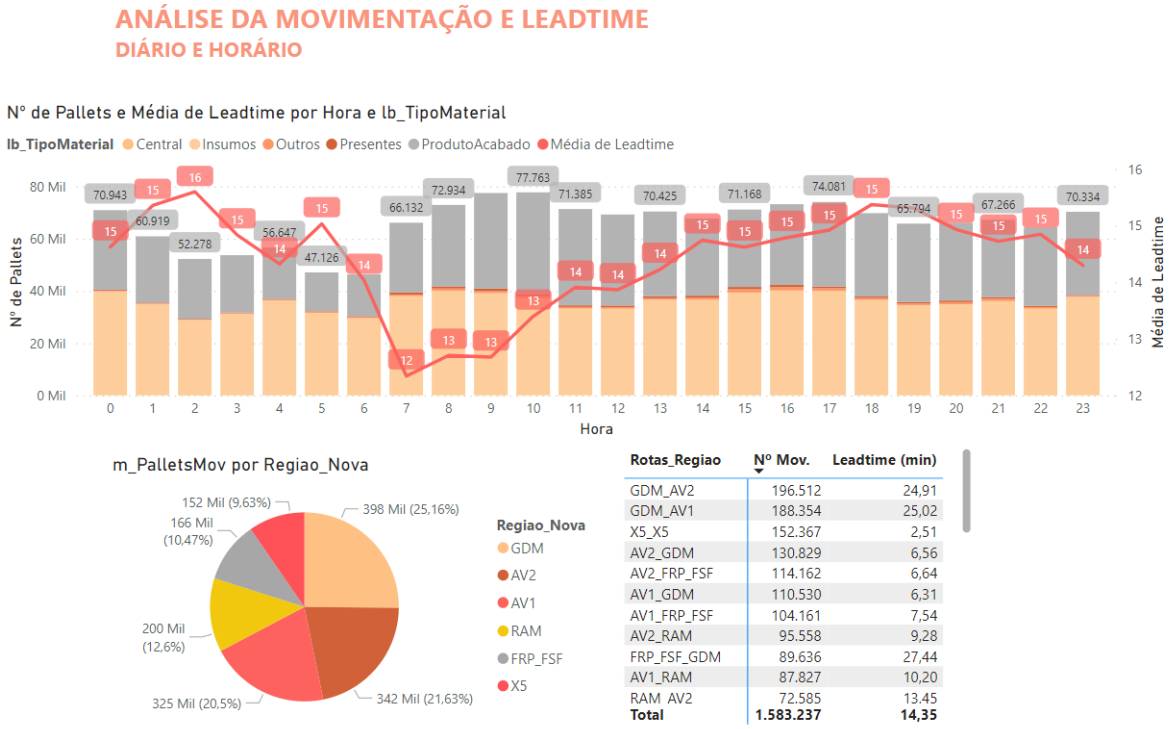


Figura 6.1: Análise do sistema atual real sem otimização

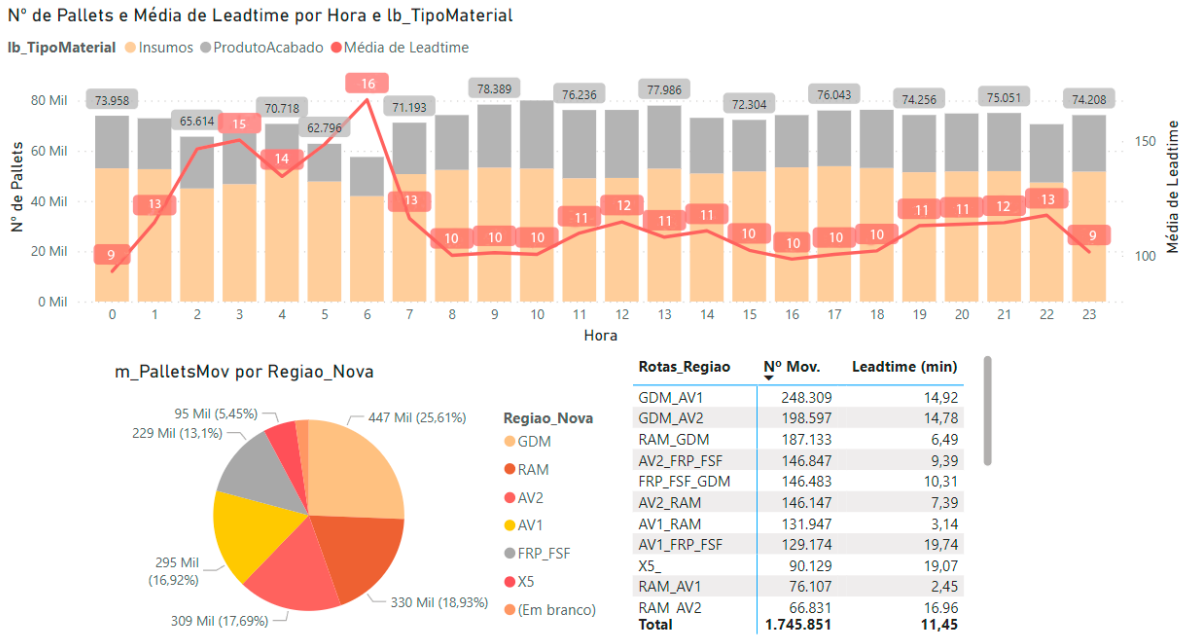


Figura 6.2: Análise do sistema pós otimização

Já os gráficos 6.1 e 6.2, nos mostram a análise da média do número de pallets movimentados e o respectivo lead time para cada hora do dia no cenário atual e pós utilização do aprendizado por reforço.

6.2 Análise dos Dados

Nos dados do estado atual, observamos que o número de paletes movimentados varia significativamente ao longo do dia, com picos às 2h, 6h. O lead time médio também oscila, com valores mais altos ocorrendo no início e no final do período analisado, e os menores valores ao redor das 15h.

No cenário otimizado, o número de paletes movimentados apresenta variações similares ao longo do dia, mas com um pico ligeiramente menor às 8h comparado ao estado atual. O lead time médio nesse cenário é consistentemente menor ao longo de todo o dia, com exceção de um aumento às 22h.

Ao analisar a movimentação de paletes por região, observamos uma distribuição mais uniforme no cenário otimizado, sugerindo uma possível melhoria na eficiência da distribuição dos AGVs. As rotas com maiores números de movimentações GDM → AV1 e GDM → AV2 apresentam lead times significativamente menores no cenário otimizado.

6.3 Discussão

A implementação de aprendizado por reforço contribuiu para a redução do lead time médio conforme a comparação da tabela das figuras 6.2 e 6.1 , o que é um indicativo de melhoria na eficiência operacional dos AGVs. Esta melhoria pode ser atribuída à capacidade do modelo de aprendizado por reforço de adaptar-se e otimizar continuamente as rotas e a alocação de recursos em tempo real.

- Aumento no Número Total de Movimentações: O cenário pós PPO mostra um aumento no número total de movimentações ao analisarmos a comparação realizada no gráfico 6.3, indicando um aumento de 10,27% no cenário pós PPO, assim havendo uma otimização na quantidade de viagens necessárias.

Tabela 6.1: Resumo das Rotas e Lead times antes da otimização

Rotas_Regiao	Nº Mov.	Lead time (min)
GDM→AV2	196512	24,91
GDM→AV1	188354	25,02
X5→X5	152367	2,51
AV2→GDM	130829	6,56
AV2→FRP_FSF	114162	6,64
AV1→GDM	110530	6,31
AV1→FRP_FSF	104161	7,54
AV2→RAM	95558	9,28
FRP_FSF→GDM	89636	27,44
AV1→RAM	87827	10,20
RAM→AV2	72585	13,45
RAM→AV1	68871	14,02
RAM→GDM	57686	13,33
FRP_FSF→AV1	38374	26,13
FRP_FSF→AV2	37129	25,49
AV1→X5	19392	10,90
GDM→X3	5493	11,31
GDM→RAM	4317	54,65
GDM→FRP_FSF	2787	47,81
AV1→AV1	1879	6,92
AV2→AV2	1525	5,95
GDM→GDM	891	40,07
AV1→AV2	818	8,32
FRP_FSF→RAM	569	27,77
AV2→AV1	414	7,25
RAM→FRP_FSF	413	11,42
X5→AV1	113	10,33
FRP_FSF→FRP_FSF	28	27,06
X5→GDM	6	9,00
RAM→RAM	6	14,75
X5→RAM	2	13,33
X5→FRP_FSF	2	8,13
X5→X3	2	12,13
FRP_FSF→X3	1	16,14
RAM→X3	1	13,29
Total de Movimentações	1583237	14,35min

Tabela 6.2: Resumo das Rotas e Lead times após otimização

Rotas_Regiao	Nº Mov.	Lead time (min)
GDM→ AV1	248309	14,92
GDM→AV2	198597	14,78
RAM→GDM	187133	6,49
AV2→FRP_FSF	146847	9,39
FRP→FSF_GDM	146483	10,31
AV2→RAM	146147	7,39
AV1→RAM	131947	3,14
AV1→FRP_FSF	129174	19,74
X5_	90129	19,07
RAM→AV1	76107	2,45
RAM→AV2	66831	16,96
FRP→FSF_AV1	40973	17,57
FRP→FSF_AV2	40868	16.67
_X5	40187	10,59
AV1→X5	18093	5,76
AV1→GDM	15919	6
AV2→GDM	15424	9,54
X5→GDM	4884	5,70
FRP→FSF_RAM	445	19,97
RAMAV2→_AV2	401	22,15
FRP_FSF	306	11,98
AV1AV2→_AV2	401	6,52
AV1	243	10,51
GDMAV2→_AV2	401	18,61
RAM	227	19,20
X5AV2→_AV2	401	10,58
AV1	105	13,50
RAMAV2→_AV2	401 0	23,40
RAM	44	19,1
GDMAV2→_AV2	401	11,86
FRP_FSF	25	13,60
GDMAV2→_AV2	401	16,25
X5	2	14,00
FRPAV2→_AV2	401	17,67
FSF_FRP_FSF	1	15,70
Total de Movimentações	1745851	11,45 min

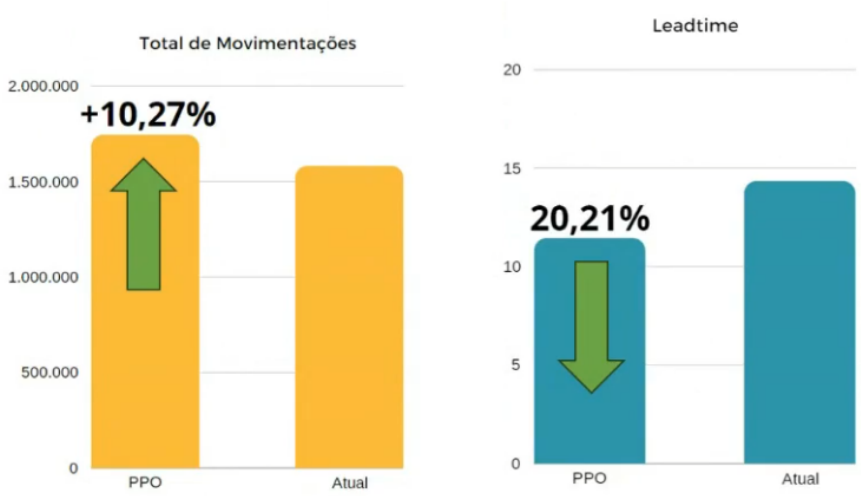


Figura 6.3: Comparação dos resultados

- Diminuição do Lead time Total: O lead time total no cenário pós PPO se mostrou com uma redução de 20,21% segundo o Gráfico 6.3, sugerindo uma eficiência aumentada no tempo médio das rotas.
- Alterações nas Rotas Mais Utilizadas: Algumas rotas que tinham um alto número de movimentações no cenário atual tiveram uma redução significativa no cenário pós PPO, enquanto outras aumentaram ou surgiram novas rotas frequentes, indicando uma redistribuição das rotas devido à otimização.
- Mudanças no Lead time das Rotas Específicas: O tempo de lead time para várias rotas específicas mudou, com algumas rotas mostrando tempos de lead time reduzidos, o que pode indicar uma melhoria na eficiência dessas rotas específicas.

A implementação de técnicas de inteligência artificial na gestão de AGVs mostra potencial não apenas para otimizar operações existentes, mas também para oferecer insights para futuras expansões ou reestruturações do sistema de movimentação.

Em resumo, os resultados encontrados da simulação indicam que a aplicação de aprendizado por reforço no modelo de simulação de AGVs pode ser uma estratégia viável para melhorar a eficiência operacional, principalmente através da redução do lead time, que é um parâmetro crítico para operações logísticas ágeis e responsivas.

Capítulo 7

Conclusão

Com a introdução do PPO, a movimentação dos AGVs se mostraram como uma redução no lead time médio devido à otimização das rotas, aumento na eficiência da tomada de decisões e o aumento do número de movimentações indicam que os AGVs podem aprender padrões ótimos e adaptar suas operações para enfrentar as demandas dinâmicas de um ambiente industrial.

Essa otimização não apenas melhora o desempenho operacional diário, mas também contribui para a escalabilidade e sustentabilidade do sistema logístico como um todo. Com os AGVs operando de forma mais inteligente, as empresas podem esperar um melhor retorno sobre o investimento em automação.

Portanto, o uso de aprendizado por reforço em AGVs parece bem promissor e altamente recomendável para ambientes que buscam melhorar continuamente suas operações logísticas e manter-se competitivos em um mercado cada vez mais dependente de soluções ágeis e automatizadas.

Este projeto visa contribuir para o avanço da aplicação de técnicas de aprendizado por reforço em sistemas industriais, especificamente na otimização de AGVs. Desejamos que os resultados obtidos possam ser aplicados em cenários reais para melhorar a eficiência operacional e reduzir os custos logísticos das empresas que utilizam essa tecnologia.

Referências bibliográficas

AZIZ, A. N. A.; SAMAN, S. M. Supply chain agility: A systematic literature review and future research. **International Journal of Physical Distribution & Logistics Management**, 2018.

BANKS, J.; CARSON, J. S.; NELSON, B. L.; NICOL, D. M. Discrete-Event System Simulation. Pearson/Prentice Hall, 2007.

BELLMAN, R. A Markovian decision process. **Journal of Mathematics and Mechanics**, JSTOR, p. 679–684, 1957.

FIGUEIRA, G.; ALMADA-LOBO, B. Hybrid simulation–optimization methods: A taxonomy and discussion. **Simulation Modelling Practice and Theory**, Elsevier, v. 46, p. 118–134, 2014. DOI: 10.1016/j.simpat.2014.03.007. Disponível em: <http://dx.doi.org/10.1016/j.simpat.2014.03.007>.

FLEXSIM Reinforcement Learning Training Documentation. [S.l.: s.n.], 2023. Disponível em: <https://docs.flexsim.com/en/22.1/ModelLogic/ReinforcementLearning/Training/Training.html>.

HENDERSON, P. et al. Deep Reinforcement Learning That Matters. **arXiv preprint arXiv:1709.06560**, 2018.

JHARKHARIA, S.; SHANKAR, R. Supply chain agility, adaptability and alignment: empirical evidence from the Indian auto components industry. **International Journal of Production Economics**, 2007.

JOHNSON, R. L. Statistical considerations for simulation in industrial applications. **Industrial Engineering and Management**, Industrial Management Press, v. 27, n. 2, p. 114–122, 2018.

JUN, J. B.; JACOBSON, S. H.; SWISHER, J. R. Application of discrete-event simulation in health care clinics: A survey. **Journal of the Operational Research Society**, Springer, v. 57, n. 2, p. 109–123, 2006.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of artificial intelligence research**, AI Access Foundation, v. 4, p. 237–285, 1996.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KONDA, V. R.; TSITSIKLIS, J. N. Actor-critic algorithms. **SIAM Journal on Control and Optimization**, SIAM, v. 38, n. 4, p. 1143–1166, 2000.

LAW, A. M. **Simulation Modeling and Analysis**. [S.l.]: McGraw-Hill New York, 2007.

LI, M. et al. **Decentralized Multi-AGV Task Allocation based on Multi-Agent Reinforcement Learning with Information Potential Field Rewards**. [S.l.: s.n.], 2021. arXiv: 2108.06886 [cs.RO].

LIU, P. et al. Reinforcement learning empowered multi-AGV offloading scheduling in edge-cloud IIoT. **Journal of Cloud Computing**, v. 11, n. 78, 2022. DOI: 10.1186/s13677-022-00352-z. Disponível em: <https://doi.org/10.1186/s13677-022-00352-z>.

LOPEZ, M. G.; KHAN, O. J. Simulation and machine learning: A dynamic duo. **Advanced Modelling and Simulations in Engineering Sciences**, Engineering Science Publishers, v. 6, n. 1, p. 23–37, 2019.

MNIH, V. et al. Human-level control through deep reinforcement learning. **Nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015.

MNIH, V. et al. Asynchronous methods for deep reinforcement learning. **arXiv preprint arXiv:1602.01783**, 2016.

NEGAHBAN, A.; SMITH, J. S. Simulation applications in the healthcare supply chain: A literature review. **Journal of Simulation**, Palgrave Macmillan UK, v. 8, n. 3, p. 129–142, 2014.

OPENAI SPINNING UP. **Proximal Policy Optimization**. [S.l.: s.n.], 2023. Disponível em: <https://spinningup.openai.com/en/latest/algorithms/ppo.html>.

PAN, J. Z.; YU, Y.; MIAO, H.; PAN, X. Towards Delivering a Coherent Self-Contained Explanation of Proximal Policy Optimization. **Unknown**, 2021.

PAN, S. J.; YANG, Q. A Survey on Transfer Learning. **IEEE Transactions on Knowledge and Data Engineering**, IEEE, v. 22, n. 10, p. 1345–1359, 2010.

PAPADIMITRIOU, C. H.; TSITSIKLIS, J. N. The Complexity of Markov Decision Processes. **Mathematics of Operations Research**, INFORMS, v. 12, n. 3, p. 441–450, 1987.

PENG, X. B. **DeepMimic**. [S.l.: s.n.], 2023. Disponível em: <https://xbpeng.github.io/projects/DeepMimic/index.html>.

PUTERMAN, M. L. **Markov Decision Processes: Discrete Stochastic Dynamic Programming**. [S.l.]: John Wiley Sons, 1995.

ROBINSON, S. Simulation: The Practice of Model Development and Use. **John Wiley & Sons**, 2004.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [S.l.]: Pearson Education Limited, 2016.

SAJAD, M. M. A.; LONG, A. A. The impact of supply chain agility on business performance: A SCM practices perspective. **International Journal of Physical Distribution & Logistics Management**, 2018.

SCHULMAN, J.; LEVINE, S. et al. Trust region policy optimization. **arXiv preprint arXiv:1502.05477**, 2015.

SCHULMAN, J.; WOLSKI, F. et al. Proximal Policy Optimization Algorithms, 2017.

SILVER, D. et al. Deterministic policy gradient algorithms. **ICML**, v. 14, p. 387–395, 2014.

SMITH, J. A.; OTHER, A. B. The role of replication in simulation. **Journal of Simulation**, Simulation Publishers, v. 10, n. 1, p. 50–59, 2015.

STABLE Baselines3 Documentation. [S.l.: s.n.], 2023. Disponível em: <https://stable-baselines3.readthedocs.io/en/master/>.

STABLE Baselines3 Proximal Policy Optimization Documentation. [S.l.: s.n.], 2023. Disponível em: <https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html>.

STOCK, T.; SELIGER, G. Opportunities of Sustainable Manufacturing in Industry 4.0. **Procedia CIRP**, Elsevier, v. 40, p. 536–541, 2016. DOI: 10.1016/j.procir.2016.01.129.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning: An Introduction**. [S.l.]: MIT press, 2018.

SUTTON, R. S.; MCALLESTER, D.; SINGH, S.; MANSOUR, Y. Policy Gradient Methods for Reinforcement Learning with Function Approximation. **Advances in neural information processing systems**, v. 12, p. 1057–1063, 2000.

TAO, F.; QI, Q.; LIU, A.; KUSIAK, A. Data-driven smart manufacturing. **Journal of Manufacturing Systems**, Elsevier, v. 48, p. 157–169, 2018. DOI: 10.1016/j.jmsy.2018.01.006.

TIOMKIN, S.; TISHBY, N. A Unified Bellman Equation for Causal Information and Value in Markov Decision Processes. **arXiv preprint arXiv:1709.06670**, 2017.

VAN BRUSSEL, H. et al. Reference architecture for holonic manufacturing systems: PROSA. **Computers in Industry**, Elsevier, v. 37, n. 3, p. 255–274, 1998. DOI: 10.1016/s0166-3615(98)00102-x.

WANG, S. et al. Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. **Computer Networks**, Elsevier, v. 101, p. 158–168, 2016. DOI: 10.1016/j.comnet.2015.12.017.

WHITTLE, P.; PUTERMAN, M. L. **Markov Decision Processes**. [S.l.]: John Wiley Sons, 1994.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. **Machine learning**, Springer, v. 8, n. 3-4, p. 229–256, 1992.

ZHONG, R. Y.; XU, X.; KLOTZ, E.; NEWMAN, S. T. Intelligent Manufacturing in the Context of Industry 4.0: A Review. **Engineering**, Elsevier, v. 3, n. 5, p. 616–630, 2017. DOI: 10.1016/j.eng.2017.05.015.

Apêndice A

Primeiro Apêndice

```python
import gym
import os
import subprocess
import socket
import json
from gym import error, spaces, utils
from gym.utils import seeding
import numpy as np

class FlexSimEnv(gym.Env):
    metadata = {'render.modes': ['human', 'rgb_array', 'ansi']}

    def __init__(self, flexsimPath, modelPath, address='localhost', port=5005, verbose=False, visible=False):
        self.flexsimPath = flexsimPath
        self.modelPath = modelPath
        self.address = address
        self.port = port
        self.verbose = verbose
        self.visible = visible

        self.lastObservation = ""

        self._launch_flexsim()

        self.action_space = self._get_action_space()
        self.observation_space = self._get_observation_space()

    def reset(self):
        self._reset_flexsim()
        state, reward, done = self._get_observation()
        return state

    def step(self, action):
        self._take_action(action)
        state, reward, done = self._get_observation()
        info = {}
        return state, reward, done, info

    def render(self, mode='human'):
        if mode == 'rgb_array':
            return np.array([0,0,0])
        elif mode == 'human':
            print(self.lastObservation)
        elif mode == 'ansi':
            return self.lastObservation
        else:
            super(FlexSimEnv, self).render(mode=mode)

    def close(self):
```

Figura A.1: FlexSimEnv (01-49)

```
49      def close(self):
50          self._close_flexsim()
51
52      def seed(self, seed=None):
53          self.seedNum = seed
54          return self.seedNum
55
56
57      def _launch_flexsim(self):
58          if self.verbose:
59              print("Launching " + self.flexsimPath + " " + self.modelPath)
60
61          args = [self.flexsimPath, self.modelPath, "-training", self.address + ':' + str(self.port)]
62          if self.visible == False:
63              args.append("-maintenance")
64              args.append("nogui")
65          self.flexsimProcess = subprocess.Popen(args)
66
67          self._socket_init(self.address, self.port)
68
69      def _close_flexsim(self):
70          self.flexsimProcess.kill()
71
72      def _release_flexsim(self):
73          if self.verbose:
74              print("Sending StopWaiting message")
75          self._socket_send(b"StopWaiting?")
76
77      def _get_action_space(self):
78          self._socket_send(b"ActionSpace?")
79          if self.verbose:
80              print("Waiting for ActionSpace message")
81          actionSpaceBytes = self._socket_recv()
82
83          return self._convert_to_gym_space(actionSpaceBytes)
84
85      def _get_observation_space(self):
86          self._socket_send(b"ObservationSpace?")
87          if self.verbose:
88              print("Waiting for ObservationSpace message")
89          observationSpaceBytes = self._socket_recv()
90
91          return self._convert_to_gym_space(observationSpaceBytes)
92
93      def _reset_flexsim(self):
94          if self.verbose:
95              print("Sending Reset message")
96          resetString = "Reset?"
```

Figura A.2: FlexSimEnv (49-96)

```
96          resetString = "Reset?"
97          if hasattr(self, "seedNum"):
98              resetString = "Reset:" + str(self.seedNum) + "?"
99          self._socket_send(resetString.encode())
100
101     def _get_observation(self):
102         if self.verbose:
103             print("Waiting for Observation message")
104         observationBytes = self._socket_recv()
105         self.lastObservation = observationBytes.decode('utf-8')
106         state, reward, done = self._convert_to_observation(observationBytes)
107
108         return state, reward, done
109
110     def _take_action(self, action):
111         actionStr = json.dumps(action, cls=NumpyEncoder)
112         if self.verbose:
113             print("Sending Action message: " + actionStr)
114         actionMessage = "TakeAction:" + actionStr + "?"
115         self._socket_send(actionMessage.encode())
116
117
118     def _socket_init(self, host, port):
119         if self.verbose:
120             print("Waiting for FlexSim to connect to socket on " + self.address + ":" + str(self.port))
121
122         self.serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
123         self.serversocket.bind((host, port))
124         self.serversocket.listen();
125
126         (self.clientsocket, self.socketaddress) = self.serversocket.accept()
127         if self.verbose:
128             print("Socket connected")
129
130         if self.verbose:
131             print("Waiting for READY message")
132         message = self._socket_recv()
133         if self.verbose:
134             print(message.decode('utf-8'))
135         if message != b"READY":
136             raise RuntimeError("Did not receive READY! message")
137
138     def _socket_send(self, msg):
139         totalsent = 0
140         while totalsent < len(msg):
141             sent = self.clientsocket.send(msg[totalsent:])
142             if sent == 0:
143                 raise RuntimeError("Socket connection broken")
```

Figura A.3: FlexSimEnv (96-143)

```
143                 raise RuntimeError("Socket connection broken")
144             totalsent = totalsent + sent
145
146     def _socket_recv(self):
147         chunks = []
148         while 1:
149             chunk = self.clientsocket.recv(2048)
150             if chunk == b'':
151                 raise RuntimeError("Socket connection broken")
152             if chunk[-1] == ord('!'):
153                 chunks.append(chunk[:-1])
154                 break;
155             else:
156                 chunks.append(chunk)
157         return b''.join(chunks)
158
159
160     def _convert_to_gym_space(self, spaceBytes):
161         paramsStartIndex = spaceBytes.index(ord('('))
162         paramsEndIndex = spaceBytes.index(ord(')'), paramsStartIndex)
163
164         type = spaceBytes[:paramsStartIndex]
165         params = json.loads(spaceBytes[paramsStartIndex+1:paramsEndIndex])
166
167         if type == b'Discrete':
168             return gym.spaces.Discrete(params)
169         elif type == b'Box':
170             return gym.spaces.Box(np.array(params[0]), np.array(params[1]))
171         elif type == b'MultiDiscrete':
172             return gym.spaces.MultiDiscrete(params)
173         elif type == b'MultiBinary':
174             return gym.spaces.MultiBinary(params)
175
176         raise RuntimeError("Could not parse gym space string")
177
178     def _convert_to_observation(self, spaceBytes):
179         observation = json.loads(spaceBytes)
180         state = observation["state"]
181         if isinstance(state, list):
182             state = np.array(observation["state"])
183         reward = observation["reward"]
184         done = (observation["done"] == 1)
185         return state, reward, done
186
187
188 class NumpyEncoder(json.JSONEncoder):
189     def default(self, obj):
190         if isinstance(obj, np.integer):
```

Figura A.4: FlexSimEnv (96-190)

```
190             if isinstance(obj, np.integer):
191                 return int(obj)
192             elif isinstance(obj, np.floating):
193                 return float(obj)
194             elif isinstance(obj, np.ndarray):
195                 return obj.tolist()
196             return json.JSONEncoder.default(self, obj)
197
198
199     def main():
200
201         env = FlexSimEnv(
202             flexsimPath = "C:/Program Files/FlexSim 2022/program/flexsim.exe",
203             modelPath = "C:/Users/USERNAME/Documents/FlexSim 2022 Projects/MyModel.fsm",
204             verbose = True,
205             visible = True
206             )
207
208         for i in range(2):
209             env.seed(i)
210             observation = env.reset()
211             env.render()
212             done = False
213             rewards = []
214             while not done:
215                 action = env.action_space.sample()
216                 observation, reward, done, info = env.step(action)
217                 env.render()
218                 rewards.append(reward)
219                 if done:
220                     cumulative_reward = sum(rewards)
221                     print("Reward: ", cumulative_reward, "\n")
222         env._release_flexsim()
223         input("Waiting for input to close FlexSim...")
224         env.close()
225
226
227     if __name__ == "__main__":
228         main()
```

Figura A.5: FlexSimEnv (190-228)

Apêndice B

Segundo Apêndice

```python
import gym
from flexsim_env import FlexSimEnv
from stable_baselines3.common.env_checker import check_env
from stable_baselines3 import PPO
from stable_baselines3.common.env_util import make_vec_env

def main():
    print("Initializing FlexSim environment...")

    # Create a FlexSim OpenAI Gym Environment
    env = FlexSimEnv(
        flexsimPath = "C:/Program Files/FlexSim 2023 Update 1/program/flexsim.exe",
        modelPath = "C:/Users/HAWKEYE_FLEXSIM/OneDrive - FlexSim Brasil/Área de Trabalho/TCC/FlexSim Reinforcemer
        verbose = False,
        visible = False
        )
    check_env(env) # Check that an environment follows Gym API.

    # Training a baselines3 PPO model in the environment
    model = PPO("MlpPolicy", env, verbose=1)
    print("Training model...")
    model.learn(total_timesteps=10000)

    # save the model
    print("Saving model...")
    model.save("MyTrainedModel")

    input("Waiting for input to do some test runs...")

```

Figura B.1: Gym Environment (1-29)

```python

    # Run test episodes using the trained model
    for i in range(2):
        env.seed(i)
        observation = env.reset()
        env.render()
        done = False
        rewards = []
        while not done:
            action, _states = model.predict(observation)
            observation, reward, done, info = env.step(action)
            env.render()
            rewards.append(reward)
            if done:
                cumulative_reward = sum(rewards)
                print("Reward: ", cumulative_reward, "\n")
    env._release_flexsim()
    input("Waiting for input to close FlexSim...")
    env.close()


if __name__ == "__main__":
    main()
```

Figura B.2: Gym Environment (29-51)

Apêndice C

Terceiro Apêndice

```python
import json
from stable_baselines3 import PPO
from http.server import BaseHTTPRequestHandler, HTTPServer
from urllib.parse import urlparse, parse_qs
import numpy as np

class FlexSimInferenceServer(BaseHTTPRequestHandler):

    def do_GET(self):
        params = parse_qs(urlparse(self.path).query)
        self._handle_reply(params)

    def do_POST(self):
        content_length = int(self.headers['Content-Length'])
        body = self.rfile.read(content_length)
        params = parse_qs(body)
        self._handle_reply(params)

    def _handle_reply(self, params):
        if len(params):
            observation = []
            if b'observation' in params.keys():
                observationBytes = params[b'observation'][0]
                observation = np.array(json.loads(observationBytes))
            elif 'observation' in params.keys():
                observationBytes = params['observation'][0]
                observation = np.array(json.loads(observationBytes))
            if isinstance(observation, list):
                observation = np.array(observation)
            action, _states = FlexSimInferenceServer.model.predict(observation)
            self.send_response(200)
            self.send_header("Content-type", "application/json")
            self.end_headers()
            self.wfile.write(bytes(json.dumps(action, cls=NumpyEncoder), "utf-8"))
            return

        self.send_response(200)
        self.send_header("Content-type", "text/html")
        self.end_headers()
        self.wfile.write(bytes("", "utf-8"))

```

Figura C.1: FlexSim Inference Server Code (1-41)

```
41
42
43  class NumpyEncoder(json.JSONEncoder):
44      def default(self, obj):
45          if isinstance(obj, np.integer):
46              return int(obj)
47          elif isinstance(obj, np.floating):
48              return float(obj)
49          elif isinstance(obj, np.ndarray):
50              return obj.tolist()
51          return json.JSONEncoder.default(self, obj)
52
53
54  def main():
55      print("Loading model...")
56      model = PPO.load("MyTrainedModel")
57      FlexSimInferenceServer.model = model
58
59      # Create server object
60      print("Starting server...")
61      hostName = "localhost"
62      serverPort = 80
63      webServer = HTTPServer((hostName, serverPort), FlexSimInferenceServer)
64      print("Server started http://%s:%s" % (hostName, serverPort))
65
66      # Start the web server
67      try:
68          webServer.serve_forever()
69      except KeyboardInterrupt:
70          pass
71
72      webServer.server_close()
73      print("Server stopped.")
74
75
76  if __name__ == "__main__":
77      main()
```

Figura C.2: FlexSim Inference Server Code (41-77)