UNIVERSIDADE FEDERAL DO MARANHÃO
Programa de Pós-Graduação em Ciência da Computação

Frederic Menezes Ferreira

Escalonamento de Produção em Manufatura em Rede: uma abordagem multicritério utilizando aprendizagem por reforço.

São Luís - MA
2024

Frederic Menezes Ferreira

Escalonamento de Produção em Manufatura em Rede: uma abordagem multicritério utilizando aprendizagem por reforço.

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Ciência da Computação, ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal do Maranhão.

Programa de Pós-Graduação em Ciência da Computação

Universidade Federal do Maranhão

Orientador: Prof. Dr. Alexandre César Muniz de Oliveira

Coorientador: Prof. Dr. Bruno Feres de Souza

São Luís - MA

2024

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Menezes Ferreira, Frederic.
Escalonamento de Produção em Manufatura em Rede : uma abordagem multicritério utilizando aprendizagem por reforço / Frederic Menezes Ferreira. - 2024.
94 p.

Coorientador(a) 1: Bruno Feres de Souza.
Orientador(a): Alexandre César Muniz de Oliveira.
Dissertação (Mestrado) - Programa de Pós-graduação em Ciência da Computação/ccet, Universidade Federal do Maranhão, São Luís, 2024.

1. Rede de Fabricação Auto-otimizada. 2. Aprendizagem Por Reforço. 3. Otimização de Política Proximal. I. César Muniz de Oliveira, Alexandre. II. Feres de Souza, Bruno. III. Título.

Frederic Menezes Ferreira

Escalonamento de Produção em Manufatura em Rede: uma abordagem multicritério utilizando aprendizagem por reforço.

Dissertação apresentada como requisito parcial para obtenção do título de Mestre em Ciência da Computação, ao Programa de Pós-Graduação em Ciência da Computação, da Universidade Federal do Maranhão.

Dissertação aprovada em, São Luís - MA, 07 de Agosto de 2024:

Prof. Dr. Alexandre César Muniz de Oliveira
Orientador
Universidade Federal do Maranhão

Prof. Dr. Bruno Feres de Souza
Coorientador
Universidade Federal do Maranhão

Prof. Dr. Areolino de Almeida Neto
Examinador Interno
Universidade Federal do Maranhão

Prof. Dr. Antônio Augusto Chaves
Examinador Externo
Universidade Federal de São Paulo

São Luís - MA
2024

Aos meus pais, mulher, filhos, irmãos, família e amigos.

Agradecimentos

Primeiramente, agradeço a Deus, por sempre mostrar o caminho e colocar pessoas boas na minha vida.

Quero expressar minha gratidão à minha família. À minha mãe, pelos conselhos, ensinamentos, e pelo carinho com meus filhos (seus netos Isaac, Heitor e Giovana) e com minha esposa, Patricia. Enquanto eu estava no Laboratório LACMOR, foi ela quem cuidou deles, levando-os para passear e muitas vezes cuidando de Giovana quando eu tinha que ir a congressos para apresentar minha pesquisa. Agradeço também ao meu pai, que me buscava de madrugada no laboratório quando eu precisava ficar até mais tarde, e à minha esposa, por estar sempre ao meu lado, me dando força. Ela viajou comigo para São José dos Campos, em São Paulo, quando apresentei meu trabalho no SBPO, e cuidou da nossa casa e dos nossos filhos na minha ausência. Aos meus filhos, Isaac, o mais velho de 17 anos, Heitor de 14 anos, e Giovana de 4 anos, vocês são a razão de tudo, minha força e minha motivação.

Agradeço profundamente ao meu orientador, por me aceitar como orientando e por tudo que fez por mim desde a graduação. Ele também foi meu orientador na graduação em Ciência da Computação. Ao meu co-orientador, Bruno Feres, pela imensa ajuda com minha pesquisa e revisão, e aos professores Marcos e Victor pelos conselhos e pela amizade. Sou muito grato a todos.

Gostaria de expressar minha gratidão ao meu chefe, Rodolfo Alván Casana Sifuentes, chefe do Departamento de Física, por todo o apoio que me deu ao me liberar para fazer o mestrado e por sempre atender às minhas solicitações. Aos professores Areolino e Antonio Chaves, por aceitarem fazer parte da minha banca. Sou grato a todos eles pelos ensinamentos.

Agradeço também a todos os professores da PPGCC, especialmente meu orientador, por todo o ensinamento, apoio, palavras de incentivo, paciência e por não ter desistido de mim.

A todos os meus amigos do laboratório, pelas ideias, sugestões, brincadeiras, lanches e por tudo mais. Obrigado, me sinto em casa com vocês. A todos os profissionais com quem trabalhei, por sempre ajudarem, tirarem dúvidas, compartilharem experiências e aprenderem junto comigo.

A todos que compõem a PPGCC-UFMA: professores, alunos e funcionários, meu muito obrigado.

Ao longo da vida, cruzamos caminhos com pessoas que estão conosco por um breve período e outras que permanecem ao nosso lado por toda a jornada. Agradeço aos amigos verdadeiros que sempre estiveram presentes em minha vida, tanto nos momentos bons quanto nos desafiadores, oferecendo ajuda, incentivo, diversão e paciência nos momentos difíceis. A

todos vocês, meu mais sincero agradecimento.

Muito obrigado a todos que, de alguma forma, contribuíram para esta conquista.

"Aprendizagem nunca esgota a mente."

(Leonardo da Vinci)

Resumo

Um controle eficiente de fabricação desempenha um papel fundamental na capacidade da indústria manufatureira de atender às crescentes demandas por produção personalizada, caracterizada por rápidas mudanças nas preferências dos clientes. Para otimizar configurações de manufatura flexíveis e altamente automatizadas, é essencial incorporar decisões autônomas no planejamento e execução da produção. Contudo, o desenvolvimento de um controle de fabricação resiliente e proativo, capaz de antecipar mudanças repentinas na prática industrial, é um desafio. Isso exige ferramentas inteligentes de programação da produção. Diversas tecnologias orientadas a dados têm sido adotadas em pesquisas de programação da produção, dentre as quais a aprendizagem por reforço destaca-se por sua capacidade de mapear observações do ambiente para ações que melhoram o desempenho. Esta dissertação apresenta um *framework* de aprendizagem por reforço para resolver o problema de programação da produção com eventos dinâmicos em uma unidade local de uma rede de manufatura auto-otimizada, visando encontrar um cronograma de produção ótimo. O algoritmo de aprendizagem por reforço treina um agente para capturar a relação entre as informações do chão de fábrica e os critérios a serem atingidos, tomando decisões em tempo real em um sistema sujeito a eventos inesperados. Propõe-se um cenário de validação no qual o agente aceita demandas de produção considerando prioridades em relação a três critérios (econômico, sustentabilidade e variabilidade), dada a carga do sistema e os possíveis atrasos e perdas financeiras. Para isso, um ambiente de aprendizagem por reforço é implementado com *software* de código aberto de última geração. O ambiente, projetado como um problema de agente único, permite que o agente decida quais demandas produzir a cada passo de tempo. A função de recompensa, projetada para orientar o agente, equilibra a influência de cada critério com um fator de priorização. Implementam-se três algoritmos de Aprendizagem por Reforço: *Deep Q-Network* (DQN), *Proximal Policy Optimization* (PPO) e *PPO Recorrente*. O DQN é um algoritmo *off-policy* baseado em *Q-learning* que utiliza uma rede neural para aproximar a função Q. O PPO é um algoritmo *on-policy* baseado em gradiente de política, com destaque para sua estabilidade e eficiência. O *PPO Recorrente* incorpora memória, tornando-o adequado para problemas com dependências temporais, como o agendamento dinâmico. A abordagem de *aprendizagem por reforço* é avaliada comparando suas soluções com dados simulados. Os resultados demonstram que a abordagem gera cronogramas mais lucrativos e personalizados, ou mais sustentáveis, dependendo do critério. O desempenho do agente é influenciado pelo fator de priorização na função de recompensa.

Palavras-chave: Rede de fabricação auto-otimizada, Aprendizagem por reforço, Otimização de Política Proximal.

Abstract

Efficient manufacturing control plays a fundamental role in the manufacturing industry's ability to meet the growing demands for personalised production, characterised by rapid changes in customer preferences. To optimise flexible manufacturing settings marked by high automation, it is essential to incorporate autonomous decisions during production planning and execution. However, the challenge lies in developing a manufacturing control system that is resilient and, ideally, proactive, anticipating sudden changes efficiently in industrial practice. Achieving such a goal requires the use of intelligent production scheduling tools. Many data-driven technologies have been adopted in production scheduling research, with Reinforcement Learning (RL) being a promising candidate capable of establishing a direct mapping from environment observation to performance-enhancing actions. This dissertation presents an RL framework to solve the dynamic scheduling problem within a local unit of a self-optimised manufacturing network, aiming to find an optimal production schedule. The RL algorithm trains a scheduling agent, capturing the relationship between factory floor information and scheduling criteria to make real-time decisions for a manufacturing system subject to frequent unexpected events. We propose an initial validation scenario where the agent must accept production demands considering priorities related to three performance criteria (economic, sustainability, and variability), given the current system load, which can impact delays and, consequently, financial losses. A Reinforcement Learning environment is introduced using state-of-the-art open-source software. The environment is designed as a single-agent problem, where the RL agent decides which demands to dispatch for production on the available machines at each time step. To guide the agent towards an optimal schedule, a reward function balances each criterion's influence using a prioritisation factor. Additionally, a state-of-the-art RL algorithm is implemented. The RL approach is evaluated by comparing its solutions to a simulated dataset. The results show that the approach can generate more profitable and personalised, or more sustainable, schedules, depending on the adopted criterion. The agent's performance is influenced by the prioritisation factor in the reward function.

Keywords: Self-optimizing manufacturing network, Reinforcement learning, Proximal Policy Optimization

Lista de ilustrações

Lista de tabelas

Lista de Algoritmos

Lista de Abreviaturas e Siglas

A2C *Asynchronous Methods for DRL.*

ANN Rede Neural Artificial, do inglês *Artificial Neural Network*.

CCGP *Cooperative Coevolution* GP.

DDPG *Deep Deterministic Policy Gradient*.

DEC-MDP Processo de Decisão de Markov descentralizado.

DQN *Deep Q-network*.

DRL Aprendizagem por Reforço Profunda, do inglês *Deep Reinforcement Learning*.

FJSP Problema de *Job Shop* Flexível, do inglês *Flexible Job Shop Problem*.

GA Algoritmo Genético, do inglês *Genetic Algorithm*.

GAE *General Advantage Estimate*.

GP Programação Genética, do inglês *Genetic Programming*.

IoT Internet das Coisas.

JSP Problema de *Job Shop*, do inglês *Job Shop Problem*.

KL Kullback-Leibler.

MDP Processo de Decisão de Markov, do inglês *Markov Decision Process*.

MTGP GP de multi-árvores.

PPO *Proximal Policy Optimization*.

RL Aprendizagem por Reforço, do inglês *Reinforcement Learning*.

SOMN Rede de Manufatura Auto Otimizadas, do inglês *Self-Optimizing Manufacturing Network*.

SOMS *Self-Organizing Manufacturing Systems*.

SVM Máquina de Vetores de Suporte, do inglês *Support Vector Machine*.

TD Diferença Temporal, do inglês *Temporal-Difference*.

TRPO *Trust Region Policy Optimization*.

UPPS Produtos e Serviços Ultrapersonalizados, do inglês *Ultra-Personalized Products and Services*.

Sumário

1 Introdução

Os paradigmas de manufatura têm se deslocado da produção e customização em massa para modelos de personalização, favorecendo a adaptação proativa no tocante a ampla variedade e fabricação de baixo volume. Tem havido uma tendência mundial de mudar o modo de produção em massa globalizado para um mais local, sem abrir mão do alto rendimento e qualidade. Outros novos requisitos são a capacidade de responder a uma escala de produção dinâmica com condições de mercado incertas e cadeias de suprimentos instáveis ou quebradas (ZHENG et al., 2021).

O regionalismo e a autenticidade também são desejáveis, pois os clientes reagem de forma ainda mais positiva à sustentabilidade e à saúde da força de trabalho. A adaptação da produção e da logística exige a otimização dos fluxos de materiais e das funções logísticas para que os produtos personalizados em massa sejam entregues próximo ao mercado (RAUCH; DALLASEGA; MATT, 2017).

A personalização permite que as empresas adotem uma estratégia de diferenciação para competir em valor adicional para o cliente, em vez de concorrentes apenas em valor econômico. Nos últimos anos, os robôs cooperativos (cobots) e fabricação aditiva tornaram a personalização de produtos um conceito mais acessível a um público amplo. Além disso, prevê-se que o mercado de personalização cresça devido aos clientes se tornarem muito exigentes (TORN; VANEKER, 2019).

Seguindo essa linha, no futuro próximo, a tendência é a produção de Produtos e Serviços Ultrapersonalizados, do inglês *Ultra-Personalized Products and Services* (UPPS), que seria uma fase de mercado envolvendo produtos adaptados às necessidades do indivíduo em pequenas séries, até o tamanho de lote igual a um. Graças à produção aditiva, o UPPS oferece um nível de personalização a um valor razoável que não era possível no passado (NACHTIGALL et al., 2020).

A personalização permite que as empresas concorram com base no valor agregado, e não apenas no custo. Para conseguir isso, os recursos de fabricação devem mudar para usar máquinas flexíveis e configuráveis e otimização dinâmica de fluxos de materiais e serviços de logística (RAUCH; DALLASEGA; MATT, 2017). Além disso, há uma tendência mais ampla de produção e logística sustentáveis, exigindo manufatura localizada, em que os itens são produzidos mais próximos do cliente. A produção localizada traz outros desafios, pois as condições de mercado, a disponibilidade de matérias-primas e a operação das redes de abastecimento podem se tornar mais instáveis, o que aumenta ainda mais a necessidade de uma produção dinâmica e flexível.

Uma Rede de Manufatura Auto Otimizadas, do inglês *Self-Optimizing Manufacturing Network* (SOMN) é um novo conceito organizacional esperado para produzir produtos perso-

nalizados em demandas dinâmicas de volume por causa de sua estrutura mutável e controle adaptativo de manufatura (QIN; LU, 2021).

No trabalho de Lu, Xu e Wang (2020), uma rede de manufatura auto-organizada é descrita como uma rede composta por unidades de manufatura autônomas que consistem em ferramentas de software, equipamentos de hardware e operadores, conectados de maneiras dependentes da situação que podem mudar sua estrutura interna e funções com intervenção externa mínima para alcançar operações de fabricação otimizadas e desempenho do sistema em resposta a condições imprevistas e evolução ao longo do tempo. Os requisitos atualizados contam com autoconfiguração (paradigma *plug-and-produce*), auto-otimização (comportamento autônomo ideal) e auto-correção (recuperação de anormalidade externa livre).

Avanços tecnológicos recentes apoiam esse desenvolvimento de manufatura localizada e personalizada, incluindo manufatura aditiva e robôs cooperativos. As características estruturais do SOMN são:

- **Distribuído**: as unidades de fabricação são conectadas de maneira ponto a ponto, segundo uma logística que favoreça o transporte de volumes variados de produtos semi-acabados e finalizados;
- **Bottom-up**: as decisões de controle são geradas por meio de negociações locais e coordenação entre unidades fabris ao invés de aceitar passivamente comandos vindos do topo;
- **Adaptabilidade**: volume de produção e variedade de produtos podem ser ajustados dinamicamente em resposta à demanda;
- **Autonomia individual**: as unidades fabris decidem sobre configurações precisas e tarefas localizadas com base na comunicação e negociação *peer-to-peer*.

A otimização em um SOMN refere-se a fornecer o controle dos processos de produção a partir de indicadores de desempenho, como *makespan*, utilização de recursos, carga de trabalho, e *lead time*; indicadores econômicos, como lucro e custos; bem como consumo de energia, redução de desperdício , emissões de CO2 e outros desempenhos em nível de sistema relacionados à sustentabilidade (AZARAKHSH; SAHEBI; HOSSEINI, 2021), adaptabilidade e escalabilidade (QIN; LU, 2021).

Com isso, a função de auto-otimização aloca recursos de fabricação e decide os cronogramas de tarefas de fabricação. Embora os algoritmos heurísticos tenham sido amplamente aplicados para otimizar cronogramas de produção, como encontrado em (OGUNSAKIN; MEHANDJIEV; MARÍN, 2018), tais algoritmos estáticos não funcionam bem para personalização em massa, pois a natureza dinâmica dos cronogramas de produção os torna continuamente re-otimizados (QIN; LU, 2021).

Além disso, a tomada de decisão requer programação de produção mais adaptável e flexível em uma rede de fabricação que entrega produtos altamente personalizados e busca objetivos locais e globais. O agendamento baseado em heurística codificada com um modelo estático predefinido carece de escalabilidade e habilidades de autoaprendizagem em um cenário que muda rapidamente (QIN; LU, 2021).

Portanto, o SOMN precisa de um algoritmo de agendamento baseado em aprendizado com características descentralizadas para atender aos requisitos de agendamento dinâmico de personalização em massa em manufatura distribuída. A questão é aprender a decidir por diferentes condições e não definir a tomada de decisão para um cenário que muda rapidamente (QIN; LU, 2021).

Nesse sentido, a Aprendizagem por Reforço, do inglês *Reinforcement Learning* (RL), se destaca como um paradigma de aprendizado de máquina promissor para resolver problemas de tomada de decisão sequencial em ambientes complexos (CUNHA et al., 2018; MNIH et al., 2015b; ZHANG et al., 2020a; ZHANG et al., 2022; CHANG et al., 2022; WANG et al., 2022). Na RL, um agente aprende a interagir com um ambiente, tomando ações e recebendo recompensas ou penalidades. O objetivo é aprender uma *política*, ou seja, uma função que mapeia estados do ambiente para ações, de forma a maximizar a recompensa acumulada ao longo do tempo.

A Aprendizagem por Reforço Profunda, do inglês *Deep Reinforcement Learning* (DRL), é um subcampo da RL que utiliza redes neurais profundas para aproximar funções complexas, permitindo lidar com ambientes com muitos estados e ações, o que facilita o aprendizado da *política*. Essa capacidade de lidar com ambientes complexos e dinâmicos é particularmente relevante para a produção personalizada em rede, que demanda abordagens de otimização robustas à incerteza, mais reativas e até proativas (QIN; LU, 2021). A DRL tem demonstrado sucesso em áreas como jogos (AlphaGo) e robótica, evidenciando seu potencial para lidar com esse tipo de problema. Os algoritmos *Deep Q-network* (DQN), *Proximal Policy Optimization* (PPO) e PPO Recorrente, utilizados neste trabalho, são exemplos de algoritmos de DRL.

A DRL se mostra especialmente adequada para a produção personalizada em rede pois, em vez de planejar com base em dados estáticos, permite aprender com experiências recentes, característica essencial para otimização *online* (QIN; LU, 2021; HUBBS et al., 2020b). Problemas de otimização online, como o agendamento dinâmico em redes de manufatura, possuem essa característica de aprendizado contínuo a partir da interação com o ambiente, e estão sendo resolvidos por abordagens baseadas em DRL com precisão satisfatória. Em cenários de produção personalizada, assumir ações antecipadas para lidar com incertezas nas relações com consumidores e fornecedores é crucial, e a capacidade adaptativa da DRL a torna uma ferramenta valiosa para esse fim.

1.1 Objetivos

O objetivo deste trabalho é propor um *framework* para tomada de decisão multicritério baseada em Aprendizagem por Reforço Profundo no contexto de Redes de Manufatura Auto Otimizadas (*Self-Optimizing Manufacturing Networks*), considerando aspectos de incerteza e equilibrando indicadores econômicos, métricas de personalização de produtos e de produção sustentável.

A proposta utiliza um simulador de uma única unidade fabril modelada a partir de variáveis de estado e processos que responde à produção de manufaturados personalizados a partir de demandas de clientes e matéria-prima entregue por uma cadeia de fornecedores locais. O sistema simulado prevê a interface com outras unidades de manufatura operando igualmente com autonomia para cooperarem em busca de indicadores de desempenhos medidos globalmente.

1.1.1 Objetivos Específicos

O objetivo geral pode ser dividido nos seguintes objetivos específicos:

- Implementar o *framework* de tomada de decisão multicritério baseado em Aprendizagem por Reforço Profundo (DRL) adequado para o contexto de *Self-Optimizing Manufacturing Networks* (SOMN), considerando a incerteza inerente ao ambiente de produção dinâmico.
- Implementar algoritmos de Aprendizagem por Reforço Profundo (DRL) para a tomada de decisões em um ambiente simulado de uma única unidade fabril;
- Validar o *framework* proposto por meio de experimentos e estudos de caso, demonstrando sua capacidade de melhorar a eficiência operacional, a adaptabilidade e a sustentabilidade em comparação com métodos tradicionais de tomada de decisão na manufatura.
- Avaliar a eficácia do *framework* proposto em equilibrar o desempenho de indicadores econômicos, de personalização de produtos (variabilidade) e de sustentabilidade em um contexto de SOMN.

1.1.2 Contribuições originais

Destacam-se como principais contribuições desse trabalho:

- **Simulador de unidade de rede de manufatura**: – Simulador implementado em linguagem Python modelando uma unidade de manufatura parametrizável com capacidade de produção de itens a partir de demandas de consumidores e matéria prima de fornecedores. A unidade de manufatura guarda relação com o *Job Shop*, na qual a especificação de cada item equivale numericamente ao tempo de máquina que pode ser otimizado por critérios econômicos e não-econômicos.

- **Agente baseado em Aprendizado por Reforço Profundo**: Agentes baseados em PPO podem ser treinados para aceitar demandas considerando o risco de atraso (com consequente não entrega do produto) e priorizar as demandas buscando equilíbrio entre os critérios de desempenho da rede de manufatura. Dessa forma, agentes aprendem a responder adequadamente a variações na distribuição de probabilidade dos atrasos gerados, que são influenciados pela quantidade e complexidade dos pedidos, bem como carga atual da unidade fabril.

1.2 Organização do Trabalho

Os capítulos desta dissertação foram organizados da seguinte forma:

- O Capítulo 1 forneceu uma introdução ao contexto da produção personalizada em rede e apresenta a motivação para o desenvolvimento deste trabalho. São discutidos os desafios enfrentados na tomada de decisões autônomas em ambientes complexos de manufatura.
- O Capítulo 2 analisa o desenvolvimento e as pesquisas mais recentes sobre agendamento de produção dinâmico. As técnicas revisadas variam desde regras tradicionais projetadas manualmente e abordagens meta-heurísticas até estudos recentes que incorporam os avanços em algoritmos evolutivos e aprendizado de máquina.
- O Capítulo 3 aborda a fundamentação teórica necessária para compreender os conceitos-chave utilizados neste trabalho. São explorados temas como Produção Personalizada em Rede, Agendamento de Produção Dinâmico, Aprendizagem por Reforço, DQN e *Proximal Policy Optimization* (PPO).
- Os Capítulos 4 e 5 descrevem a taxonomia e adotada e a modelagem de variáveis e processos para simular, treinar e avaliar os agentes baseados em Aprendizado por Reforço.
- O Capítulo 6 apresenta os resultados obtidos a partir dos experimentos realizados com os agentes treinados. São discutidos os desempenhos dos agentes em termos de recompensa, perda e capacidade de adaptação a diferentes perfis de atraso.
- O Capítulo 7 traz as conclusões deste trabalho, destacando as contribuições alcançadas, as limitações encontradas e as possíveis direções para pesquisas futuras na área de produção personalizada em rede.

2 Trabalhos Relacionados

A crescente demanda por personalização em massa requer métodos de agendamento mais flexíveis e adaptáveis, capazes de lidar com configurações de produtos e operações diversificadas (QIN; LU, 2021). Em cenários de produção flexíveis, o problema de agendamento é conhecido como Problema de *Job Shop* Flexível, do inglês *Flexible Job Shop Problem* (FJSP), que busca otimizar a alocação de tarefas e a sequência de operações em um conjunto de máquinas. Diferentemente do Problema de *Job Shop*, do inglês *Job Shop Problem* (JSP), que considera apenas a sequência de tarefas, o FJSP envolve também a seleção das máquinas (roteamento), aumentando a complexidade do problema.

Encontrar o agendamento ideal, ou seja, aquele que maximiza a eficiência e minimiza o tempo de espera, é uma tarefa matematicamente difícil. Mesmo em cenários de agendamento estático, onde as variáveis não mudam ao longo do tempo, encontrar a solução perfeita é um desafio. Isso ocorre porque os problemas de agendamento, tanto estáticos quanto dinâmicos, são considerados como pertencentes à classe de problemas NP-difíceis (GOREN; SABUNCUOGLU, 2009). Esses problemas são conhecidos por serem intratáveis do ponto de vista analítico, o que significa que não existe uma solução que possa ser encontrada em um tempo viável por algoritmos convencionais à medida que o tamanho do problema aumenta. Portanto, ao lidar com o agendamento, é necessário fazer compromissos entre diferentes objetivos, como tempo de resposta e qualidade do agendamento, para encontrar uma solução que seja praticamente útil.

A pesquisa de agendamento estático foi responsável pela criação de uma grande quantidade de técnicas, e muitas tentativas foram feitas para transferi-las para o ambiente dinâmico. Essas técnicas são revisadas na Seção 2.1. O progresso recente em estatísticas, algoritmos evolucionários, ciência de dados e aprendizado de máquina alimenta o desenvolvimento de abordagens baseadas em dados, que visam utilizar uma variedade mais ampla de informações no chão de fábrica para fornecer decisões sólidas com maior qualidade e responsividade; essas técnicas são revisadas nas Seções 2.2 e 2.3.

2.1 Abordagens Tradicionais

Da grande família de técnicas criadas a partir da pesquisa em agendamento estático, as mais populares são as regras de prioridade e as meta-heurísticas. Devido à diferença no tempo e na frequência de engajamento, suas aplicações a problemas de agendamento dinâmico podem ser categorizadas como abordagens completamente reativas, robustas-proativas e preditivas-reativas (OUELHADJ; PETROVIC, 2009).

As regras de prioridade constroem um cronograma de maneira reativa, e são amplamente

aplicadas devido à sua facilidade de implementação. O desempenho e a comparação das regras de prioridade sob diferentes cenários têm sido estudados por um longo tempo, e o consenso é que nenhuma única regra de prioridade fornece desempenho estritamente superior em todos os objetivos em todos os cenários (SELS; GHEYSEN; VANHOUCKE, 2012; XIONG et al., 2017; ĐURASEVIĆ; JAKOBOVIĆ, 2018). Também é amplamente aceito que o desempenho das regras de prioridade compostas geralmente é melhor do que seus blocos de construção (SELS; GHEYSEN; VANHOUCKE, 2012).

Cronogramas (*Schedules*) também podem ser produzidos antecipadamente considerando eventos dinâmicos, meta-heurísticas como Busca Tabu, Recozimento Simulado, Algoritmo Genético, Otimização por Enxame de Partículas e Otimização por Colônia de Formigas são populares no desenvolvimento de cronogramas robustos-proativos (OUELHADJ; PETROVIC, 2009). Além dos objetivos de desempenho comumente usados, como tempo de fluxo, *makespan* e atraso, a robustez deve ser incluída como uma métrica de desempenho adicional para avaliar a capacidade de um cronograma de absorver interrupções (LIU et al., 2017; ZHANG; SONG; WU, 2020). O desenvolvimento de um cronograma robusto começa a partir de uma solução randomizada ou uma população de soluções; a qualidade do cronograma é melhorada através do processo demorado de avaliação iterativa (geralmente por simulação de produção com eventos dinâmicos) e reprodução. O alto custo resultante do desenvolvimento do cronograma compromete a capacidade das meta-heurísticas de lidar com eventos dinâmicos frequentes. A maioria dos estudos baseados em técnicas de meta-heurísticas foca em quebras de máquina; o agendamento não inclui ou apenas inclui um processo de re-agendamento simples, já que o impacto de quebras de máquinas infrequentes pode ser minimizado efetivamente pela incorporação de métricas de robustez no desenvolvimento do cronograma.

Quando o re-agendamento é necessário, o cronograma de produção desenvolvido pode ser reparado por heurísticas ou substituído por regras de prioridade. Tal abordagem é referida como a abordagem preditiva-reativa, que mantém a eficiência mais alta possível até a ocorrência de eventos e fornece decisões de agendamento viáveis e oportunas posteriormente. Estudos recentes também recorrem a abordagens híbridas para reduzir a latência decisória das meta-heurísticas, possibilitando assim sua capacidade de lidar com interrupções frequentes, como a chegada e o cancelamento de tarefas. Exemplos incluem a combinação de busca tabu e algoritmo genético (LI et al., 2020); combinação de otimização por enxame de partículas e algoritmo genético (REN et al., 2022).

2.2 Abordagens Recentes Baseadas em Dados

As abordagens tradicionais enfrentam o dilema da eficiência versus qualidade. O alto custo computacional das meta-heurísticas compromete sua utilidade em um ambiente com eventos dinâmicos frequentes, onde a eficácia do cronograma ótimo pode deteriorar rapidamente.

Por outro lado, abordagens baseadas em regras produzem soluções quase em tempo real a um custo de qualidade, devido ao tipo limitado de informações e operações matemáticas simples envolvidas na tomada de decisões.

Pesquisas recentes visam desenvolver estratégias de agendamento que usem uma grande quantidade de informações do chão de fábrica para tomar decisões de agendamento de alta qualidade quase em tempo real, deslocando o processo de desenvolvimento demorado offline e aplicando a estratégia desenvolvida à tomada de decisões online. Algumas abordagens candidatas são Programação Genética, do inglês *Genetic Programming* (GP), aprendizado supervisionado e RL.

GP pode ser usado para desenvolver regras de prioridade. Comparadas às regras de prioridade projetadas manualmente, que utilizam uma variedade limitada de informações sobre tarefas ou máquinas, o GP oferece uma maneira de incluir uma variedade de informações em tempo real na tomada de decisões; o desenvolvimento de regras é automatizado por meio de evolução iterativa, baseada em população e randomizada.

Estudos iniciais simplificam o FJSP para JSP acoplando a regra de sequenciamento com uma regra de roteamento universal (HO; TAY; LAI, 2007; PICKARDT et al., 2010). Recentemente, o GP também é usado para co-evoluir regras de roteamento e sequenciamento. Yska, Mei e Zhang (2018) propõem um *framework Cooperative Coevolution* GP (CCGP) para evoluir regras de roteamento e sequenciamento em duas populações separadas. Zhang, Mei e Zhang (2018) propõem a arquitetura GP de multi-árvores (MTGP) para explorar a interação entre regras de roteamento e sequenciamento. Também há pesquisas que visam melhorar o desempenho por meio da seleção de características (ZHANG; MEI; ZHANG, 2019; ZHANG et al., 2020b). As abordagens baseadas em GP realmente melhoram a eficácia das regras desenvolvidas ao incorporar uma variedade maior de informações como entrada; no entanto, sua tomada de decisão ainda depende de cálculos simples e operações lógicas.

Outra possibilidade está no uso de técnicas com parametrização complexa, como algoritmos de aprendizado supervisionado. A suposição-chave, em sua aplicação ao agendamento, é que o conhecimento de agendamento ótimo pode ser aprendido por algoritmos parametrizados, como Rede Neural Artificial, do inglês *Artificial Neural Network* (ANN), Máquina de Vetores de Suporte, do inglês *Support Vector Machine* (SVM) e Árvore de Decisão. Algoritmos de aprendizado supervisionado melhoram a qualidade da solução por meio de otimização analítica ou abordagem baseada em gradiente, levando a uma convergência mais rápida para uma solução satisfatória em comparação com abordagens evolucionárias.

ANN é principalmente usada para seleção de regras de prioridade. Mouelhi-Chibani e Pierreval (2010) propõem um mecanismo de seleção de regra dinâmica que é acionado por eventos em tempo real para resolver o problema de agendamento dinâmico de tarefas; Guh, Shiue e Tseng (2011) propõem um mecanismo adaptativo de seleção de regras que atribui periodicamente diferentes regras de prioridade a vários centros de trabalho em um sistema de

manufatura flexível.

Também há pesquisas voltadas para encontrar a sequência ótima de tarefas. Weckman, Ganduri e Koonce (2008) dividem o JSP em uma série de problemas de classificação orientados para tarefas e usam cronogramas ótimos gerados por Algoritmo Genético, do inglês *Genetic Algorithm* (GA) para treinar a ANN. Zang et al. (2019) também dividem o JSP em vários subproblemas a serem resolvidos por uma rede híbrida com camadas totalmente conectadas e convolucionais em paralelo. Gupta, Majumder e Laha (2020) propõem uma abordagem para encontrar a posição relativa de tarefas em um cronograma ótimo em vez de desenvolver a sequência ótima diretamente. Outras aplicações de aprendizado supervisionado incluem o uso de Árvores de Decisão para aprender a estratégia a partir de cronogramas ótimos (OLAFSSON; LI, 2010; JUN; LEE; CHUN, 2019), e SVM como o seletor de regras adaptativas (SHIUE, 2009; PRIORE et al., 2010).

2.3 Agendamento Baseado em Aprendizagem por Reforço e Aprendizagem por Reforço Profunda

O problema de agendamento dinâmico pode ser formulado como um processo de controle estocástico discreto no tempo que se conforma à definição de Processo de Decisão de Markov, do inglês *Markov Decision Process* (MDP) (ZHANG; XIE; ROSE, 2017). Mais especificamente, uma unidade fabril que opera sob uma política reativa tomada por agentes de agendamento distribuídos pode ser modelada como um Processo de Decisão de Markov descentralizado (DEC-MDP) (BERNSTEIN et al., 2002). A natureza markoviana do problema de agendamento dinâmico torna o RL, uma ferramenta para tomada de decisão sequencial em MDP, adequado para resolver o problema de agendamento.

Estudos iniciais utilizam algoritmos de Aprendizagem por Reforço (RL) com representação tabular para realizar agendamento em toda a unidade fabril (WANG; USHER, 2005; AISSANI; BELDJILALI; TRENTESAUX, 2008). Nessa abordagem, o espaço de estados contém informações abstratas do sistema, e as ações correspondem à seleção de regras de prioridade a serem aplicadas às máquinas. Alternativamente, o problema de agendamento pode ser dividido em subproblemas menores, cada um abordado por um agente RL independente (WANG et al., 2021).

Comparados a outros algoritmos de aprendizado supervisionado desenvolvidos principalmente para tarefas de classificação, os algoritmos DRL podem criar um mapeamento direto da observação do ambiente para a ação; eles também têm a capacidade de lidar com dados complexos devido à utilização de rede neural artificial (ANN) como representação de valor ou política. Essas duas características tornaram o DRL uma abordagem promissora para o controle de produção em um sistema de manufatura intensivo em dados.

A Aprendizagem por Reforço Profunda (DRL) utiliza essas redes neurais profundas para aproximar as funções de valor ou políticas, permitindo lidar com espaços de estados e ações de alta dimensão, que seriam intratáveis com a representação tabular. Algoritmos de DRL têm sido incorporados à arquitetura de agendamento para melhorar o desempenho em unidades fabris complexas. Exemplos incluem o uso de DQN (LIN et al., 2019), *Deep Deterministic Policy Gradient* (DDPG) (LIU; CHANG; TSENG, 2020) e PPO (PARK et al., 2021).

Algoritmos DRL ligados a cada máquina ou centro de trabalho também podem ser tomadores de decisão distribuídos. Gabel e Riedmiller (2012) descrevem o JSP como um processo de decisão de Markov descentralizado com conjuntos de ações mutáveis e dependências de transição parcialmente ordenadas e propõem uma abordagem de gradiente de política para resolvê-lo. Uma abordagem semelhante pode ser encontrada em (LANG et al., 2020), onde dois tipos de agentes de agendamento treinados por DQN realizam sequenciamento de tarefas e seleção de máquinas independentemente, para gerenciar uma unidade fabril flexível.

Abordagens desenvolvidas para resolver problemas de agendamento de tamanho fixo são difíceis de generalizar para diferentes classes de tarefas. Também há muito poucos estudos de agendamento dinâmico que não restringem a especificação do problema para se aproximar de um ambiente prático. Wang (2020) propõe uma abordagem *Q-learning* descentralizada para selecionar regras de agendamento, para lidar com JSP com chegadas de *jobs* de forma dinâmica. Luo (2020) propõe uma abordagem de aprendizado duplo DQN centralizada para resolver FJSP com chegadas dinâmicas de tarefas, na qual a ação é semi-integrada para que os agentes selecionem regras de sequenciamento e seleção de máquinas acopladas.

Alguns estudos visam o agendamento em um contexto específico, tendo assim representações de estado e ação especializadas. Waschneck et al. (2018) propõem uma abordagem de DQN para agendar o movimento de lotes entre posições em uma instalação de manufatura de semicondutores. Qu, Wang e Jasperneite (2019) propõem uma abordagem ator-crítico descentralizada para alocar tarefas a servidores. Kuhnle, Röhrig e Lanza (2019) desenvolvem um sistema de despacho de pedidos autônomo com base em *Trust Region Policy Optimization* (TRPO) em uma instalação de produção de semicondutores. Shi et al. (2020) propõem uma abordagem de DQN para transferir tarefas em um sistema de manufatura discreto. Hubbs et al. (2020a) propõem uma abordagem ator-crítico profundo para o agendamento em um sistema de manufatura química com tempos de processamento não determinísticos. Malus, Kozjek et al. (2020) propõem uma abordagem multiagente para gerenciar robôs móveis autônomos em uma fábrica. Yang e Xu (2022) propõem uma abordagem *Asynchronous Methods for DRL* (A2C) para gerenciar um sistema de produção reconfigurável com chegada dinâmica de tarefas. Luo et al. (2022) propõem uma abordagem PPO para resolver o problema dinâmico de *job shop* restrito por múltiplos recursos, como máquina, ferramenta e trabalhador.

As tarefas também podem ser a entidade de tomada de decisões. Bouazza, Sallez e Beldjilali (2017) propõem uma abordagem com *Q-learning* que permite que produtos inteligentes

escolham uma seleção de máquinas e uma regra de sequenciamento para cada centro de trabalho e máquina que visitam. Baer et al. (2019) modelam o sistema de manufatura flexível como uma rede de Petri (ZHOU, 2012), na qual os agentes DQN anexados a produtos individuais decidem a rota para diferentes máquinas.

O agendamento dinâmico baseado em DRL ainda é uma área de pesquisa nova que está crescendo bastante em número de publicações. Esta pesquisa adota um versão multicritério mais realista, no qual o ambiente apresenta certo grau de incerteza, como *jobs* chegando de forma dinâmica, cancelamento de pedidos e estoques limitados. E o agente deve tomar decisões levando em consideração a carga atual do sistema, que pode impactar atrasos e perdas financeiras. Além disso, o agente deve agir levando em consideração prioridades em relação a três critérios de desempenho: lucro, sustentabilidade e variabilidade.Uma definição formal do problema é apresentada na próxima seção.

2.4 Considerações Finais

Este capítulo explorou diferentes abordagens para problemas de agendamento em ambientes dinâmicos de manufatura. Foram revisadas abordagens tradicionais, como regras de prioridade e meta-heurísticas, além de técnicas mais recentes baseadas em dados, como Programação Genética e aprendizado supervisionado, que buscam usar informações em tempo real para aprimorar as decisões. Por fim, foram discutidas abordagens baseadas em Aprendizagem por Reforço (RL), que se mostram promissoras para lidar com a complexidade e o dinamismo inerentes a esses problemas.

A RL se destaca em ambientes dinâmicos, sujeitos a eventos como chegadas de novos trabalhos, cancelamentos e mudanças nas condições do sistema. Algoritmos de RL adaptam suas estratégias com base na observação do ambiente e em experiências passadas, lidando com incertezas. Em manufatura, demonstram potencial para otimizar a eficiência, reduzir tempos de espera e aumentar a flexibilidade dos sistemas, sendo aplicados em sequenciamento de tarefas, alocação de recursos e tomada de decisões em tempo real. Neste trabalho, foca-se na DRL, que utiliza redes neurais profundas, em vez de abordagens tabulares como *Q-learning* e SARSA.

A *Aprendizagem por Reforço Profunda* (DRL) emerge como um subcampo da RL que utiliza redes neurais profundas para aproximar funções complexas, permitindo que os agentes lidem com ambientes com alta dimensionalidade de estados e ações. Ao aprender padrões complexos, a DRL se torna eficaz em jogos (AlphaGo, AlphaZero), robótica e, mais recentemente, em agendamento dinâmico, onde a adaptação a mudanças no ambiente é essencial. Os algoritmos DQN, PPO e PPO Recorrente, foco deste trabalho, exemplificam a DRL. **Doravante, por simplicidade, o termo Aprendizagem por Reforço (ou RL) será utilizado como sinônimo de DRL, exceto quando explicitamente indicado o contrário.**

Esta dissertação diferencia-se por propor um *framework* de DRL multicritério para

agendamento dinâmico em uma SOMN, considerando explicitamente critérios de desempenho econômico, sustentabilidade e variabilidade da produção. O objetivo é equilibrar esses critérios para gerar cronogramas otimizados e personalizados. O próximo capítulo apresenta os fundamentos teóricos da RL e da DRL, base para a metodologia proposta.

3 Fundamentação Teórica

O objetivo deste capítulo é fornecer uma introdução ao arcabouço teórico fundamental necessário para descrever o método de solução de aprendizado por reforço para o SOMN apresentado nesta dissertação e para ilustrar seu uso como uma ferramenta para a personalização em massa.

Primeiramente, a Seção 3.1 introduz o conceito de SOMN e sua importância no estudo de algoritmos de agendamento baseado em aprendizado, com características dinâmicas, para promover avanços no controle dos processos de produção. Em segundo lugar, a Seção 3.2 foca na divergência de Kullback-Leibler (KL), um parâmetro essencial do algoritmo de aprendizado por reforço usado neste trabalho. Em terceiro lugar, a Seção 3.3 introduz o MDP como o arcabouço matemático sobre o qual o aprendizado por reforço é construído. Por fim, a Seção 3.4 aborda os algoritmos fundamentais de aprendizado por reforço e os conceitos que são essenciais para este trabalho.

3.1 *Self-organizing manufacturing network*

Nesta seção, apresenta-se uma análise desse conceito emergente na indústria. Inicia-se definindo o que é sistemas de manufatura auto-organizado e destacando suas características fundamentais. Em seguida, discute-se como esse paradigma se originou, seu desenvolvimento ao longo do tempo e sua definição atual. Apresenta-se também uma comparação com outros paradigmas de sistemas de manufatura para destacar as vantagens e benefícios distintivos da abordagem auto-organizável. Ao longo desta seção, o objetivo é fornecer uma base sólida para a compreensão dos princípios e fundamentos subjacentes à rede de manufatura auto-organizável, preparando o terreno para a discussão mais detalhada nos próximos tópicos sobre sua aplicação na personalização em massa e na configuração de redes de manufatura.

O sistema de manufatura auto-organizado é um paradigma emergente que visa aprimorar a eficiência e a flexibilidade dos processos industriais. Originado a partir do desenvolvimento de abordagens como sistemas multiagentes e computação em nuvem, esse conceito possibilita a coordenação autônoma dos recursos de produção para otimizar o desempenho global do sistema (MAGANHA; SILVA; FERREIRA, 2018). A capacidade de auto-organização desses sistemas permite uma adaptação dinâmica às mudanças nas condições do mercado e nos requisitos de produção, tornando-os flexíveis e resilientes (PARK; TRAN; PARK, 2012).

A origem do sistema de manufatura auto-organizado remonta ao surgimento de tecnologias como a computação em nuvem e a Internet das Coisas (IoT), que possibilitaram a criação de ambientes de produção conectados e interoperáveis (MAGANHA; SILVA; FERREIRA, 2018).

Com o desenvolvimento dessas tecnologias, tornou-se viável a implementação de sistemas que fossem capazes de coordenar suas atividades de forma autônoma, sem a necessidade de intervenção humana constante (PARK; TRAN; PARK, 2012).

Ao comparar o sistema de manufatura auto-organizado com paradigmas tradicionais de sistemas de produção, como os sistemas centralizados e hierárquicos, destacam-se suas vantagens em termos de flexibilidade, adaptabilidade e escalabilidade (MAGANHA; SILVA; FERREIRA, 2018). Enquanto os sistemas tradicionais tendem a ser rígidos e inflexíveis, os sistemas auto-organizados são capazes de se ajustar dinamicamente às mudanças nas condições do ambiente de produção, garantindo uma resposta rápida e eficaz a perturbações e demandas não previstas.

Os fatores por trás da personalização em massa incluem a busca por produtos exclusivos e adaptados às preferências individuais dos consumidores, bem como avanços tecnológicos que permitem a produção eficiente e rentável de pequenas séries de produtos customizados (BORTOLINI; GALIZIA; MORA, 2018). A crescente disponibilidade de tecnologias como impressão 3D e manufatura aditiva está revolucionando a forma como os produtos são projetados, fabricados e distribuídos, abrindo novas oportunidades para a personalização em massa na indústria.

Os sistemas de manufatura auto-organizados desempenham um papel fundamental na realização da personalização em massa, permitindo uma produção ágil e flexível de produtos customizados. Ao contrário dos sistemas de produção convencionais, que são projetados para produzir grandes volumes de produtos padronizados, os *Self-Organizing Manufacturing Systems* (SOMS) são capazes de se adaptar dinamicamente às demandas do mercado, reconfigurando-se e ajustando-se rapidamente para atender às necessidades específicas dos clientes (BORTOLINI; GALIZIA; MORA, 2018). Essa capacidade de adaptação dinâmica é essencial para atender às demandas de um mercado cada vez mais diversificado e exigente.

Por fim, avança-se ao conceito de *self-organizing manufacturing network*, e sua importância na realização da personalização em massa e na criação de sistemas de produção eficientes e adaptáveis. Ao integrar os princípios da manufatura auto-organizada com as redes de produção distribuídas e interconectadas, esses sistemas são capazes de coordenar de forma eficiente as atividades de produção em uma ampla variedade de contextos industriais (QIN; LU, 2021). A capacidade de auto-organização dessas redes as torna flexíveis e adaptáveis, permitindo uma resposta rápida e eficaz a mudanças nas condições do mercado e nos requisitos de produção (ESTRADA-JIMENEZ et al., 2023). Assim, os sistemas de manufatura auto-organizados representam uma abordagem inovadora para a realização da personalização em massa na indústria, proporcionando uma vantagem competitiva significativa em um mercado global cada vez mais dinâmico e diversificado.

3.2 Divergência de Kullback-leibler

A divergência de Kullback-Leibler, introduzida por (KULLBACK; LEIBLER, 1951), é uma distância estatística utilizada para descrever a disparidade entre duas distribuições de probabilidade. Essa medida estatística é amplamente empregada em métodos de aprendizado por reforço para descrever a disparidade entre uma distribuição de probabilidade aproximada e a distribuição de probabilidade dos dados observados (KOCHENDERFER; WHEELER; WRAY, 2022). Portanto, este capítulo fornece uma breve introdução à divergência de *Kullback-Leibler*, um elemento essencial do algoritmo de aprendizado por reforço utilizado neste trabalho.

Seja P a distribuição de probabilidade discreta dos dados observados e Q o modelo aprendido para aproximar P, ambos definidos sobre o espaço amostral $\mathcal{X}$, então a divergência de KL é dada por:

$$D_{\mathrm{KL}}(P||Q) = \sum_{x \in \mathcal{X}} P(x) \log \frac{P(x)}{Q(x)}. \tag{3.1}$$

Por exemplo, se a probabilidade para um evento em Q é grande, enquanto a probabilidade para o mesmo evento em P é pequena, ou vice-versa, há uma divergência relativamente grande. Portanto, a divergência de KL pode ser usada para definir quão bom é um modelo aprendido.

A divergência de KL também pode descrever a disparidade de duas densidades de probabilidade p e q sobre a variável aleatória contínua x. Portanto, a divergência de KL de duas distribuições de probabilidade contínuas é dada por:

$$D_{\mathrm{KL}}(p||q) = \int p(x) \log \frac{p(x)}{q(x)} dx. \tag{3.2}$$

Inerentemente à natureza das probabilidades, a divergência de KL é sempre não negativa $D_{\mathrm{KL}} \geq 0$. Por fim, uma vez que a divergência de KL pode ser interpretada como a entropia cruzada para Q em relação a P, e a auto-entropia de P, ela é frequentemente referida como a entropia relativa de Q para P (JOYCE, 2011).

3.3 Processo de Decisão de Markov

O MDP é um *framework* matemático usado para formular problemas de decisão estocásticos de tempo discreto. Ele está intimamente relacionado ao aprendizado por reforço, que utiliza o arcabouço do MDP em sua metodologia de solução (SUTTON; BARTO, 2018). Mais tarde neste trabalho, o SOMN será formulado como um MDP para aproveitar técnicas de aprendizado por reforço. Uma propriedade chave do modelo MDP é a propriedade de Markov, que expressa que um estado depende apenas de estados anteriores e da ação tomada nesse estado. Essa propriedade garante que, dado o par estado-ação anterior, o estado seja independente de todos os outros estados (KOCHENDERFER; WHEELER; WRAY, 2022). Um MDP discreto no tempo e finito pode ser expresso por uma 4-tupla $(\mathcal{S}, \mathcal{A}, T, R)$, onde:

- $\mathcal{S}$ denota o espaço de estados, um conjunto de todos os estados. O estado do passo de tempo t, onde $t \in [0, T]$, é denotado como s_t.
- $\mathcal{A}$ denota o espaço de ações, um conjunto de todas as ações possíveis. A ação em um determinado passo de tempo t, onde $t \in [0, T]$, é denotada como a_t.
- T denota o modelo de transição de estado $T(s_{t+1}|s_t, a_t)$. O modelo define a probabilidade de transição de um estado s_t para s_{t+1}, tomando a ação a_t.
- R denota a função de recompensa. A transição do estado s_t para o próximo estado s_{t+1} devido à ação a_t, recebe a recompensa para o próximo passo de tempo $r_{t+1} \doteq R(s_t, a_t, s_{t+1})$. A imagem da função R é denotada por $\mathcal{R}$.

Em MDP, os espaços de ação e estado podem ser finitos ou infinitos. O espaço de estado fornece uma representação abstrata das situações modeladas. Por exemplo, no contexto do SOMN em um determinado passo de tempo t, o estado pode descrever as operações disponíveis e o status da unidade fabril, indicando o nível de carga, matéria-prima ou se está com o estoque cheio ou não. Além disso, uma ação pode se referir a uma operação que pode ser despachada em uma máquina. Esse processo de tomada de decisão envolve decidir qual operação agendar em qual máquina com base nas informações disponíveis sobre o estado atual. O processo de decisão pode ser complexo, pois pode haver várias ações a considerar.

O modelo de transição T pode ser probabilístico, onde a transição de um estado para outro é modelada usando uma distribuição de probabilidade (SUTTON; BARTO, 2018). Por exemplo, quando um robô opera em um ambiente parcialmente observável, a transição de um estado para outro geralmente é modelada por uma distribuição de probabilidade. No caso do SOMN, a probabilidade de transição de um estado para outro dada a ação a é ou zero, indicando que a transição é impossível, ou um, se a transição ocorre após executar a ação a.

3.3.1 A Interação Agente-Ambiente

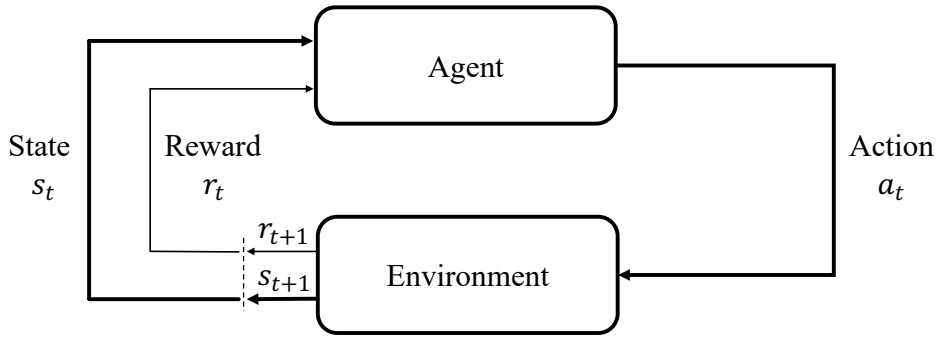
Muitos problemas de decisão podem ser modelados como uma interação agente-ambiente, onde o agente deve tomar decisões na forma de ações em um ambiente. Por exemplo, o agente pode ser um trabalhador em uma fábrica de manufatura que precisa decidir qual operação despachar em qual máquina. O MDP é um conceito direto para modelar interações agente-ambiente, como mostrado na Figura 1. Aqui, em cada passo discreto de tempo, representado por t, o agente recebe uma representação do estado do ambiente $s_t \in \mathcal{S}$ e decide com base nisso, dada o espaço de ação $\mathcal{A}(s)$, qual ação seleciona. Como consequência de sua ação, o ambiente transita do estado s_t para o próximo estado s_{t+1} e o agente recebe uma recompensa numérica por sua ação, denotada por r_{t+1} (KOCHENDERFER; WHEELER; WRAY, 2022; SUTTON; BARTO, 2018). Além disso, essa interação agente-ambiente também pode ser interpretada como

uma trajetória:

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, a_2, r_3, \ldots \tag{3.3}$$

Normalmente, um agente interage com seu ambiente para alcançar um objetivo específico. A próxima seção introduz esse objetivo no contexto da interface agente-ambiente, começando com uma descrição da função de recompensa e como ela informa o objetivo do agente.

Figura 1 – Interface agente-ambiente de um MDP.



Fonte: (SUTTON; BARTO, 2018, p. 48).

3.3.2 A Função de Recompensa e o Retorno das Recompensas

A função de recompensa R é responsável por definir a recompensa associada à transição entre estados ao tomar uma ação específica. Para satisfazer os critérios de um MDP, a função de recompensa deve depender tanto do estado atual s quanto do estado subsequente s', bem como da ação a tomada no estado s. Conforme observado em Ng, Harada e Russell (1999), a função de recompensa geralmente é uma função limitada com valores reais $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$.

Em um MDP, o objetivo do agente é tipicamente maximizar o retorno total das recompensas que recebe. Isso é denotado por G_t, que representa o retorno esperado das recompensas futuras em um determinado passo de tempo t. O retorno esperado G_t pode ser calculado como a soma de todas as recompensas coletadas após começar no passo de tempo t e terminar no passo de tempo final T. Um passo de tempo final só pode ser definido em problemas não contínuos, onde o problema começa no passo de tempo t e termina em um passo de tempo final T. Por exemplo, o passo de tempo final pode ser a situação em que o agente alcança seu objetivo. Assim, como descrito no trabalho de Sutton e Barto (2018), o retorno esperado G_t para problemas não contínuos é dado por:

$$G_t = r_{t+1} + r_{t+2} + \cdots + r_T. \tag{3.4}$$

Problemas contínuos não possuem um estado final, tornando impossível usar a equação 3.4, já que a soma das recompensas cresceria infinitamente. Um fator de desconto $\gamma \in [0, 1)$ é

usado para resolver esse problema. O fator de desconto reduz o impacto dos retornos futuros na recompensa total. Conforme descrito no trabalho de (SUTTON; BARTO, 2018), o retorno descontado das recompensas é dado por:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{3.5}$$

Quando γ está próximo de zero, G_t é geralmente dominado pela recompensa imediata. Se, por outro lado, γ está próximo de um, a influência de recompensas subsequentes em G_t aumenta.

3.3.3 Políticas e Funções de Valor

O comportamento do agente é definido pela sua política π, que especifica a probabilidade de selecionar cada ação em um determinado estado. Especificamente, a função $\pi(a|s)$ denota a probabilidade de selecionar a ação a quando o agente está no estado s, no passo de tempo t, seguindo a política π. Em outras palavras, a política define uma distribuição de probabilidade sobre o conjunto de ações possíveis $\mathcal{A}(s)$ para cada estado s no espaço de estados $\mathcal{S}$ (SUTTON; BARTO, 2018).

As funções de valor desempenham um papel crucial na busca por uma política ótima em MDPs e aprendizado por reforço. Essas funções de valor medem a qualidade de um estado ou de uma ação. A função de valor do estado $V_\pi(s)$ descreve o retorno futuro esperado G_t de estar no estado $s \in \mathcal{S}$ e seguir a política π. Ela indica o valor de longo prazo de um estado, levando em consideração as recompensas esperadas que podem ser obtidas a partir desse estado sob a política fornecida. Assim, a função de valor do estado é definida da seguinte forma:

$$V_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| S_t = s\right], \ \forall s \in \mathcal{S}, \tag{3.6}$$

onde $\mathbb{E}_\pi[\cdot]$ denota o valor esperado de uma variável aleatória (SUTTON; BARTO, 2018). A segunda função de valor usada em MDPs e aprendizado por reforço é a função de valor da ação, denotada por $Q_\pi(s, a)$, ou simplesmente a função Q. Essa função mede a qualidade de tomar uma ação específica a no estado s enquanto segue a política π. Em outras palavras, a função Q fornece uma estimativa do retorno esperado se o agente selecionar a ação a no estado s e, em seguida, seguir a política π para as etapas restantes. De acordo com o trabalho de Sutton e Barto (2018), a função Q é definida da seguinte forma:

$$Q_\pi(s, a) \doteq \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \middle| S_t = s, A_t = a\right]. \tag{3.7}$$

Ambas as equações também podem ser expressas como uma função do estado sucessor s' ou do par estado-ação sucessor. A equação de Bellman, também conhecida como iteração de valor, expressa essa propriedade. Dada a equação 3.6 e seguindo o trabalho de Sutton e Barto

(2018), a equação de Bellman para a função de valor do estado é dada por:

$$\begin{aligned} V_\pi(s) &\doteq \mathbb{E}_\pi[G_t|S_t=s] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1}\middle|S_t=s\right] \\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\Big[r+\gamma\mathbb{E}_\pi[G_{t+1}|S_{t+1}=s']\Big] \\ &= \sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)\Big[r+\gamma V_\pi(s')\Big], \end{aligned} \tag{3.8}$$

onde r denota uma recompensa numérica que pode ser recebida no estado s', e a denota uma ação do espaço de ações $\mathcal{A}(s)$. Além disso, $p(s',r|s,a)$ representa a probabilidade de transição do estado s para o estado s' ao tomar a ação a e subsequentemente receber a recompensa r. Em essência, p é a distribuição de probabilidade do modelo de transição $\mathcal{T}$ introduzido na Seção 3.3, que também leva em consideração a recompensa recebida. A equação de Bellman também vale para a função Q:

$$Q_\pi(s,a) = \sum_{s',r} p(s',r|s,a)\Big[r+\gamma V_\pi(s')\Big]. \tag{3.9}$$

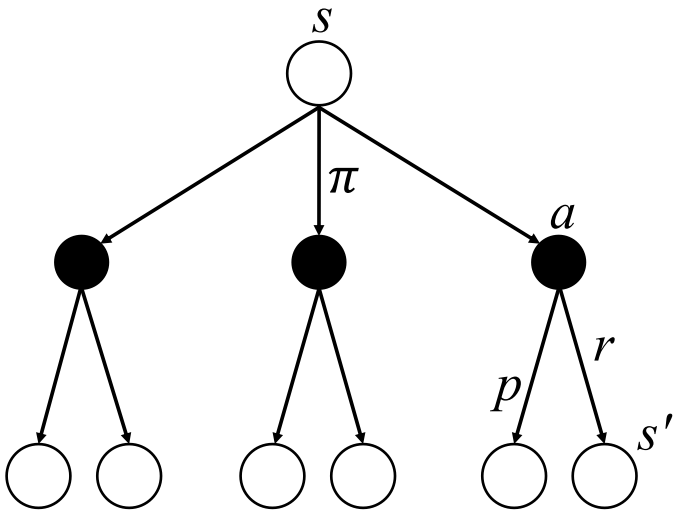
A Figura 2 ajuda a ilustrar como a equação de Bellman funciona. A figura representa um estado s e seus seis possíveis estados sucessores. Os círculos abertos representam estados individuais, e os círculos fechados representam pares estado-ação. No centro da figura está o nó raiz s, que tem três ações disponíveis, cada uma com sua própria probabilidade, conforme definido pela política π. Uma vez que uma ação é escolhida, o ambiente responde com uma recompensa r e leva o agente a um dos possíveis estados sucessores s'. A figura assume um ambiente estocástico, o que significa que se o agente selecionar uma ação a no estado s, a transição para um estado sucessor é determinada pela distribuição de probabilidade p. No entanto, em um ambiente determinístico, selecionar a ação a sempre leva ao mesmo estado sucessor. Assim, há apenas um estado sucessor possível ao selecionar a ação a no estado s. Consequentemente, a transição de um estado precedente s para um estado sucessor específico s', dada a ação tomada, sempre tem uma probabilidade de zero ou um em um ambiente determinístico.

O principal objetivo da figura é mostrar que o valor do estado s' depende do valor de seu estado antecessor, bem como da política π, das ações a que podem ser tomadas e das recompensas r associadas a essas ações.

3.3.4 Políticas Ótimas e Funções de Valor Ótimo

Para acumular o máximo de recompensa possível ao longo dos estados futuros, os métodos de aprendizado por reforço e os MDPs, em geral, buscam encontrar uma política ótima π_*. A qualidade de uma política π é definida pelo seu retorno esperado para todos os estados $s \in \mathcal{S}$. Em outras palavras, a política π é considerada melhor ou igual em comparação com a

Figura 2 – Diagrama de backup que expressa a relação entre s e s'.



Fonte: (SUTTON; BARTO, 2018, p. 59)

política π' se seguir a política π resultar em um retorno esperado acumulado maior ou igual ao retorno esperado acumulado de π' para todos os estados. Formalmente, isso significa que, se $v_\pi(s) \geq v_{\pi'}(s),\ \forall s \in \mathcal{S}$, então $\pi \geq \pi'$ (SUTTON; BARTO, 2018).

Finalmente, uma política ótima π_* deve ser melhor ou igual a todas as outras políticas para todos os estados. Todas as políticas ótimas compartilham a mesma função de valor de estado V_*. Assim, a função de valor de estado ótima V_* é dada por:

$$V_*(s) \doteq \max_\pi V_\pi(s),\ \forall s \in \mathcal{S}. \tag{3.10}$$

Além disso, de acordo com o trabalho de Sutton e Barto (2018), a função Q ótima Q_* é dada por:

$$Q_*(s) \doteq \max_\pi Q_\pi(s, a),\ \forall s \in \mathcal{S},\ \forall a \in \mathcal{A}. \tag{3.11}$$

Sempre escolher a melhor ação da função Q ótima é intuitivamente semelhante a seguir a função de valor de estado ótima. Portanto, a relação entre V_* e Q_* é dada por:

$$V_*(s) = \max_{a \in \mathcal{A}(s)} Q_{\pi*}(s, a). \tag{3.12}$$

O uso da equação de Bellman introduzida na Seção 3.3.3 pode levar a uma função de valor ótima. No entanto, o conhecimento completo do ambiente, incluindo a distribuição de probabilidade de transição p e a política π, é necessário para calcular a equação de Bellman, conforme descrito no diagrama de backup na Figura 2. O próximo capítulo introduz métodos de aprendizado por reforço que lidam com esse problema amostrando experiências no ambiente e aproximando a função de valor do estado precedente. Portanto, a equação de Bellman é um aspecto importante da maioria dos algoritmos de aprendizado por reforço (SUTTON; BARTO, 2018).

3.4 Aprendizagem por Reforço

Além da aprendizagem supervisionada e não supervisionada, o aprendizado por reforço é um dos três paradigmas de aprendizado de máquina. Ele descreve uma classe de problemas computacionais, bem como métodos de solução no domínio do aprendizado direcionado por objetivos e tomada de decisões. O aprendizado por reforço geralmente é baseado no *framework* de um MDP, onde um agente aprende a otimizar seu comportamento interagindo com o ambiente e recebendo *feedback* na forma de recompensas (SUTTON; BARTO, 2018). O objetivo do agente é aprender uma política que maximize a recompensa cumulativa que recebe ao longo do tempo. Sutton e Barto (2018, p.1) resume o aspecto fundamental do aprendizado por reforço em uma frase simples:

> O aprendizado por reforço é aprender o que fazer — como mapear situações para ações — de forma a maximizar um sinal numérico de recompensa.

Como os métodos de aprendizado por reforço são baseados no *framework* de um MDP, os elementos-chave do aprendizado por reforço incluem um agente, o ambiente, uma política, uma função de valor e, opcionalmente, um modelo do ambiente. O modelo do ambiente é um modelo estocástico que prevê o comportamento do ambiente, permitindo que o agente infira como o ambiente reagirá a uma ação específica. Por exemplo, o modelo pode prever a recompensa imediata de uma ação no estado atual. Usar um modelo para prever eventos futuros é conhecido como planejamento, e os métodos de aprendizado por reforço que dependem de planejamento são chamados de métodos baseados em modelo. Em contraste, os métodos sem modelo aprendem por tentativa e erro sem prever eventos futuros. Nas seções seguintes, três métodos de aprendizado por reforço sem modelo são apresentados, sendo que cada método se baseia no anterior.

3.4.1 Aprendizagem por Diferença Temporal

Os métodos de aprendizado por Diferença Temporal, do inglês *Temporal-Difference* (TD) utilizam a equação de Bellman e técnicas de *bootstrapping* para estimar a função de valor do estado. Com os métodos de TD, a função de valor do estado estimada é atualizada usando estimativas aprendidas anteriormente (SUTTON; BARTO, 2018). Muitos métodos de aprendizado por reforço são construídos com base no aprendizado por TD.

Por exemplo, um método de TD simples utiliza o valor do estado sucessor $V(s_{t+1})$ e a recompensa r_t recebida no passo de tempo $t+1$ para atualizar $V(s_t)$. Portanto, seguindo o trabalho de (SUTTON; BARTO, 2018), um método de TD simples é dado por:

$$V(s_t) \leftarrow V(s_t) + \alpha \left[r_{t+1} + \gamma V(s_{t+1}) - V(s_t) \right]. \tag{3.13}$$

Aqui, o termo $r_{t+1} + \gamma V(s_{t+1})$ é chamado de alvo de TD. Além disso, o termo entre colchetes é frequentemente chamado de erro de TD δ_t e aparece em diferentes formas em muitos métodos de aprendizado por reforço (SUTTON; BARTO, 2018):

$$\delta_t \doteq r_t + \gamma V(s_{t+1}) - V(s_t). \tag{3.14}$$

Finalmente, os métodos de TD incluem um aspecto essencial do aprendizado por reforço: aprender por reforço. Em outras palavras, isso significa atualizar uma estimativa ao receber uma recompensa dependendo do estado atual e da ação selecionada (SUTTON; BARTO, 2018). No entanto, para espaços de estados grandes, os métodos de TD simples sofrem com o alto uso de memória, pois todas as estimativas dos estados precisam ser armazenadas em cache.

3.4.2 Deep Q Network

Deep Q Network (DQN) é um algoritmo de aprendizado por reforço livre de modelo que aproxima a função de valor de ação ótima, chamada de função Q. Ao contrário de um algoritmo TD básico, o algoritmo DQN é adequado para espaços de estados grandes, pois aproxima a função de valor de ação ótima em vez de procurar a verdadeira função Q ou função de valor de estado ótimo.

Mnih et al. (2015a) introduziu a ideia de usar uma rede neural para aproximar a função Q ótima, conhecida como *Deep Q Network* (DQN). Essa abordagem alcançou resultados excepcionais ao jogar jogos Atari (BELLEMARE et al., 2013), que são amplamente utilizados como referências para algoritmos de aprendizado por reforço.

O vetor de entrada do DQN é uma representação vetorial do estado s_t, enquanto o vetor de saída fornece estimativas dos valores Q para esse estado. Como cada ação no estado s_t tem seu próprio valor Q $Q(s_t, a_t)$, o tamanho do vetor de saída corresponde ao tamanho do espaço de ação $|\mathcal{A}(s_t)|$. Para aprender com as interações com o ambiente, Mnih et al. (2015a) introduziram uma nova função de perda para atualizar o DQN. Dada a iteração i, os parâmetros correspondentes do DQN θ_i, o fator de desconto γ e a recompensa amostrada r_{t+1}, a função de perda é dada por:

$$L_i(\theta_i) = \mathbb{E}_{s,a,r,s'}\left[\left(r_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_i^-) - Q(s_t, a_t; \theta_i)\right)^2\right]. \tag{3.15}$$

Os parâmetros do DQN são atualizados usando retro-propagação com o gradiente da função de perda $\nabla_{\theta_i} L_i(\theta_i)$. A função de perda (ver Equação 3.15) é uma média de múltiplas interações amostradas do ambiente no mesmo passo de tempo, denotado por $\mathbb{E}_{s,a,r,s'}$. Isso melhora a eficiência do algoritmo, já que atualizar os parâmetros θ do DQN é computacionalmente caro. Atualizá-lo uma vez com base em uma média de múltiplas amostras é mais eficiente do que atualizá-lo frequentemente com uma única experiência. Essa técnica é comumente usada em aprendizado de máquina e é referida como amostragem de mini-lote.

O Algoritmo 1 ilustra a implementação do método DQN. Aqui, uma técnica chamada *replay* de experiência é usada para armazenar interações entre o agente e o ambiente na chamada memória de *replay* para reutilização em iterações futuras. Essas amostras são então usadas para calcular a função de perda média em mini-lotes, conforme explicado acima, e são denotadas pelo índice j.

A linha sete do Algoritmo 1 mostra que o método DQN usa uma política ϵ-greedy para gerar amostras que são então armazenadas na memória de *replay*. Mais especificamente, durante a seleção de ação em DQN, uma ação aleatória é escolhida com probabilidade ϵ, e com probabilidade $1-\epsilon$, a ação é selecionada com base na política aprendida. Geralmente, algoritmos de aprendizado por reforço que não seguem sempre a política aprendida para selecionar uma ação são chamados de algoritmos *off-policy*. Por outro lado, algoritmos que amostram ações unicamente de acordo com a política aprendida são chamados de *on-policy*.

Algoritmo 1 *Deep Q Network* (DQN) com Replay de Experiência seguindo o trabalho de Mnih et al. (2015a)

1: Inicialize a memória de replay $\mathcal{D}$ com capacidade N
2: Inicialize a função de valor de ação Q_θ com pesos aleatórios θ
3: Inicialize a função de valor de ação alvo $Q_{\hat{\theta}}$ com pesos $\hat{\theta} = \theta$
4: **para cada** episódio $\in \{1, 2, \ldots, M\}$ **faça**
5: amostrar o estado inicial s_0 a partir da distribuição inicial $\sim \mathcal{I}_s$
6: **para cada** t $\in \{1, 2, \ldots, T\}$ **faça**
7: Com probabilidade ϵ, selecione uma ação aleatória a_t
caso contrário, selecione $a_t = \arg\max_a Q(s_t, a; \theta)$
8: Execute a_t no ambiente e observe r_{t+1} e s_{t+1}
9: Armazene $s_t, a_t, r_{t+1}, s_{t+1}$ em $\mathcal{D}$
10: Amostrar um minibatch aleatório $s_j, a_j, r_{j+1}, s_{j+1}$ de $\mathcal{D}$
11: Defina $y_j = \begin{cases} r_{j+1} & \text{para } s_{j+1} \text{ terminal} \\ r_{j+1} + \gamma \max_{a_{j+1}} Q_{\hat{\theta}}(s_{j+1}, a_{j+1}), & \text{para } s_{j+1} \text{ não terminal} \end{cases}$
12: Realize um passo de descida do gradiente em $(y_j - Q_\theta(s_j, a_j))^2$ em relação a θ
13: A cada C passos, redefina $Q_{\hat{\theta}} = Q_\theta$
14: **fim para**
15: **fim para**

3.4.3 Métodos de Gradiente de Política

Os métodos de política de gradiente diferem dos algoritmos baseados em valor, como DQN e aprendizado TD, pois eles aproximam diretamente uma política ótima em vez de uma função de valor ótima. Tipicamente, esses métodos utilizam uma rede neural para aproximar a política (SUTTON; BARTO, 2018). Dado os parâmetros θ da rede neural, a política é dada por:

$$\pi(a|s, \theta) = Pr(a|s, \theta), \tag{3.16}$$

Assim, a política é a probabilidade de escolher uma ação dado o estado s e os parâmetros da rede neural θ. Além disso, a função de perda da rede neural $L(\theta)$ expressa o objetivo do problema de aprendizado por reforço: maximizar o retorno total de recompensas G_T. O algoritmo

REINFORCE, introduzido por Williams (1992), utiliza G_T para expressar a função de perda. No entanto, calcular G_T é apenas possível realizando uma simulação completa de um episódio, o que envolve simular um episódio até o estado final, como mostrado na Equação 3.4. Uma simulação completa pode levar a alta variância e custos computacionais para espaços de estados grandes. Como resultado, métodos de gradiente de política de ponta utilizam uma função de vantagem aprendida $\hat{A}_t$ para estimar o retorno total de recompensas G_T em vez disso. A função de vantagem é uma estimativa do retorno total de recompensas, similar ao erro de TD descrito na equação 3.14, e assim também pode ser vista como uma estimativa da função de valor do estado. Redes neurais são comumente usadas para aproximar a função de vantagem (SCHULMAN et al., 2018). A função de perda para métodos de gradiente de política, conforme proposto por Schulman et al. (2017a), é dada por:

$$L(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t|s_t) \hat{A}_t \right]. \tag{3.17}$$

Além disso, o gradiente $\nabla L(\theta)$, posteriormente denotado por $\hat{g}$, usado para atualizar a política é dado por:

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t \right]. \tag{3.18}$$

Já que o valor esperado é uma média empírica sobre um lote finito de amostras, ele é um estimador para o verdadeiro valor esperado denotado por $\hat{\mathbb{E}}_t$. Eventualmente, o gradiente $\hat{g}$ é inserido em uma ascensão de gradiente estocástica para atualizar o vetor de parâmetros θ da rede neural. Assim, a atualização dos parâmetros θ_{t+1} é dada por

$$\theta_{t+1} = \theta_t + \alpha \hat{g}, \tag{3.19}$$

onde $\alpha > 0$ é o fator de passo da ascensão do gradiente (SUTTON; BARTO, 2018).

Métodos ator-crítico são algoritmos de gradiente de política que usam uma função de vantagem aprendida para expressar a função de perda, conforme ilustrado na Equação 3.17. Neste método, o "ator"atualiza a política usando a abordagem de gradiente de política, enquanto o "crítico"utiliza a função de vantagem aprendida como um objetivo (MNIH et al., 2016).

No entanto, como os métodos ator-crítico usam uma função de vantagem estimada na função de perda, isso introduz um viés que pode impedir o algoritmo de convergir ou levar a soluções sub-ótimas (SCHULMAN et al., 2018; SCHULMAN et al., 2017a). Esse viés surge devido à dependência da estimativa da função de vantagem, o que pode resultar em um gradiente impreciso da função de perda.

Na próxima seção, é apresentado um algoritmo de gradiente de política que aborda esses problemas de convergência.

3.4.4 *Proximal Policy Optimization*

Conforme discutido na seção anterior (Seção 3.4.3), os métodos de gradiente de política podem enfrentar dois problemas: alta variância e esforço computacional ao empregar o retorno

total de recompensas como função de perda ou viés ao empregar uma função de vantagem aprendida. Para lidar com esses desafios, Schulman et al. (2017a) introduziram um método chamado *Proximal Policy Optimization* (PPO). Esta abordagem ainda utiliza uma função de vantagem aprendida, mas supera o viés associado a ela ao implementar uma nova função de perda, uma função de vantagem especializada e opcionalmente uma penalidade KL adaptativa.

A abordagem PPO utiliza a seguinte função de perda:

$$\begin{aligned} L^{CLIP}(\theta) &= \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}\hat{A}_t, \text{clip}(\epsilon, \hat{A}_t)\right) \\ \text{clip}(\epsilon, \hat{A}_t) &= \begin{cases} (1+\epsilon)\hat{A}_t, & \text{se } \hat{A}_t \geq 0 \\ (1-\epsilon)\hat{A}_t, & \text{se } \hat{A}_t < 0 \end{cases} \end{aligned} \tag{3.20}$$

onde ϵ é um hiper-parâmetro, frequentemente chamado de parâmetro de corte, que controla o tamanho do passo da política antiga θ_{old} para a nova política θ. Além disso, a função $clip(\epsilon, \hat{A})$ é chamada de termo de corte na publicação original de Schulman et al. (2017a). A nova função de perda estabiliza o processo de otimização limitando a mudança da função de perda. Isso é alcançado limitando a função de perda ao mínimo do termo de corte e do termo não cortado. Schulman et al. (2017a) demonstra que o uso da nova função de perda ajudou o algoritmo PPO a ter, em média, um desempenho melhor do que outros métodos de gradiente de política de ponta no conjunto de dados de *benchmark* clássico do Atari introduzido pelo trabalho de Bellemare et al. (2013).

Além da nova função de perda, o algoritmo PPO incorpora duas outras melhorias em comparação com métodos de gradiente de política convencionais. Primeiro, Schulman et al. (2017a) introduz uma penalidade adaptativa de KL que pode ser adicionada à função de perda L^{CLIP} conforme definido na equação 3.20. Essencialmente, a penalidade adaptativa de KL tem como objetivo garantir que a divergência de KL (consulte a Seção 3.2) atinja um valor alvo, denominado alvo de KL, em cada atualização de política. Para alcançar esse objetivo, a função de perda é ajustada com uma penalidade adaptativa dependente da divergência de KL entre $\pi_\theta(a_t|s_t)$ e $\pi_\theta(a_t|s_t)$, que é subtraída do gradiente da função de perda conforme dado pela equação 3.18.

Segundo, o algoritmo PPO utiliza a função de vantagem introduzida por Schulman et al. (2018), chamada *General Advantage Estimate* (GAE). Esta função de vantagem utiliza uma estimativa aprendida da função de valor do estado V e é dada por

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{3.21}$$

$$\text{onde } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)). \tag{3.22}$$

Aqui, os hiper-parâmetros γ e λ controlam o fator de desconto e o viés da função de vantagem, respectivamente. Além disso, T é um hiper-parâmetro que representa o número de passos de tempo usados para estimar a função de vantagem, sendo T menor que o comprimento do episódio completo. A função de vantagem pode ser aprendida usando uma abordagem semelhante ao DQN, o que torna o PPO um método ator-crítico (SCHULMAN et al., 2018).

A implementação do método PPO é resumida no Algoritmo 2. O algoritmo utiliza N atores em paralelo, que interagem com o ambiente e coletam amostras por T passos de tempo. A natureza paralela do algoritmo o torna adequado para computação paralela. O algoritmo de otimização no PPO é tipicamente o Adam (KINGMA; BA, 2014). O algoritmo é executado por E episódios para garantir a convergência, onde o valor de E deve ser definido suficientemente alto para garantir a convergência.

Algoritmo 2 Algoritmo PPO introduzido por Schulman et al. (2017a).

para cada iteração $\in \{1, 2, \ldots, E\}$ **faça**
    para cada ator $\in \{1, 2, \ldots, N\}$ **faça**
        Executar política $\pi_{\theta_{old}}$ no ambiente por T passos de tempo
        Calcular estimativas de vantagem $\hat{A}_1, \ldots, \hat{A}_T$
    fim para
    Otimizar L em relação a θ, com K épocas e tamanho de minibatch $M \leq NT$
    $\theta_{old} \leftarrow \theta$
fim para

3.5 Considerações Finais

Ao longo do Capítulo de Fundamentação Teórica, exploram-se os fundamentos do aprendizado por reforço e apresenta-se uma visão geral dos principais algoritmos utilizados nessa área. Inicia-se discutindo os Processos de Decisão Markovianos (MDPs), que são fundamentais para modelar problemas de decisão sequencial sob incerteza, e introduz-se a equação de Bellman, que descreve a relação entre a função de valor de um estado e as recompensas futuras esperadas. Esta discussão proporcionou uma base sólida para compreender os princípios subjacentes ao aprendizado por reforço.

Em seguida, nos aprofunda-se nas principais classes de algoritmos de aprendizado por reforço, começando com os métodos baseados em Aprendizagem por Diferença Temporal (TD), que atualizam iterativamente estimativas da função de valor diretamente dos dados de amostra. Explora-se então o *Deep Q Network* (DQN), uma abordagem que combina aprendizado por reforço com redes neurais profundas para lidar com espaços de estados de alta dimensionalidade. Por fim, discute-se os Métodos de Gradiente de Política, que buscam otimizar diretamente a política de ação, culminando em uma compreensão abrangente das diferentes abordagens de aprendizado por reforço. Para cada classe de algoritmos, provê-se uma explicação detalhada dos conceitos fundamentais, apresenta-se um algoritmo representativo e discute-se suas características e desafios. Essa abordagem permitiu uma compreensão abrangente das técnicas mais relevantes empregadas no aprendizado por reforço.

Finalmente, concentra-se no algoritmo Proximal Policy Optimization (PPO), uma abordagem avançada que supera desafios significativos encontrados em métodos de gradiente de política convencionais. Exploram-se os componentes-chave do PPO, incluindo a função de perda clipada, a penalidade adaptativa de KL e a função de vantagem GAE, destacando sua eficácia e robustez em aplicações práticas.

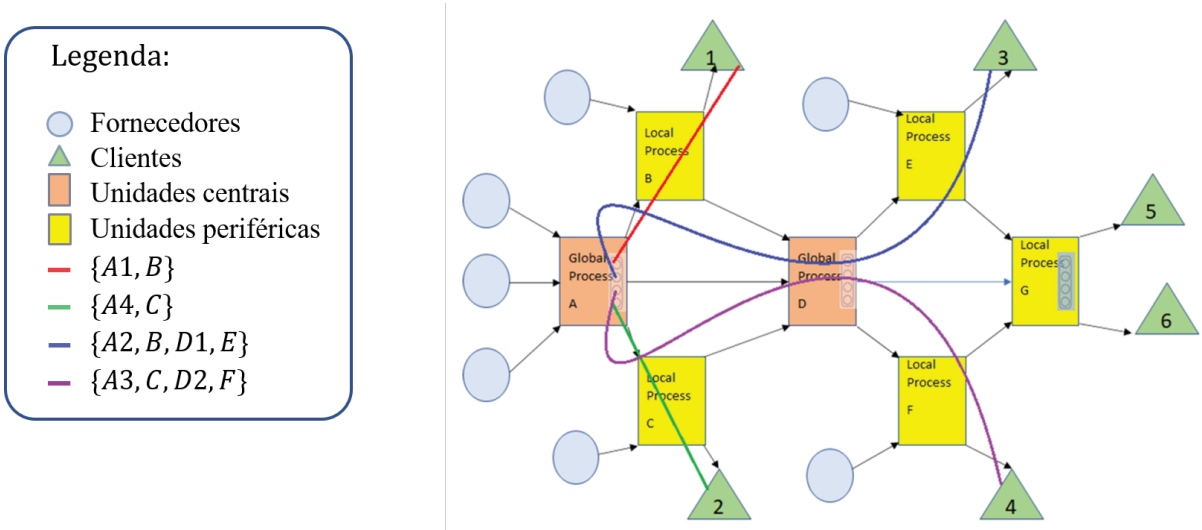
4 Modelagem do Ambiente de Produção

4.1 Definição do Escopo de Modelagem

A modelagem considera um ambiente de produção flexível composto por unidades fabris autônomas em rede que negociam quantidades de matéria-prima e prazos de entrega para atender às demandas de produtos personalizados. A rede de unidades fabris é um sistema complexo com pelo menos duas funções específicas: unidades periféricas e centrais *(backbone)*. As unidades de *backbone* são responsáveis por produzir elementos estruturais de interesse para muitos clientes (estágio de produção em massa). Unidades de *backbone*, em geral, guardam proximidade com grandes *hubs* de distribuição com facilidades para alcançar as demais unidades da rede.

Por outro lado, uma unidade periférica é responsável por imprimir recursos específicos e regionais (etapa personalizada). A localização de unidades periféricas perto de mercados de clientes e cadeias de suprimentos regionais também é estratégica. De fato, ocorrem fluxos de produtos semiacabados, aos quais são agregados valores a partir de fornecedores locais.

Figura 3 – Rede de manufatura: duas unidades dedicadas às funcionalidades estruturais (*backbone*), cinco unidades que lidam com *hubs* de clientes (triângulos) e fornecedores (círculos). Fluxos de produção com sequências de produtos 2-montados {A1, B}, {A4, C} e 4-montados {A2, B, D1, E}, {A3, C, D2, F}. Recorte da rede com uma unidade periférica e o fluxo de produção local.



Fonte: (O autor, 2024)

As unidades da rede de manufatura contribuem para o objetivo global de otimizar o lucro,

enquanto equilibram critérios de variabilidade e sustentabilidade. A abordagem multicritério visa capturar a complexidade e a dinâmica do ambiente de produção, permitindo que o sistema responda de forma adaptativa às mudanças nas condições de mercado e às necessidades dos clientes. A modelagem também leva em conta as interações entre componentes internos e externos, incluindo fornecedores, clientes e outras partes interessadas.

As preocupações com a variabilidade são introduzidas por características específicas e regionais agregadas às estruturais. Assim podemos conceber um produto final como uma sequência de unidades fabris por onde passa. Na Figura 3, o contexto geral da rede de manufatura é representado como um conjunto de unidades conectadas topologicamente. Cada linha colorida significa uma sequência de unidades responsáveis por agregar um conjunto de características específicas e introduzir variedade de produtos (personalização). Todas as linhas começam em uma unidade de fabricação de *backbone* (quadrado laranja) e terminam em um *hub* de distribuição para os clientes (triângulo verde). Por exemplo, os produtos *P41* e *P42* podem ser montados (ou representados) pelas seguintes sequências, $\{A2, B, D1, E\}$ e $\{A3, C, D2, F\}$ respectivamente.

Embora o foco deste trabalho seja desenvolver um simulador de uma única unidade fabril (**Figura 4**), é importante destacar que essa unidade está concebida para ser parte integrante de uma rede de manufatura mais ampla. A simulação de uma única unidade permite compreender o processo de decisão baseado em aprendizado por reforço, e estabelecer conexões com processos mais abrangentes de decisão por meio de compartilhamento de informações.

4.2 Exclusão de Escopo

A colocação de unidades não é o objetivo deste trabalho, mas apenas para decidir os fluxos da cadeia de abastecimento e entrega de produtos. A unidade simulada é periférica específica (Unidade B, por exemplo), pois tem que lidar com fornecedores e clientes regionais, ou seja, seu ecossistema é mais heterogêneo, como mostra a Figura 3.

Para simplificar, nenhuma competição predatória entre as unidades fabris é permitida. Competição é um efeito colateral que pode ser hipoteticamente evitado ao não permitir negociação de preços e contenção de matéria-prima, evitando-se assim que os fornecedores escolham quais demandas suprir. Assim, o único fator a ser considerado é o tempo para fornecer matérias-primas ou produtos. Como os negócios podem ser acordados, as cadeias de decisão podem conectar ou desconectar fluxos produzindo apenas produtos com boa relação custo-benefício no tempo.

4.3 Componentes Externos e Internos

O sistema de manufatura em rede é composto de componentes externos e internos, incluindo demandas de clientes, fornecedores e a unidade fabril, possibilitando ainda a comunica-

ção com outras unidades de manufatura que compõem a rede e buscam equilibrar o desempenho global.

4.3.1 Componentes Externos

Os componentes externos compreendem todos os elementos que interagem com a unidade fabril, mas que não fazem parte diretamente de suas operações internas. Esses componentes influenciam significativamente a entrada e saída de materiais, bem como a demanda por produtos. Entre os principais componentes externos, destacam-se os clientes, que geram as demandas de produção com características específicas; e os fornecedores, que fornecem as matérias-primas necessárias para atender às demandas. A interação com esses componentes é fundamental para a unidade fabril, pois determina a entrada, disponibilidade e consumo de matéria prima, bem como entrega de produtos manufaturados.

4.3.1.1 Clientes

Clientes são componentes externos que geram demandas para a unidade fabril. Essas demandas são formalizadas em lotes, em certa quantidade, representando pedidos de produtos personalizados que devem ser atendidos dentro de prazos específicos. Cada demanda possui um tempo de entrada (DIDI) e um tempo de saída (DODO), que é um prazo máximo para entrega que o cliente está disposto a esperar. O tempo estimado de produção (*lead time* - LTLT) é calculado em função das características do produto demandado. Cada característica é realizada por uma máquina específica em tempo proporcional ao parâmetro de máquina. A partir da disponibilidade de matéria prima (saldo BABA) e dos tempos estimados para produção, LTLT, o tempo de produção real (TPTP) é calculado e, devido a indisponibilidade de matéria prima e sobrecarga da fábrica, pode eventualmente ultrapassar tempo de saída (DODO), ou seja, o prazo máximo de espera , obrigando clientes a cancelarem demandas.

A aceitação de uma demanda pode gerar receita para a fábrica, se o produto for fabricado e entregue dentro do prazo que o cliente está disposto a esperar (DO). Caso perca o prazo, o lote com o produto é movido para um pátio de manufaturados, ocupando um recurso limitado até que um novo cliente faça um pedido igual ou similar. Uma situação indesejada é a total ocupação do pátio de manufaturados de forma que um novo produto rejeitado pelo cliente. Em termos realistas, casos assim geram custos adicionais de terceirização de estocagem, ou reciclagem de componentes. Nesta modelagem, produtos não entregues e não estocados são perdidos e saem do sistema.

As demandas são recebidas em janelas de tempo específicas e são ordenadas pelo tempo de entrada (DI) em uma fila. Os atributos de demanda são: custo (CO), consumo de espaço (SP), lucro líquido (PR), e tamanho do lote (AM), entre outros. As características das demandas são representadas por um vetor $FT[1..M]$, que indica também o tempo de processamento em cada máquina.

Cada demanda (DE) gerada por um cliente é descrita por esse conjunto de atributos:

```
demands = DE[CU, DI, DO, TP, PR, CO, AM,
             SP, PE, LT, VA, SU, ST, FT[1..M]]
```

onde cada atributo está descrito na Tabela 1.

Tabela 1 – Descrição dos Atributos das Demandas

Atributo	**Descrição**
CU	ID do cliente
DI	Tempo de entrada
DO	Tempo de saída (Prazo máximo de espera)
TP	Tempo de produção
PR	Preço
CO	Custo
AM	Quantidade
SP	Consumo de espaço
PE	Multa por não entrega
LT	Tempo de entrega previsto
VA	Variabilidade
SU	Sustentabilidade
ST	Status da demanda
$FT[1..M]$	Matriz de características (tempo de máquina para cada recurso)

Cada lote de demanda tem seu custo esperado (CO), calculado em função das características (tempos de máquina) e tamanho de lote. O consumo de espaço (SP) é aplicado se o lote for movido para o pátio. Em termos econômicos, além do custo, estão modelados o lucro líquido (PR) para cada unidade solicitada e multa (PE) por aceitar e não entregar o lote. O atributo ST assume o status da demanda no sistema, conforme é mostrado adiante.

$$CO_d = AM_d \cdot FT_d \cdot EU \tag{4.1}$$

onde EU é o valor gasto pelo processamento da matéria-prima em cada máquina incluída no FT_d.

O consumo de espaço é calculado pela função γ:

$$SP_d = \gamma(AM_d \cdot FT_d) \tag{4.2}$$

onde a função γ expressa um fator de consumo de espaço considerando o vetor de características FT_d. Os demais atributos de demanda LT, VA e SU são calculados por funções específicas mostradas nas equações 4.3 a 4.5.

$$LT_d = \tau(AM_d \cdot FT_d) \tag{4.3}$$

onde τ é uma função referente à quantidade, características e expectativas de tempo atual para produção. A variabilidade (VA) e a sustentabilidade (SU) das demandas são calculadas pelas funções υ e σ, respectivamente:

$$VA_d = \upsilon(FT_d) \tag{4.4}$$

$$SU_d = \sigma(FT_d) \tag{4.5}$$

Cada demanda pode assumir diferentes status (ST_d) durante o processo de produção, variando de recebida (0), pronta (1), rejeitada (2), produzida (3), armazenada (4) e entregue (5):

$$ST_d = \{0, 1, 2, 3, 4, 5\} \tag{4.6}$$

4.3.1.2 Fornecedores

Os fornecedores entregam matéria-prima para atender às demandas de produção. A matéria-prima é solicitada com base nas necessidades da unidade fabril e pode estar sujeita a variabilidade na entrega, adicionando incerteza no modelo.

O estoque de matéria-prima é gerenciado através das variáveis de inventário. Dessa forma, o inventário de matérias-primas é gerido com base no saldo (BA), nas entradas (IN) e nas saídas (OU).

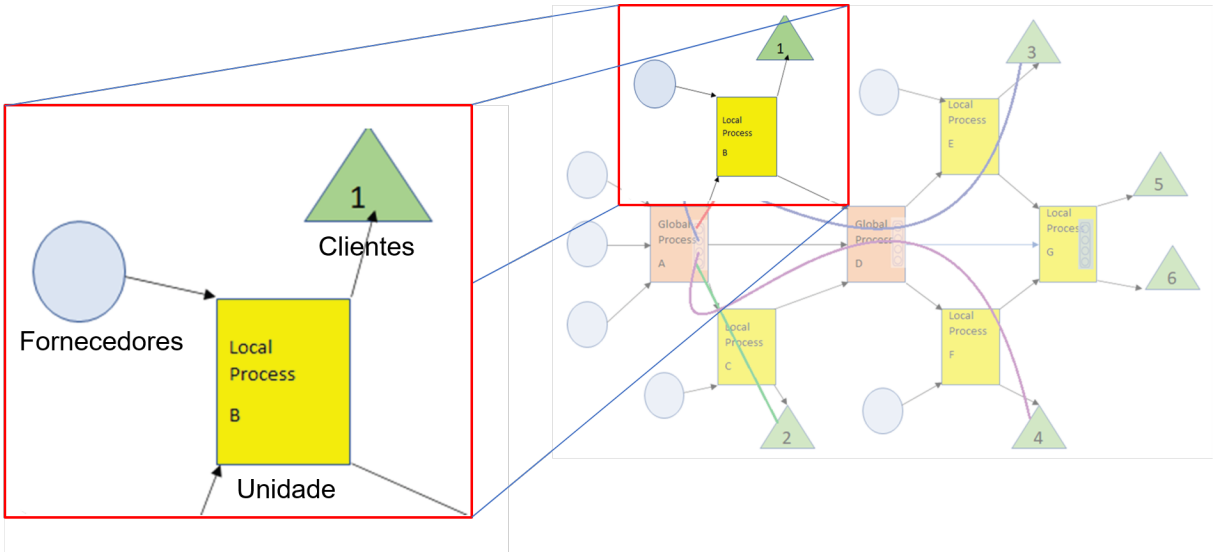
```
balance := BA[1..M]
income  := IN[1..M]
outcome := OU[1..M]
```

A disponibilidade para cada matéria-prima (AV) pode ser calculada pela fórmula:

$$AV = BA + IN - OU \tag{4.7}$$

A interação com fornecedores inclui a solicitação e recebimento de matéria-prima, conforme modelado no processo de tomada de decisão. A decisão de encomendar uma matéria-prima em falta, $\mathcal{O}(\overrightarrow{AV})$, é tomada pela indisponibilidade caso não haja um produto semelhante já disponível no pátio, pronto para ser entregue e falte matéria-prima para a produção.

Figura 4 – Unidade Fabril Periférica: mostra uma única unidade periférica (Processo Local B) que interage diretamente com fornecedores (círculos) e clientes (triângulos). A imagem detalha as conexões e fluxos de matéria-prima e produtos, entre a unidade periférica e seus fornecedores e clientes, destacando a interação local no contexto de uma rede de manufatura mais ampla.



Fonte: (O autor, 2024)

4.3.2 Componentes Internos

Os componentes internos do modelo incluem o pátio, as máquinas e o agente responsável pela negociação. O pátio armazena produtos não entregues a tempo, enquanto as máquinas são responsáveis pela produção dos produtos. O agente de negociação toma decisões sobre o agendamento de produção com base na observação do ambiente e nas políticas definidas.

4.3.2.1 Máquinas

As máquinas são componentes essenciais do ambiente fabril, responsáveis pelo processamento das demandas. Cada demanda passa por um conjunto de máquinas, e o tempo de processamento em cada máquina é representado pelo vetor de características (FT) que define o tempo que cada produto em cada máquina, influenciando diretamente os critérios de variabilidade e sustentabilidade. Por exemplo, uma demanda com $FT = [0, 4, 2, 3, 0]$ indica que ela não passa

pela primeira máquina, passa 4 unidades de tempo (u.t) na segunda máquina, 2 u.t na terceira, e 3 u.t na quarta.

A carga de trabalho (*Workload*) no sistema diz respeito a quantidade de lotes em produção em um dado instante e é um fator crítico que afeta o tempo de processamento e a eficiência da produção. Uma alta carga pode resultar em atrasos na produção, impactando a capacidade de atender às demandas no prazo estabelecido pelos clientes. O processo decisório deve considerar o *workload* atual ao planejar o agendamento de produção, evitando colocar em produção demandas com alto risco de não serem entregues no prazo.

4.3.2.2 Pátio

O pátio é o local onde os produtos acabados que não foram entregues dentro do prazo estabelecido pelos clientes são armazenados. O pátio serve como um *buffer* para produtos que perderam a janela de prazo e não puderam ser despachados a tempo. A gestão eficiente do pátio é crucial para minimizar os custos de armazenamento e evitar desperdícios. É esperado de sistemas sustentáveis um uso eficiente dos recursos, sejam eles energia, espaço de armazenamento (tanto de matéria-prima, quanto manufaturados), dentre outros.

Produtos armazenados no pátio podem ser despachados em pedidos futuros, caso sejam compatíveis com novas demandas. A função de correspondência $\mathcal{M}(FT_d, \overrightarrow{YA})$ é utilizada para verificar a similaridade entre as demandas atuais e os produtos armazenados, facilitando o reaproveitamento de produtos já produzidos. Um sistema de manufatura eficiente usa parte do recurso pátio evitando a ocupação total e risco de perda de lotes por falta de espaço.

4.4 Considerações Finais

A proposta de modelagem delineada neste capítulo proporciona uma estrutura robusta para a tomada de decisões de agendamento de produção em ambientes de manufatura dinâmicos e complexos. Com a aplicação prática e a validação experimental dessa metodologia, espera-se melhorar a eficiência, a flexibilidade e a sustentabilidade dos processos de produção em massa personalizada.

A próxima etapa envolverá a aplicação desta metodologia em cenários de simulação e, posteriormente, em ambientes reais de manufatura. A análise dos resultados permitirá o refinamento e a otimização da abordagem proposta, visando atender melhor às demandas da indústria de manufatura moderna.

5 Modelagem do Agente baseado em Aprendizado por Reforço

Para controlar a produção da unidade de manufatura, propõe-se algoritmos baseados em aprendizado por reforço que permitem ao agente decisor aprender e adaptar suas políticas de decisão com base em recompensas e penalidades adquiridas a partir do ambiente simulado. O agente coleta dados, e atualiza os parâmetros de decisão buscando aumentar a expectativa de recompensas futuras.

Nas próximas seções, são detalhados os componentes essenciais para a modelagem do agente, incluindo:

- **Processo de Decisão Sequencial**
  Descreve o processo de decisão sequencial em problemas de aprendizado por reforço, destacando a diferença entre métodos on-policy (como PPO) e off-policy (como DQN). A função de utilidade é obtida a partir do estado alcançado como resultado de toda a sequência de ações do agente, com o objetivo de encontrar a política ótima.

- **Representação de Estados e Ações**
  Apresenta a representação do espaço de estados e ações, permitindo que o agente tome decisões informadas. Inclui a descrição das variáveis observadas a cada passo e das ações tomadas pelo agente. A Figura 6 ilustra os estados operacionais e finais do sistema.

- **Dinâmica de Simulação**
  Descreve a interação contínua entre o agente e o ambiente, incluindo uma explicação detalhada dos estágios operacionais (recebido, pronto, produzido) e finais (rejeitado, armazenado, entregue). O agente ajusta suas políticas com base nas observações e resultados obtidos.

- **Fila de Prioridade**
  Explica o uso de filas de prioridade para organizar as demandas com base em diferentes critérios de priorização, como preço, variabilidade e sustentabilidade. Os índices de prioridade são calculados para cada demanda, ajudando o agente a decidir quais devem ser processadas primeiro.

- **Função Objetivo e Restrições**
  Define a função objetivo e as restrições associadas, que determinam os critérios de desempenho do agente. A função objetivo é uma composição de múltiplos objetivos, incluindo lucro, sustentabilidade e variabilidade, sujeita a restrições típicas do flow shop.

- **Recompensa**
  Descreve a função de recompensa, que está associada aos estágios de sucesso das demandas (armazenado ou entregue). A recompensa é modelada para refletir as consequências econômicas das decisões de produção, considerando também penalidades para situações indesejadas.

- **Agente**
  Detalha o agente de decisão, responsável por tomar decisões sobre o agendamento de produção com base nas observações do ambiente e nas recompensas e penalidades associadas. O agente utiliza aprendizado por reforço para otimizar suas estratégias e maximizar o lucro enquanto equilibra variabilidade e sustentabilidade.

- **Considerações Finais**
  Resume a metodologia delineada, destacando a estrutura sólida para a tomada de decisões de agendamento de produção em ambientes de manufatura dinâmicos e complexos. A aplicação prática e validação experimental dessa metodologia visam melhorar a eficiência, flexibilidade e sustentabilidade dos processos de produção em massa personalizada.

5.1 Processo de Decisão Sequencial

Em problemas de decisão sequencial, a função de recompensa acumulada é obtida a partir do estado alcançado como resultado de toda a sequência de ações do agente. Cada ação é escolhida a partir de um conjunto de decisões disponíveis e é baseada em uma política que define a regra de decisão em determinadas circunstâncias, considerando informações completas do estado do sistema. Um valor de recompensa imediata é obtido para cada decisão e o sistema transita para um novo estado. O objetivo de um problema de decisão sequencial é encontrar a política ótima, ou seja, uma sequência de regras de decisão para cada passo de tempo futuro, que maximize a recompensa acumulada ao longo do tempo (STRØM et al., 2018).

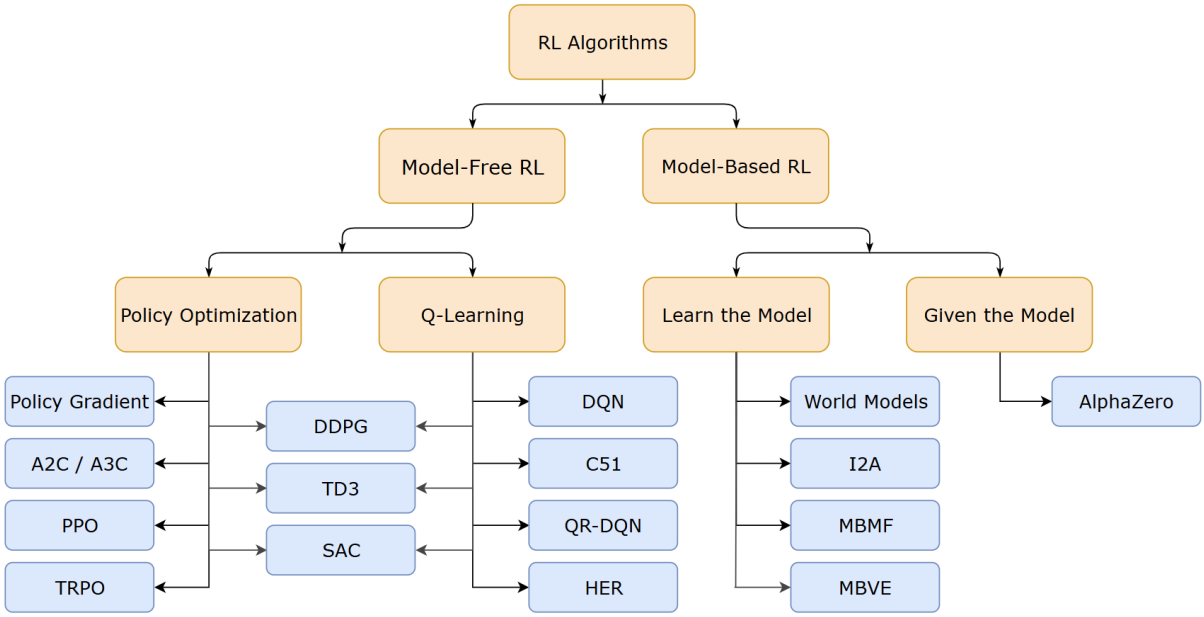
No aprendizado por reforço, o processo de decisão sequencial é utilizado para otimizar as ações do agente de forma a maximizar a recompensa acumulada. Dentro de RL, existem dois tipos principais de algoritmos: baseados em modelos (model-based) e sem modelo (model-free). Os algoritmos baseados em modelos utilizam um modelo explícito do ambiente para prever os efeitos de ações futuras, enquanto os algoritmos sem modelo dependem apenas das interações reais com o ambiente. Exemplos incluem AlphaZero, que combina planejamento baseado em modelo com aprendizado profundo para alcançar desempenho super-humano em jogos de tabuleiro (SILVER et al., 2017).

Além de Model-based e Model-free, os algoritmos de Aprendizagem por reforço podem ser categorizados em On-policy e Off-policy. No contexto do Aprendizagem por Reforço, existe um trade-off entre otimização de políticas (on-policy) e Q-learning (off-policy) (SUTTON;

BARTO, 2018). Métodos on-policy, como o PPO, otimizam diretamente a política atual, resultando em maior estabilidade, mas exigem mais amostras de dados (SCHULMAN et al., 2017b). Métodos off-policy, como o DQN, aprendem de experiências passadas sob diferentes políticas, sendo mais eficientes em termos de dados, mas potencialmente menos estáveis (MNIH et al., 2013).

A escolha entre on-policy e off-policy depende do problema. Em ambientes dinâmicos, on-policy é preferível, enquanto em cenários com coleta de dados limitada, off-policy é vantajoso (SUTTON; BARTO, 2018). Ambos os métodos visam otimizar a política do agente para maximizar a recompensa acumulada, utilizando interações com o ambiente para coletar dados e melhorar as decisões.

Figura 5 – Taxonomia de alguns dos algoritmos de Aprendizagem por Reforço.



Fonte: OpenAI Spinning Up (2024). Disponível em: <https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html>. Acesso em: 15 jul. 2024.

Neste trabalho, utilizaram-se três algoritmos de aprendizado por reforço: o Deep Q-Network (DQN), o Proximal Policy Optimization (PPO) e o Proximal Policy Optimization Recorrente (PPO Recorrente). A escolha desses algoritmos baseou-se em sua adequação para tarefas de manufatura, onde é necessário lidar com problemas complexos de otimização, adaptação a incertezas e eficiência na coleta de amostras (PANZER; BENDER, 2022). O aprendizado por reforço (RL) destaca-se nesse contexto por sua capacidade de aprender por tentativa e erro, adaptando-se a incertezas sem a necessidade de dados pré-coletados ou conhecimento especializado prévio. Esses algoritmos são amplamente utilizados em aplicações industriais, especialmente em tarefas de agendamento e controle de processos, tornando-os ideais para o

ambiente de manufatura estudado neste trabalho.

A Figura 5 apresenta uma taxonomia de alguns dos principais algoritmos de aprendizado por reforço. Os algoritmos utilizados neste trabalho, DQN e as duas versões do PPO (PPO e PPO Recorrente), estão posicionados à esquerda da árvore, sob a categoria de Model-Free RL. O DQN é um algoritmo off-policy, enquanto o PPO e o PPO Recorrente são algoritmos on-policy. Isso ilustra que ambos os algoritmos não dependem de um modelo explícito do ambiente, mas aprendem diretamente a partir das interações com o ambiente. A escolha desses algoritmos está diretamente relacionada ao processo de decisão sequencial, pois eles são capazes de otimizar as ações do agente em cada passo do tempo, com base nas recompensas acumuladas e nas transições de estado, proporcionando um aprendizado eficiente e adaptativo para o controle da produção na unidade de manufatura.

5.2 Representação de Estados e Ações

O ambiente de simulação foi desenvolvido utilizando a biblioteca *Gym*, que permite a criação de ambientes personalizados para o treinamento de algoritmos de aprendizagem por reforço. A representação dos estados, com informações relevantes do ambiente, permite que o agente possa tomar decisões informadas.

5.2.1 Representação de Estados

O estado do sistema é representado por um vetor que captura informações importantes sobre o ambiente de produção. As principais características incluídas na representação de estado são:

- Nível de estoque de matérias-primas.
- Número de demandas recebidas e seu status (prontas, em produção, rejeitadas, armazenadas, entregues).
- Disponibilidade das máquinas, carga e tempo estimado para conclusão das tarefas em execução.
- Capacidade atual do pátio de armazenamento.
- Penalidades acumuladas devido a atrasos ou rejeições.

5.2.2 Representação de Ações

As ações no ambiente representam as decisões do agente em relação às demandas recebidas, especificamente a previsão do atraso para cada demanda. O agente, ao observar o estado atual do sistema (nível de estoque, disponibilidade de máquinas, etc.), escolhe uma ação

que corresponde a uma previsão de atraso, em **unidades de tempo**, para a demanda em questão. Essa previsão influencia a decisão de produção: se o prazo de entrega (DO) da demanda for maior que a soma do tempo atual (t), o *lead time* (LT) da demanda e a ação (previsão de atraso), a demanda é produzida. Caso contrário, a demanda é rejeitada.

No presente trabalho, o espaço de ações é discreto e varia de 0 a *MAX_ATRASO*, representando a previsão do atraso em unidades de tempo. O valor *MAX_ATRASO* é um hiperparâmetro do ambiente, definido como 100 unidades de tempo. A escolha de uma ação pelo agente corresponde à seleção de um valor discreto dentro desse intervalo, que representa a sua estimativa do atraso para a demanda em análise. Essa representação discreta facilita o aprendizado e permite usar algoritmos de Aprendizagem por Reforço com espaços de ações discretos. A unidade de tempo, neste contexto, é utilizada de forma genérica, sem especificar segundos, minutos, horas ou qualquer outra unidade específica.

5.3 Dinâmica de Simulação

A dinâmica de simulação descreve a interação entre o agente e o ambiente. O agente recebe informações sobre o estado do sistema, toma decisões de produção e ajusta sua *política* com base nos resultados. Essa interação captura a complexidade e a variabilidade do ambiente de produção. É importante distinguir os **estados** da Aprendizagem por Reforço, que representam a configuração completa do sistema em um determinado instante, dos **estágios** da simulação, que descrevem as diferentes fases por que uma demanda pode passar. Um estado da RL pode incluir informações sobre todos os estágios das demandas em processamento.

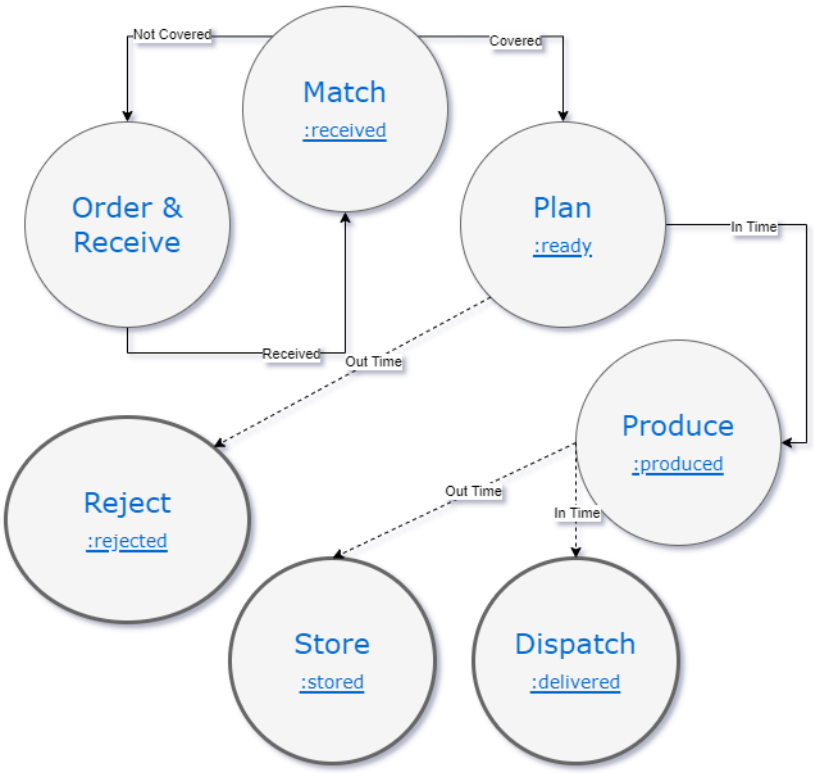
A Figura 6 ilustra a sequência de estágios que uma demanda pode atravessar dentro do ambiente de simulação, incluindo recebido (*Order-Receive-Match*), pronto (*Plan*), produzido (*Produce*), rejeitado (*Reject*), armazenado (*Store*) e entregue (*Dispatch*). A transição entre os estágios é determinada pelas ações do agente, que prevê o atraso das demandas e tomam decisões de produção ou rejeição com base nessas previsões.

O processo local investigado pode ser isolado da rede de manufatura restante considerando algumas interações com clientes e fornecedores locais. Portanto, o processo de tomada de decisão pode ser significativamente simplificado para uma sequência de decisão que avance ou cancele as etapas de produção de bens manufaturados, dependendo do risco de lucro ou prejuízo diante das circunstâncias e incertezas.

Os círculos representam procedimentos e valores de estágios assumidos, enquanto as setas conectam estágios dependendo de certas condições. As linhas traçadas conectam os estágios finais nos quais uma recompensa ou penalidade pode ser aplicada às demandas atendidas.

Os procedimentos *Order & Receive* e *Match* compõem o procedimento de negociação, no qual os fornecedores são contatados para fornecer as matérias-primas necessárias para cobrir

Figura 6 – Estágios da simulação, incluindo 4 estágios operacionais: recebido (*Order-Receive-Match*), pronto (*Plan*), produzido (*Produce*); e 3 estágios finais: rejeitado (*Reject*), armazenado (*Store*) e entregue (*Dispatch*)



Fonte: (O autor, 2024)

os produtos demandados. Todas as demandas que chegam podem ser gerenciadas para serem priorizadas pelas mais promissoras, considerando o lucro e os recursos disponíveis para atender a produção.

A tomada de decisão gira em torno do risco inerente ao processo produtivo, que depende de muitos fatores, como disponibilidade de fornecedores, confiabilidade de processos e máquinas que ocupam espaço na própria fábrica, além de penalidades e multas formatadas em contratos (HECKMANN; COMES; NICKEL, 2015; SILBERMAYR; MINNER, 2014).

Neste trabalho, foi utilizada a definição clássica de risco considerando exclusivamente o fator tempo. No entanto, o risco pode também ser calculado assumindo a necessidade de encomendar quantidades adequadas de matéria-prima (risco de interrupção de fornecedores) ou complexidade de produção (BRINTRUP et al., 2020). Como mostra a Figura 6, a decisão de prosseguir com o cronograma de produção depende apenas do fator tempo (`in Time` ou `out Time`).

Quando uma determinada demanda atinge o estágio pronto para produção (*covered*), chegando em planejamento (*Plan*), e o risco de produção fora do prazo é aceitável, o pedido é colocado na linha de produção (*Produce*). *Dispatch* e *Store* são estágios finais para uma produção

finalizada, mas diferenciáveis por entrega bem-sucedida ou não, respectivamente. Por outro lado, se o risco for alto a demanda não será produzida, ou seja, ela vai para o estágio de rejeitada (*Reject*).

A tabela 2 traz um resumo dos estágios, procedimentos e regras de decisão do ambiente implementado:

Tabela 2 – Procedimentos e regras de decisão no ambiente de simulação.

Estágios	**Procedimentos e regras de decisão**
Order & Receive	As demandas são recebidas e ordenadas de acordo com a prioridade. Se houver estoque suficiente de matérias-primas (*covered*), a demanda é movida para o estágio de planejamento (*Plan*).
Match	As demandas são casadas com o estoque disponível. Se o produto que está no estoque cobre completamente a demanda do cliente, ela é retirada do estoque, faturada e entregue.
Plan	No estágio de planejamento, o agente decide o momento adequado para iniciar a produção com base nas previsões de atraso.
Produce	A produção das demandas é realizada conforme o planejamento. Se a produção é completada dentro do tempo, a demanda é movida para o estágio de despacho (*Dispatch*); se há atraso, a demanda pode ser armazenada (*Store*) ou descartada se o pátio estiver cheio.
Dispatch	As demandas produzidas dentro do prazo são despachadas aos clientes.
Store	As demandas produzidas fora do prazo podem ser armazenadas se houver espaço disponível no pátio.
Reject	As demandas que não podem ser produzidas a tempo (ou seja, previsão de atraso muito alto) são rejeitadas.

5.4 Fila de Prioridade

As filas de prioridade são utilizadas para organizar as demandas de acordo com diferentes critérios de priorização, como preço, variabilidade e sustentabilidade. Essas filas ajudam o agente a decidir quais demandas devem ser processadas primeiro, considerando a maximização do lucro e o equilíbrio dos critérios de desempenho.

A priorização das demandas é realizada com base nos índices de prioridade calculados para cada demanda:

$$h_{1,d} = \frac{CO_d}{PR_d} \tag{5.1}$$

$$h_{2,d} = \frac{1}{VA_d} \tag{5.2}$$

$$h_{3,d} = \frac{1}{SU_d} \tag{5.3}$$

$$h_{4,d} = \frac{1}{(DO_d - LT_d - t + 1)} \tag{5.4}$$

Quanto menor o valor calculado, maior a prioridade para uma dada demanda.

O índice h_1 é calculado para toda demanda d considerando a relação entre custo e lucro. Lembrando que o lucro já está descontado de todos os custos de produção, ou seja, é um lucro líquido. O índice h_2 é uma proporção inversa da variedade de características de uma determinada ordem. Lembrando que a variedade é um conceito subjetivo calculado pela soma do vetor de características incluídas (quanto mais características, mais variedade incluída). O índice h_3 faz o mesmo para a suposta sustentabilidade de uma determinada demanda. Lembrando que variedade é um conceito subjetivo calculado por uma função de sustentabilidade, flexível o suficiente para incorporar questões relacionadas a fornecedores de agenda verde ou pegada de carbono, como exemplos. O índice h_4 considera o tempo de folga para formular o risco de falha na entrega como uma proporção inversa do tempo restante, a partir do instante t, para produzir uma dada demanda, considerando o prazo DO_d, e descontado o *lead* esperado tempo LT_d.

Os índices permitem tomar decisões considerando diferentes observações do sistema, estabelecendo assim um conjunto flexível de variáveis de estado que modelam a política de decisão. Neste trabalho, emprega-se o índice h_4, que permite uma política de decisão voltada para minimizar a perda de entrega por atraso.

5.5 Função Objetivo e Restrições

A função objetivo e as restrições associadas definem os critérios de desempenho do agente. A função objetivo é uma composição não normalizada de três objetivos:

$$\max \sum_{x_d \in ST=:delivered} PR \cdot x_d \quad + \sum_{x_d \in ST=:produced} (SU \cdot x_d + VA \cdot x_d) \tag{5.5}$$

Não excluindo outras restrições típicas do flow shop, a função objetivo está sujeita a

$$C(t)_{\mathbf{d}_{\leq d}} + LT \cdot x_d \leq \max(0, DO \cdot x_d - t) \quad \text{for} \quad x_d \in ST =_{:produced} \tag{5.6}$$

onde $C(t)_{\mathbf{d}_{\leq d}}$ é o tempo de produção efetivo para todas as demandas **d** antes de d e converge para o *makespan* global, $C(max)$, após a última demanda produzida, excluindo aquelas que violam a restrição de tempo representada pela Equação 5.6.

5.6 Recompensa

A função de recompensa está associada a uma determinada demanda d atingindo um dos estágios de sucesso: armazenado ou entregue. Lembrando, armazenada é uma situação relacionada à produção de demanda bem-sucedida, que perdeu o prazo para ser despachada. Então, o produto é colocado no estoque para que possa ser entregue em um pedido futuro.

Em relação aos sistemas de produção personalizados, um dos principais desafios é combinar as demandas de entrada específicas com o estoque existente. Simplesmente estocar lotes pode não ser benéfico com espaço de armazenamento limitado. Em vez disso, é importante evitar aceitar pedidos urgentes que podem levar a riscos de encalhe. Se houver espaço de armazenamento disponível, o excesso de estoque pode ser uma opção de último recurso. No entanto, isso não deve ser incentivado como política, e a decisão deve ser baseada na avaliação criteriosa dos riscos e benefícios. Quando há espaço de armazenamento, armazenar o excesso de estoque só deve ser recompensado. Se nenhum espaço de armazenamento estiver disponível, ele deve ser penalizado proporcionalmente por violar a restrição de espaço. Por outro lado, a remuneração pela entrega de produtos customizados deve ser baseada no lucro nominal definido para o lote.

A equação 5.7 modela apenas a recompensa econômica, considerando os três estágios viáveis: armazenado, entregue e rejeitado, em um determinado episódio. Em relação a um alinhamento de demandas processadas antes do d atual ($\mathbf{d} \leq d$), a recompensa atualizada **d+1** pode ser adicionada por um fator proporcional ao lucro nominal $PR_d \cdot AM_d$, em caso de entrega (estágio *:delivered*), ou ser penalizado geometricamente pelo consumo atual SP_d do espaço do pátio $|YARD|$ (chegando a zero para uma situação de pátio quase transbordado), no caso de armazenando (estágio *:stored*). **Os valores de recompensa e lucro são expressos em unidades monetárias genéricas (u.m.), sem especificar a moeda (real, dólar, euro, etc.).**

$$RW_{\mathbf{d+1}} = \begin{cases} \frac{RW_{\mathbf{d}_{\leq d}}}{|YARD| - SP \cdot x_d + 1} & \text{se a demanda d for } {}_{:stored} \\ RW_{\mathbf{d}_{\leq d}} + PR \cdot x_d \cdot AM \cdot x_d & \text{se a demanda d for } {}_{:delivered} \\ RW_{\mathbf{d}} & \text{se a demanda d for } {}_{:rejected} \end{cases} \tag{5.7}$$

A rejeição da ordem de produção é tratada neste trabalho como uma decisão operacional baseada no risco de falha na entrega e, portanto, não passível de penalidade. Em um cenário

de rede fabril, podem existir alternativas, como encaminhar a demanda de produção para outra unidade para a qual haja matéria-prima disponível. Em um cenário de manufatura aditiva, os tempos de transporte podem ser negligenciados.

5.7 Agente

O agente é responsável por tomar decisões sobre o agendamento de produção, com base nas observações do ambiente e nas recompensas e penalidades associadas. Neste trabalho, o agente utiliza aprendizagem por reforço para aprender e adaptar suas estratégias, buscando otimizar o lucro e equilibrar os critérios de variabilidade e sustentabilidade com base em uma decisão simples de aceitação ou rejeição de demandas. A técnica de aprendizagem por reforço é capaz de maximizar a expectativa de ganho no futuro a partir de séries complexas de dados que representam o estado do sistema em constante alteração. As variáveis mais importantes para essa decisão são o prazo estabelecido pelo cliente DO, o vetor de características FT, a estimativa de tempo de processamento (LT), e a carga atual do sistema.

5.8 Considerações Finais

A metodologia delineada neste capítulo proporciona uma estrutura sólida para a tomada de decisões de agendamento de produção em ambientes de manufatura altamente dinâmicos e complexos. Ao considerar a revisão da literatura sobre Produção em Massa Personalizada e Redes de Manufatura Auto-Organizadas, concentra-se esforços em capacitar a tecnologia baseada em aprendizado por reforço para fornecer um *framework* inicial para essa tomada de decisões.

Ao modelar o ambiente de produção, identificam-se as unidades fabris como sistemas complexos, compostos por unidades periféricas e centrais. Essa abordagem nos permitiu simplificar o sistema, assumindo unidades de *backbone* responsáveis pela produção em massa e unidades periféricas voltadas para a personalização de produtos. Além disso, definem-se suposições específicas para o ambiente de produção simulado, incluindo a autonomia das unidades fabris, a negociação de matérias-primas e prazos, e a preocupação com a variabilidade introduzida pelas características regionais dos produtos.

O processo de decisão sequencial foi detalhadamente descrito, destacando a importância de encontrar a política ótima para cada passo de tempo futuro. Além disso, apresentam-se as variáveis e funções necessárias para modelar o processo de decisão sequencial simplificado para a simulação de Sistemas de Manufatura Orientados para o Cliente (SOMN). Essas variáveis e funções incluem inventário, demandas, procedimentos e regras de decisão, e índices para avaliação multicritério.

A avaliação multicritério foi discutida em detalhes, destacando a importância de equilibrar

os objetivos de lucro, sustentabilidade e atendimento ao cliente local. A função objetivo e as restrições associadas foram definidas, permitindo uma abordagem flexível para a tomada de decisões que leva em consideração múltiplos critérios. Além disso, a função de recompensa foi modelada para refletir as consequências das decisões de produção, considerando a entrega bem-sucedida, armazenamento e rejeição de ordens de produção.

Com base nessa metodologia robusta, pode-se agora prosseguir para a implementação prática e validação experimental. A próxima etapa envolverá a aplicação dessa metodologia em cenários de simulação e, posteriormente, em ambientes reais de manufatura. A análise dos resultados obtidos nos permitirá refinar e otimizar ainda mais nossa abordagem, visando melhorar a eficiência, a flexibilidade e a sustentabilidade dos processos de produção em massa personalizada.

6 Resultados computacionais

Os experimentos foram projetados para validação do simulador de unidade fabril e do agente decisor baseado em aprendizagem por reforço. A unidade é controlada pelo agente de forma a aceitar, produzir e entregar pedidos dentro dos prazos ou rejeitá-los considerando restrições de estoque, espaço no pátio, ao mesmo tempo em que persegue objetivos de econômicos e não econômicos, relacionados com sustentabilidade e diversificação.

6.1 Implementação do Framework

O framework foi implementado na linguagem *Python* (versão 3.10.11), utilizando a biblioteca *stable-baselines3* (RAFFIN et al., 2021) e o toolkit *OpenAI Gym* (BROCKMAN et al., 2016), especificamente para o simulador de unidade (ambiente). Também foi utilizada a otimização de hiperparâmetros com *Weights & Biases* (WandB) (BIEWALD, 2020) para sintonia e registro dos resultados. Os experimentos computacionais foram executados no VSCode e no ambiente Google Colab[1], e podem ser encontrados no repositório no github[2] .

6.2 Dados de instância

O simulador SOMN é flexível na incorporação de diferentes perfis de sistema por meio do ajuste de parâmetros. Os principais parâmetros são a capacidade de armazenamento (`YA`), o número de recursos ou matérias-primas (`M`), o tamanho do conjunto de demanda (`N`) e os limites superiores para os seguintes parâmetros : Tempo de vencimento (`MAXDO`), tamanho do lote (`MAXAM`), fator de lucro (`MAXPR`), penalidade de rejeição (`MAXPE`), tempo de máquina de recurso (`MAXFT`), custo do recurso (`MAXEU`).

Destacam-se `YA`, `M` e `N` como aqueles relacionados ao tamanho da instância, enquanto os demais parâmetros são mais impactantes nas tomadas de decisão. A partir desses parâmetro várias instâncias são geradas aleatoriamente a cada experimento, dependendo do número de episódios. Neste trabalho, os parâmetros usados são {`YA=10`, `M=5`, `N=7`, `MAXDO=10`, `MAXAM=5` , `MAXPR=1.5`, `MAXPE=10`, `MAXFT=5`, `MAXEU=10`}.

6.3 Espaço de controle e ação

O risco de aceitação é inversamente proporcional à folga, calculada pela Eq. 5.4, e diretamente proporcional ao *Workload* (WL_t) atual da unidade. Dessa forma, o risco de aceitação

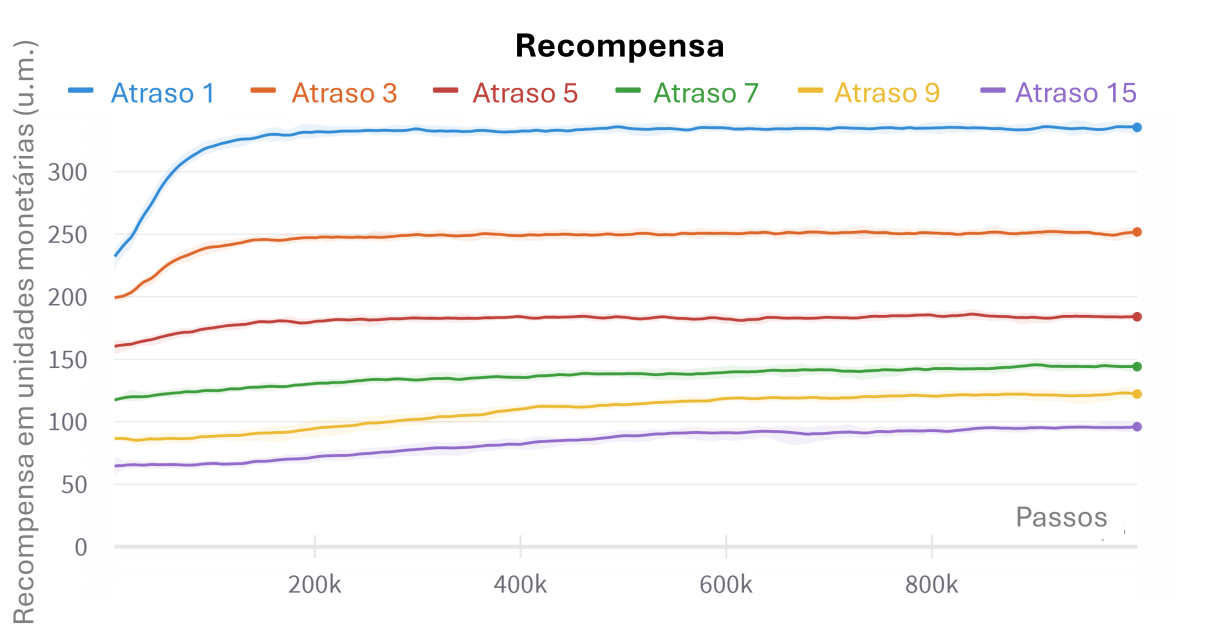
[1] <https://colab.research.google.com/>
[2] <https://github.com/fredericmenezes/SOMN>

é desconhecido, pois depende do *lead time* esperado LT, incluindo atrasos probabilísticos relacionados também ao número de demandas em produção em um dado instante.

O atraso probabilístico é modelado pela distribuição de Poisson, centrada em $LT_d + WL_t$ $\rho(\mu = LT + WL_t)$. Um resultado válido é um agente prevendo e aprendendo o atraso e revelando a função de probabilidade. Se o agente superestimar o atraso, provavelmente rejeitará solicitações que poderiam ser concluídas a tempo de gerar lucro e recompensa. Caso contrário, subestimando, o agente aceita demandas de alto risco que podem não ser entregues e ocuparão um espaço valioso no pátio.

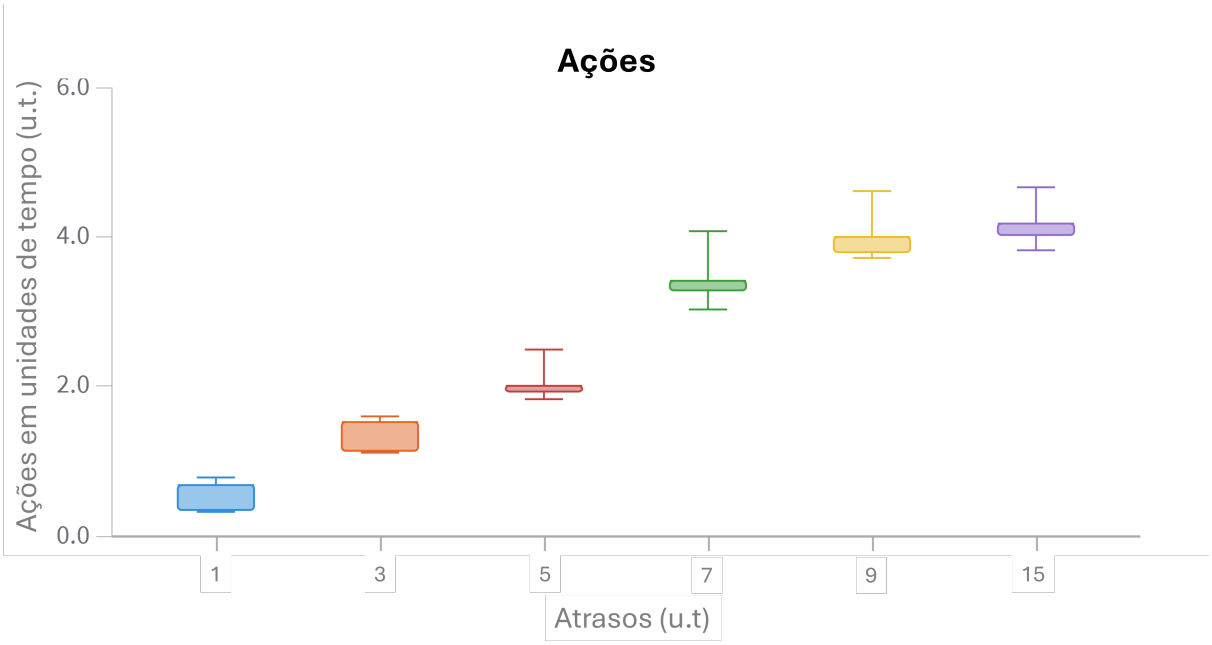
A Figura 7 mostra o desempenho do agente PPO para diferentes perfis de atraso. Observa-se que as ações de aceitação da demanda representam como o agente entende a distribuição de probabilidade de atraso. Neste caso, os atrasos foram particularmente fixados em: 1, 3, 5, 7, 9 e 15, para permitir a observação das ações, e verificar se estão respondendo de forma coerente. É perceptível que o agente tenta acompanhar o atraso, mas sempre com subestimação dos riscos, supostamente buscando aumentar a recompensa pelo lucro. Além disso, para perfis de atraso maiores, o sistema tende a perder desempenho em relação à recompensa obtida em cada intervalo de tempo.

Figura 7 – Agente PPO reagindo aos perfis de atrasos fixos 1, 3, 5, 7, 9 e 15. As ações subestimam a função de recompensa decai para $1/6$ a medida que o atraso real aumenta de 1 para 15.



Fonte: (O autor, 2024)

Figura 8 – A medida que o atraso aumenta de 1 para 15, as ações estão crescendo junto com o atraso real. As ações estão aumentando, mas sempre subestimando o atraso real.



Fonte: (O autor, 2024)

6.4 Hiper-parametrização

Para encontrar os melhores parâmetros para os algoritmos de aprendizado por reforço, utilizou-se a otimização *bayesiana*, implementada através do WandB. Foram realizadas 150 execuções para cada algoritmo (PPO, PPO Recorrente, DQN). No WandB, existem principalmente três métodos para encontrar os melhores parâmetros: método em grade (*grid*), método randômico (*random*) e método de Otimização Bayesiana (*bayes*). A Pesquisa em Grade (*grid*) examina todas as combinações possíveis de um conjunto predefinido de hiperparâmetros. Embora exaustivo e computacionalmente caro, garante a exploração completa do espaço de busca. A Pesquisa Aleatória (*random*) seleciona combinações de hiperparâmetros aleatoriamente dentro dos limites definidos. Esse método é mais eficiente que a pesquisa em grade, especialmente em espaços de busca grandes, pois pode encontrar boas combinações com menos avaliações. A Otimização Bayesiana (*bayes*) utiliza um modelo probabilístico para escolher de forma inteligente as próximas combinações de hiperparâmetros a serem avaliadas. Este método equilibra a exploração de novas áreas com a exploração de áreas promissoras.

A Otimização Bayesiana é uma técnica avançada que constrói um modelo probabilístico do espaço de hiperparâmetros e utiliza esse modelo para guiar a exploração e a exploração dos hiperparâmetros. Esta abordagem é particularmente eficiente porque requer menos avaliações para encontrar combinações ótimas de hiperparâmetros em comparação com métodos tradicionais como *grid* e *random*. Um modelo probabilístico, geralmente um processo gaussiano, aproxima a função objetivo, que é o desempenho do modelo em relação aos hiperparâmetros. Este modelo é

atualizado com os resultados de cada nova avaliação.

Uma função de aquisição determina a próxima combinação de hiperparâmetros a ser avaliada com base no modelo *surrogate*. Funções de aquisição comuns incluem *Expected Improvement* (EI), *Upper Confidence Bound* (UCB) e *Probability of Improvement* (PI) (WANG et al., 2023). Essas funções equilibram a exploração de áreas do espaço de hiperparâmetros que ainda não foram exploradas com a exploração de áreas que já mostraram resultados promissores. A cada iteração, o modelo *surrogate* é atualizado com os novos resultados e a função de aquisição é usada para selecionar a próxima combinação de hiper parâmetros a ser avaliada. Este processo se repete até que um critério de parada seja alcançado, como um número máximo de avaliações ou uma melhoria mínima na métrica de desempenho.

Para o PPO Recorrente, DQN e PPO utilizaram-se as seguintes configurações de hiperparâmetros como demostradas nas tabelas 3, 4 e 5 respetivamente. Essas são as faixa de valores utilizadas para escolha em cada uma das execuções. Ou seja, em cada execução é escolhido um valor para cada parâmetro dentre a faixa de valores apresentadas abaixo.

Tabela 3 – Configurações de hiperparâmetros para PPO Recorrente

Hiperparâmetro	**Valores**
Batch Size	[8; 16; 32; 64; 128; 256]
N Steps	[8; 16; 32; 64; 128; 256; 512; 1024; 2048]
Gamma	[0,9; 0,95; 0,98; 0,99; 0,995; 0,999; 0,9999]
Learning Rate	Uniforme, 1e-05 a 1,0
Entropy Coefficient (ent_coef)	Uniforme, 1e-08 a 0,1
Clip Range	[0,1; 0,2; 0,3; 0,4]
N Epochs	[1; 5; 10; 20]
GAE Lambda	[0,8; 0,9; 0,92; 0,95; 0,98; 0,99; 1,0]
Max Grad Norm	[0,3; 0,5; 0,6; 0,7; 0,8; 0,9; 1; 2; 5]
Value Function Coefficient (vf_coef)	0,0 a 1,0

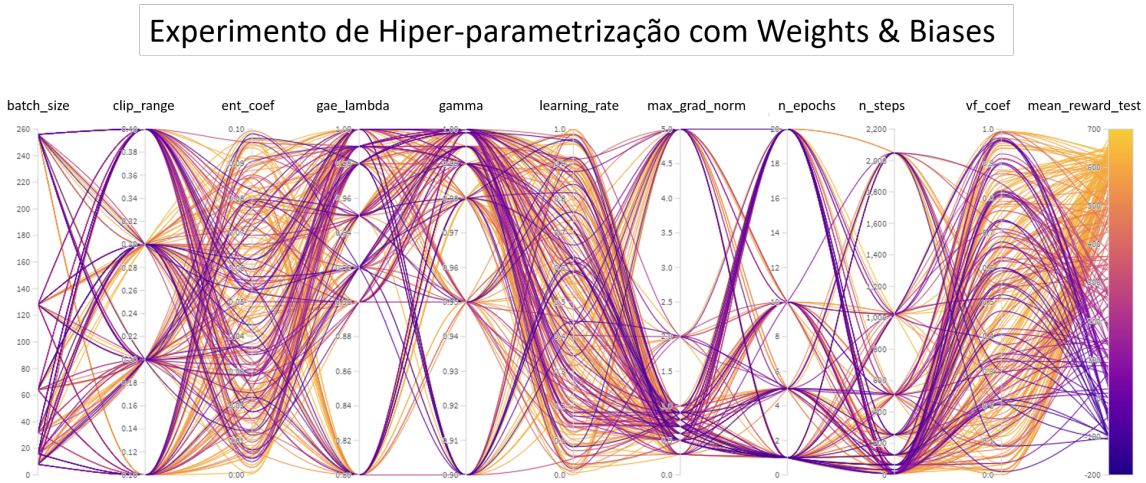
A Figura 9 mostra um gráfico de coordenadas paralelas (*parallel coordinates plot*) gerado pelo *Weights & Biases* (WandB) durante um *sweep* de hiperparâmetros. Esse tipo de gráfico é utilizado para visualizar a relação entre múltiplas variáveis (neste caso, os hiperparâmetros) e o desempenho do modelo. Nesse exemplo, foram testados 10 parâmetros, 150 vezes, com configurações diferentes para um *Algoritmo PPO Recorrente*. A cada teste, era medida a média da recompensa representada na última coluna, com cores variando, de cima para baixo, de

Tabela 4 – Configurações de hiperparâmetros para DQN

Hiperparâmetro	**Valores**
Batch Size	[32; 64; 128; 256]
Gamma	0,8 a 0,999
Learning Rate	1e-05 a 0,01
Buffer Size	[10000; 50000; 100000; 200000]
Learning Starts	[1000; 5000; 10000]
Target Update Interval	[100; 500; 1000]
Train Frequency	[1; 4; 10]
Gradient Steps	[1; 4; 8]
Exploration Fraction	0,1 a 0,3
Exploration Final Epsilon	0,01 a 0,05
Max Grad Norm	0,5 a 2,0

amarelo (melhor recompensa), até um tom de roxo mais escuro (pior execução).

Figura 9 – Análise de hiperparâmetros do algoritmo PPO Recorrente utilizando o *WandB*.



Fonte: (O autor, 2024)

Para visualizar os resultados da hiper-parametrização, empregaram-se gráficos de coordenadas paralelas, particularmente úteis para observar a relação entre múltiplos hiperparâmetros e o desempenho do modelo. Cada linha colorida no gráfico representa uma execução individual do *sweep*, e a cor da linha indica o valor da métrica de desempenho, permitindo identificar

Tabela 5 – Configurações de hiperparâmetros para PPO

Hiperparâmetro	Valores
Learning Rate	Uniforme, 1e-05 a 1e-01
N Steps	[1024; 1536; 2048; 2304; 2560; 2816; ... ... 3072; 3328; 3584; 3840; 4096]
Batch Size	[16; 32; 64; 128; 256]
N Epochs	Inteiro uniforme, 3 a 30
Gamma	Uniforme, 0,9 a 0,9999
GAE Lambda	Uniforme, 0,8 a 0,95
Clip Range	0,1 a 0,3
Entropy Coefficient (ent_coef)	Uniforme, 0,0 a 0,01
Value Function Coefficient (vf_coef)	0,5 a 1,0
Max Grad Norm	0,5 a 1,0
Target KL	Uniforme, 0,003 a 0,03

rapidamente quais combinações de hiperparâmetros resultaram em melhor desempenho.

O uso da *Otimização Bayesiana* permitiu explorar de forma eficiente o espaço de hiperparâmetros e encontrar combinações que maximizaram o desempenho dos algoritmos PPO Recorrente, PPO e DQN. A visualização dos resultados através de gráficos de coordenadas paralelas facilitou a análise das relações complexas entre os hiperparâmetros e o desempenho do modelo, contribuindo para a escolha das melhores configurações para os experimentos.

Os parâmetros apresentados na Tabela 6 refletem configurações específicas que resultaram no melhor desempenho para cada algoritmo durante os experimentos. O ajuste fino desses hiperparâmetros é essencial para garantir que os algoritmos aprendam ou encontrem, de maneira eficiente e robusta, os parâmetros que representam decisões no ambiente de manufatura.

Para o PPO, os parâmetros como **batch_size**, **clip_range**, **ent_coef**, **gae_lambda** e **learning_rate** foram otimizados para equilibrar a estabilidade e a eficiência do treinamento. O parâmetro **n_steps** foi ajustado para garantir que o agente pudesse coletar informações suficientes sobre o ambiente antes de atualizar a política. A escolha de um valor mais alto para o **gamma** assegurou que o agente priorizasse recompensas futuras, alinhando-se com a natureza sequencial das decisões de produção.

No caso do PPO Recorrente, ajustes semelhantes foram encontrados, com algumas variações, para acomodar a natureza recorrente do algoritmo. O **batch_size** e o **n_steps** foram aumentados para 256 e 4096, respectivamente, permitindo ao agente explorar um maior número de interações e aprender padrões temporais mais complexos. O aumento do valor de **gae_lambda**,

Tabela 6 – Melhores Parâmetros para PPO, PPO Recorrente e DQN

Parâmetro	PPO	PPO Recorrente	DQN
batch_size	128	256	128
clip_range	0,2	0,3	-
ent_coef	0,01	0,01	-
gae_lambda	0,95	0,99	-
gamma	0,99	0,999	0,95
learning_rate	0,001	0,0005	0,001
max_grad_norm	0,9	1	1
n_epochs	10	20	-
n_steps	2048	4096	-
vf_coef	0,5	0,5	-
buffer_size	-	-	100000
exploration_final_eps	-	-	0,02
exploration_fraction	-	-	0,2
gradient_steps	-	-	4
learning_starts	-	-	5000
target_update_interval	-	-	500
train_freq	-	-	10

e do valor de **max_grad_norm**, ajudou a melhorar a estabilidade do treinamento em ambientes parcialmente observáveis.

Para o DQN, a otimização focou em parâmetros específicos do algoritmo, como por exemplo o **buffer_size**, **exploration_final_eps** e **learning_starts**. O **buffer_size** foi definido como 100000, permitindo ao agente armazenar e reutilizar uma grande quantidade de experiências passadas, crucial para a eficiência de amostras em métodos *off-policy*. Parâmetros como o **exploration_final_eps** e **exploration_fraction** foram ajustados para balancear a exploração e a explotação, enquanto **learning_starts** e **target_update_interval** foram configurados para garantir que o treinamento começasse com uma base de experiências suficientemente rica.

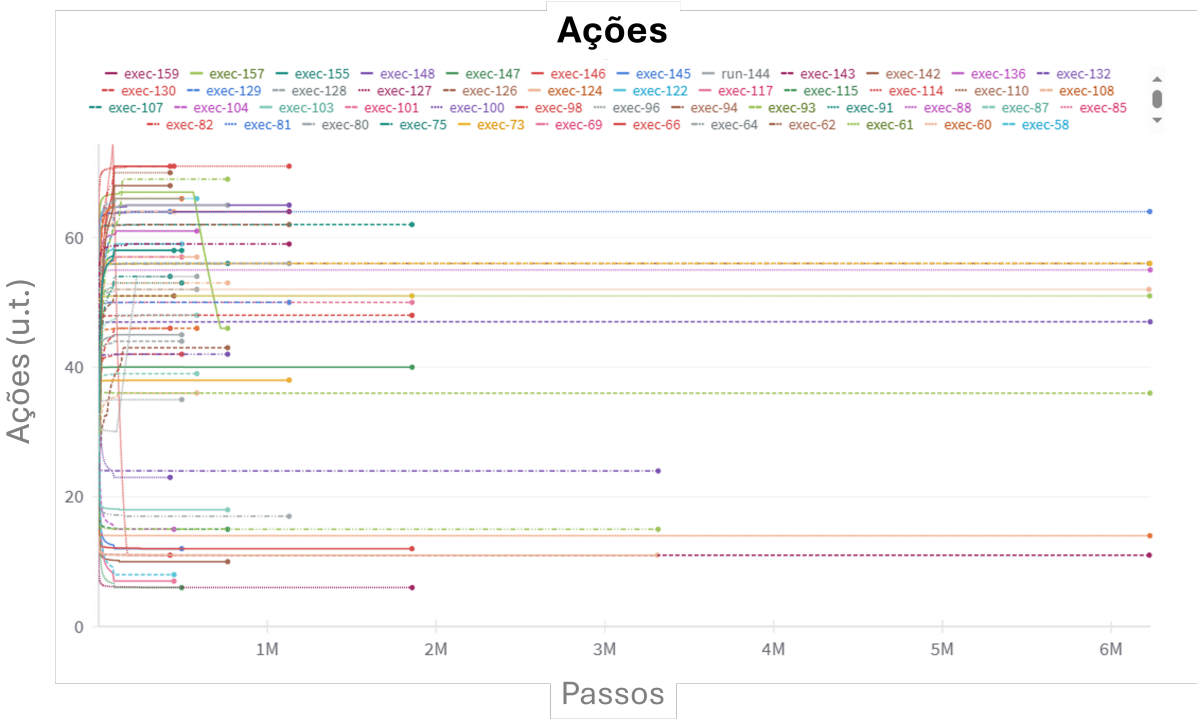
A seleção cuidadosa desses parâmetros demonstra a importância da otimização de hiperparâmetros no aprendizado por reforço, especialmente em ambientes industriais complexos. Cada ajuste contribui para a capacidade do agente de aprender políticas eficazes que otimizam o processo de produção, levando em consideração a variabilidade e a sustentabilidade dos critérios de desempenho. Ao utilizar os parâmetros otimizados, os algoritmos PPO, PPO Recorrente e

DQN são capazes de alcançar um desempenho robusto e eficiente, adaptando-se às dinâmicas do ambiente de manufatura e contribuindo para a tomada de decisões mais informadas e eficazes.

6.5 Validação do ambiente

A validação do ambiente foi realizada por meio de um experimento no qual as ações de decisão, em cada execução, seguiram regras fixas, *sem inteligência* (ver Figura 10). As decisões variaram entre aceitar todas as demandas, rejeitar todas as demandas ou decisões aleatórias. Analisou-se o comportamento do ambiente e a coerência com as decisões tomadas.

Figura 10 – Várias execuções distintas com diferentes ações ao longo dos passos de tempo

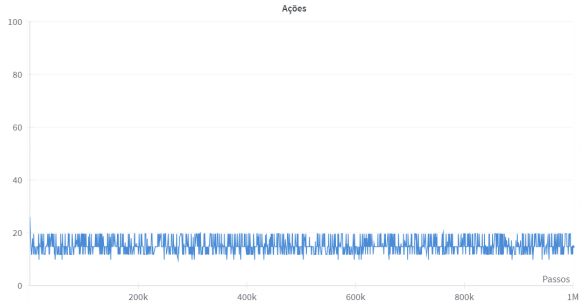


Fonte: (O autor, 2024)

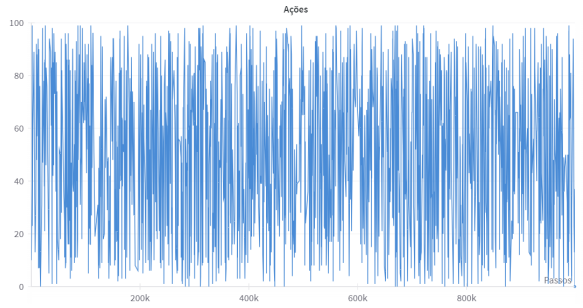
As execuções de regras fixas (não inteligente) são comparadas com uma execução utilizando um agente PPO Recorrente (inteligente), que toma decisões buscando maximizar a recompensa futura. Os gráficos comparativos das ações, lucro, variabilidade e sustentabilidade para ambas as execuções são apresentados na Figura 11.

Na execução inteligente (Figura 11a), o agente PPO Recorrente demonstra um comportamento eficiente, aceitando demandas que proporcionavam maior lucro, sustentabilidade e variabilidade. Pode-se dizer que o agente conseguiu aproximar a função que modela os atrasos, resultando em um desempenho geral superior. Os gráficos mostram que, ao longo do tempo,

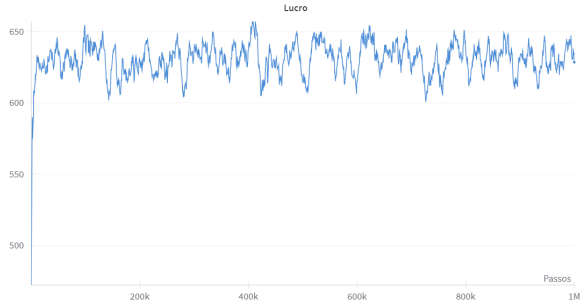
Figura 11 – Comparação entre Execuções: Ações, Lucro, Variabilidade e Sustentabilidade



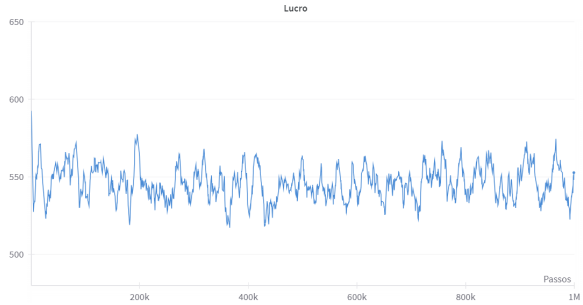
(a) Execução Boa: Ações



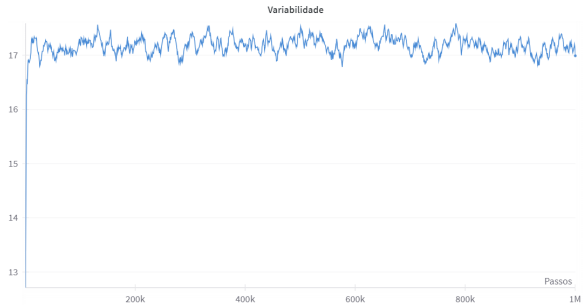
(b) Execução Ruim: Ações



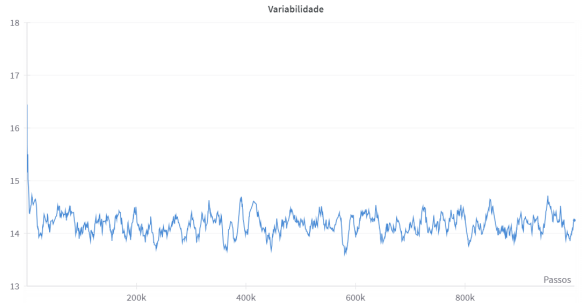
(c) Execução inteligente: Lucro



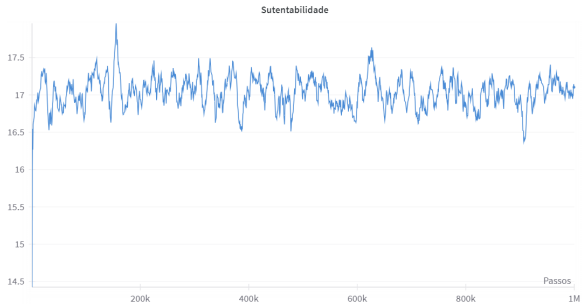
(d) Execução não inteligente: Lucro



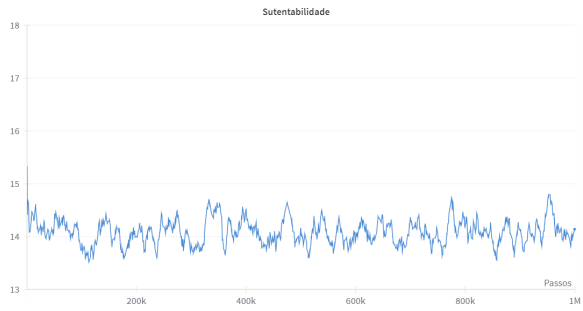
(e) Execução inteligente: Variabilidade



(f) Execução não inteligente: Variabilidade



(g) Execução inteligente: Sustentabilidade



(h) Execução não inteligente: Sustentabilidade

Fonte: (O autor, 2024)

as ações do agente são consistentes: o lucro permanece alto, e tanto a variabilidade quanto a sustentabilidade mantém-se em níveis elevados.

Por outro lado, na execução de regras fixas (não inteligente), observada na Figura 11b,

o agente randômico apresentou um desempenho significativamente inferior. Este agente, ao tomar decisões aleatórias, ora superestima ora subestima os atrasos, resultando em perdas no lucro e nos níveis de variabilidade e sustentabilidade. Os gráficos evidenciam que a abordagem randômica, como esperado, está longe de obter resultados satisfatórios.

Os resultados confirmam que o ambiente responde corretamente às ações do agente que, de forma autônoma, cumpre o papel de um decisor que poderia ser um humano bem-treinado para o ambiente fabril modelado. A execução inteligente demonstra que o agente PPO Recorrente é capaz de aproximar uma função atraso aleatória, mas dependente da complexidade das demandas e da carga do sistema de produção em um dado instante.

Além disso, é possível observar como as ações refletem na ocupação do pátio e na carga da unidade fabril, bem como a penalidade associada ao uso do pátio. Três execuções representativas são detalhadas a seguir na Figura 12.

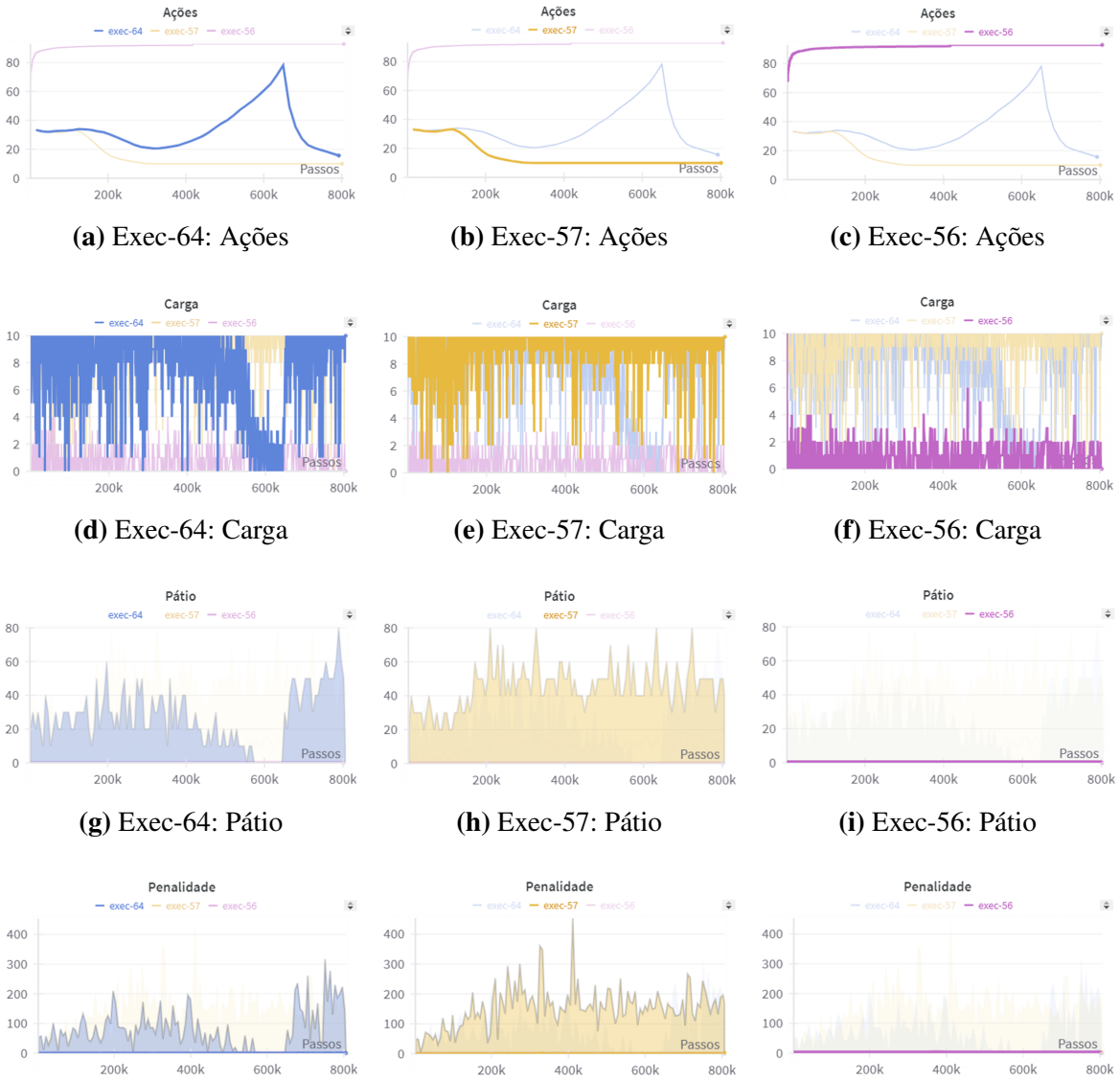
Na primeira execução (exec-64), apresentada na Figura 12a, o agente inicia com ações de valores abaixo de 40, representando uma baixa expectativa de atraso (risco), alcançando 80 e depois recuando para abaixo de 20. Essa execução está representada pela cor azul e dura 800k passos, com o pico de atraso previsto ocorrendo pouco depois do passo 600k. Esse comportamento reflete-se na carga (*workload*) do sistema, apresentada na Figura 12d, onde a carga começa alta, diminui devido ao aumento na expectativa de atraso e sobe novamente quando o valor das ações recuam. Esse comportamento é esperado, pois as ações representam previsões de atraso e, ao prever atrasos significativos, o agente rejeita as demandas, resultando em máquinas ociosas e carga reduzida. Posteriormente, com a diminuição das ações, a carga aumenta novamente.

Na segunda execução (exec-57, Figura 12b), representada pela cor amarela, o agente prediz atrasos baixos (ações abaixo de 20), aceitando todas as demandas que chegam. Isso resulta em carga alta (Figura 12e) e pátio cheio (Figura 12h), pois a alta carga afeta a produção, causando atrasos. Quando a produção atrasa e o produto não é entregue a tempo, o cliente cancela a ordem, e o produto vai para o pátio, consumindo recursos da unidade fabril e aumentando a penalidade (Figura 12k).

A última coluna, exec-56, representada pela Figura 12c), mostra um agente que rejeita todas as demandas, indicado por uma linha de valores de ações de cor vermelha. As ações são sempre altas (acima de 80), indicando previsões de grandes atrasos para as demandas. Ao rejeitar as demandas, o agente evita a produção, resultando em baixa carga (Figura 12f), pátio vazio (Figura 12i) e baixa penalidade (Figura 12l). No entanto, embora a penalidade seja baixa, essa situação não é ideal, pois a unidade fabril não gera receita sem produzir.

Este experimento demonstra a consistência do ambiente e dos indicadores de desempenho da unidade fabril, em resposta à tomada de decisão do agente. Ao todo, foram realizadas 450 execuções, evidenciando a robustez e a flexibilidade do framework desenvolvido.

Figura 12 – Comparação entre Execuções Exec-64, Exec-57 e Exec-56: Ações, Carga, Pátio e Penalidade



(a) Exec-64: Ações **(b)** Exec-57: Ações **(c)** Exec-56: Ações

(d) Exec-64: Carga **(e)** Exec-57: Carga **(f)** Exec-56: Carga

(g) Exec-64: Pátio **(h)** Exec-57: Pátio **(i)** Exec-56: Pátio

(j) Exec-64: Penalidade **(k)** Exec-57: Penalidade **(l)** Exec-56: Penalidade

Fonte: (O autor, 2024)

6.6 Comparação entre os agentes inteligentes

A análise comparativa dos agentes inteligentes PPO, PPO Recorrente, DQN e Randômico (baseline) é apresentada nesta seção, confrontando métricas como ações tomadas, lucro, penalidade, utilização do pátio, sustentabilidade e variabilidade.

6.6.1 Comparação das Ações Tomadas

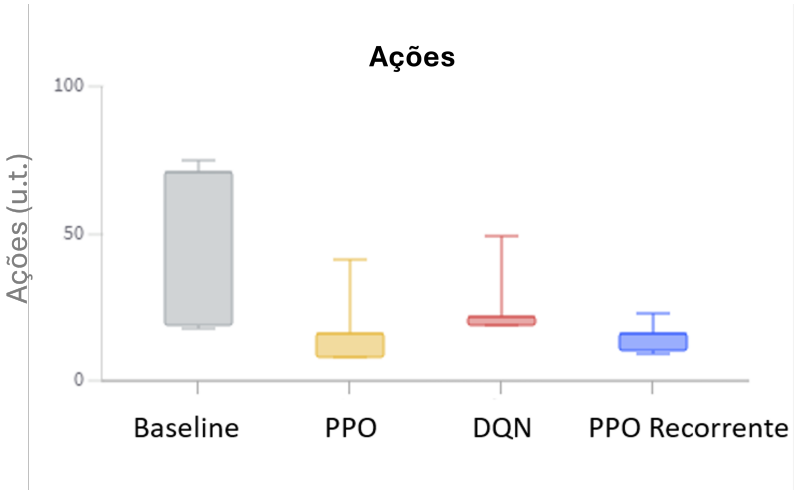
As ações tomadas pelos diferentes agentes foram comparadas utilizando *boxplots* e gráficos de linha. A Tabela 7 mostra as métricas das ações tomadas pelos agentes, com destaque

para o comportamento mais preciso do agente PPO Recorrente, com menor variabilidade $15,16 \pm 4,99$ e intervalo menor $[9,24 \;\; 23,04]$.

Tabela 7 – Métricas das ações tomadas pelos agentes.

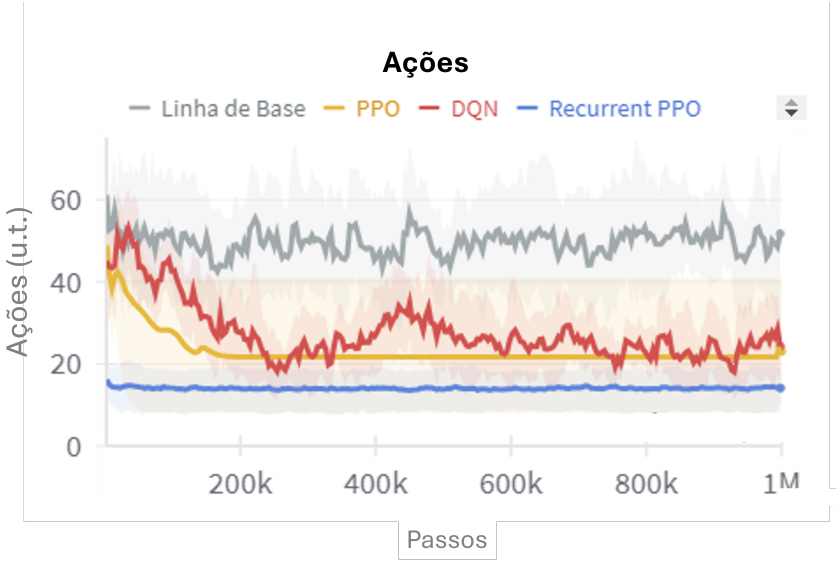
Agente	Média	Desvio Padrão	Mínimo	Máximo
DQN	30,00	16,52	19,00	49,00
PPO	21,67	17,21	8,00	41,00
PPO Recorrente	**15,16**	4,99	9,24	23,04
Randômico	48,67	42,19	0,00	75,00

Figura 13 – *Boxplot* das ações tomadas pelos agentes.



Fonte: (O autor, 2024)

Figura 14 – Gráfico de linha das ações tomadas pelos agentes ao longo do tempo.



Fonte: (O autor, 2024)

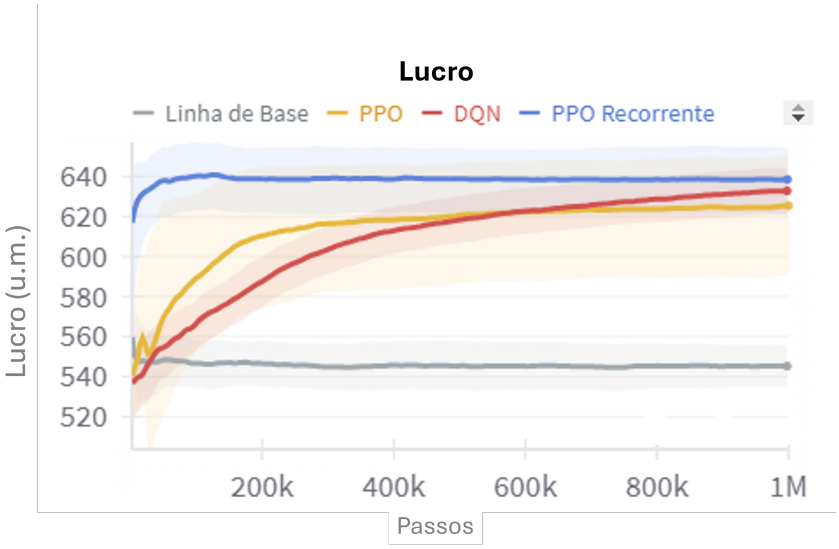
6.6.2 Análise do Lucro

O lucro foi utilizado como uma métrica de desempenho para avaliar a eficácia dos agentes em maximizar a receita. A Tabela 8 apresenta as métricas do lucro obtido por cada agente. Como uma variabilidade menor, o agente PPO Recorrente destaca-se mais uma vez por alcançar com maior rapidez a média de lucro em torno de $643,81 \pm 10,70$ contra $627,12 \pm 24,40$ do agente DQN, com segundo melhor rendimento.

Tabela 8 – Métricas do lucro obtido pelos agentes.

Agente	Média	Desvio Padrão	Mínimo	Máximo
Randômico	545,67	7,19	522,38	639,89
PPO	616,00	13,88	517,12	648,80
DQN	627,12	24,40	521,92	657,76
PPO Recorrente	**643,91**	10,69	551,83	685,44

Figura 15 – Gráfico de linha do lucro obtido pelos agentes ao longo do tempo.



Fonte: (O autor, 2024)

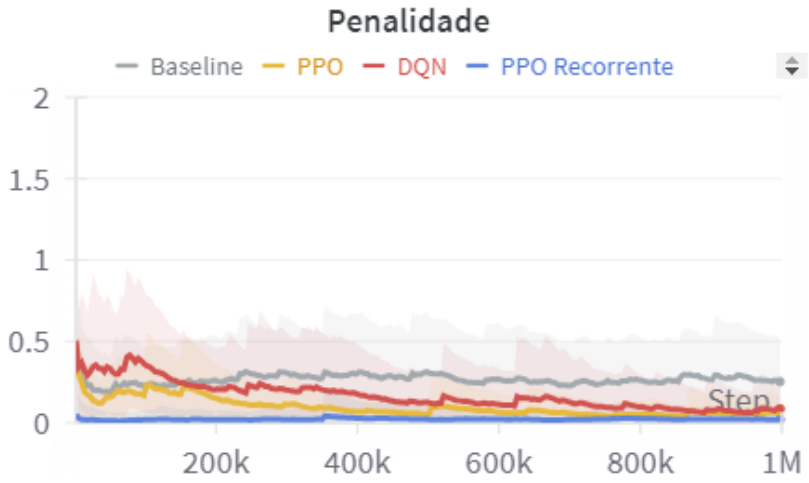
6.6.3 Análise da Penalidade

A penalidade foi utilizada para medir os custos associados ao atraso na produção. A Tabela 9 mostra as métricas das penalidades acumuladas pelos agentes. Mais uma vez, o agente PPO Recorrente apresenta melhor média e menor variabilidade, seguido do agente PPO.

Tabela 9 – Métricas das penalidades acumuladas pelos agentes.

Agente	Média	Desvio Padrão	Mínimo	Máximo
Randômico	0,25	0,58	0,00	12,84
PPO	0,06	0,36	0,00	21,79
DQN	0,09	0,58	0,00	32,28
PPO Recorrente	**0,02**	0,12	0,00	9,48

Figura 16 – Gráfico de linha das penalidades acumuladas pelos agentes ao longo do tempo.



Fonte: (O autor, 2024)

6.6.4 Análise do Pátio

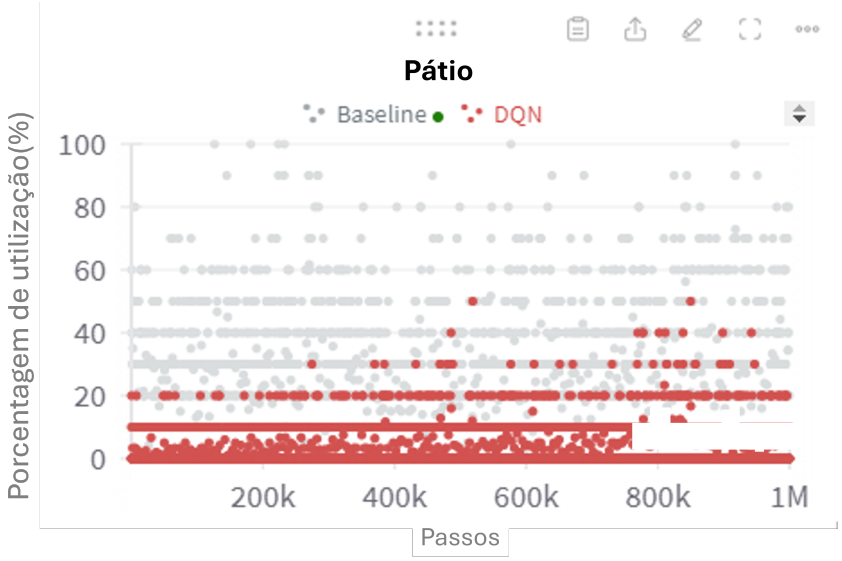
A utilização do pátio foi medida para avaliar como os agentes lidam com o armazenamento de produtos não entregues a tempo. A Tabela 10 mostra que o agente PPO Recorrente utilizou menos o pátio em comparação aos outros agentes, com uma média de 0,129 e desvio padrão de 1,178.

Tabela 10 – Métricas da utilização do pátio dos agentes.

Agente	Média	Desvio Padrão	Mínimo	Máximo
Randômico	8,97	17,15	0,00	100,00
PPO	1,07	3,80	0,00	40,00
DQN	2,17	5,23	0,00	60,00
PPO Recorrente	**0,13**	1,18	0,00	30,00

A Figura 17 compara a utilização do pátio pelo agente DQN com o agente randômico. Observa-se que o DQN utilizou significativamente menos o pátio em comparação ao *baseline* randômico.

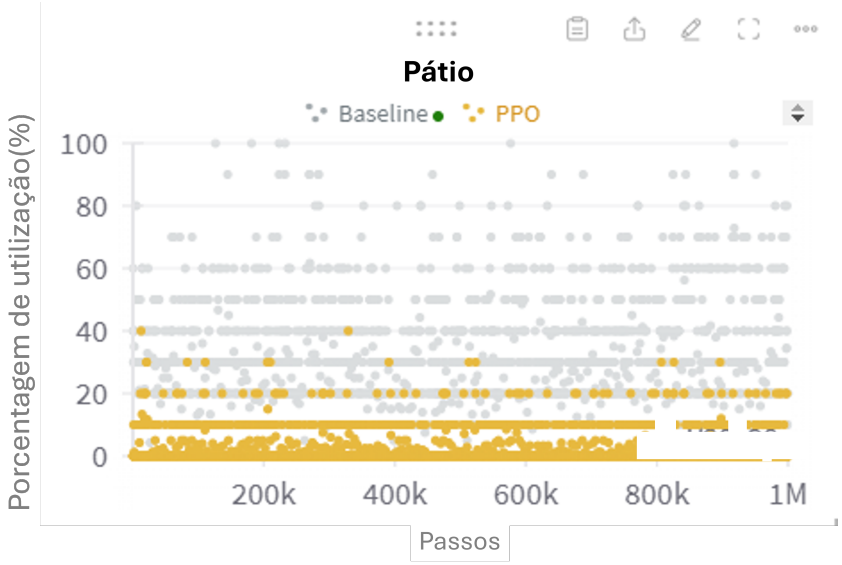
Figura 17 – Comparação da utilização do pátio pelo agente DQN e o baseline randômico.



Fonte: (O autor, 2024)

A Figura 18 mostra a comparação da utilização do pátio pelo agente PPO em relação ao *baseline* randômico. O agente PPO também apresenta uma utilização reduzida do pátio.

Figura 18 – Comparação da utilização do pátio pelo agente PPO e o baseline randômico.

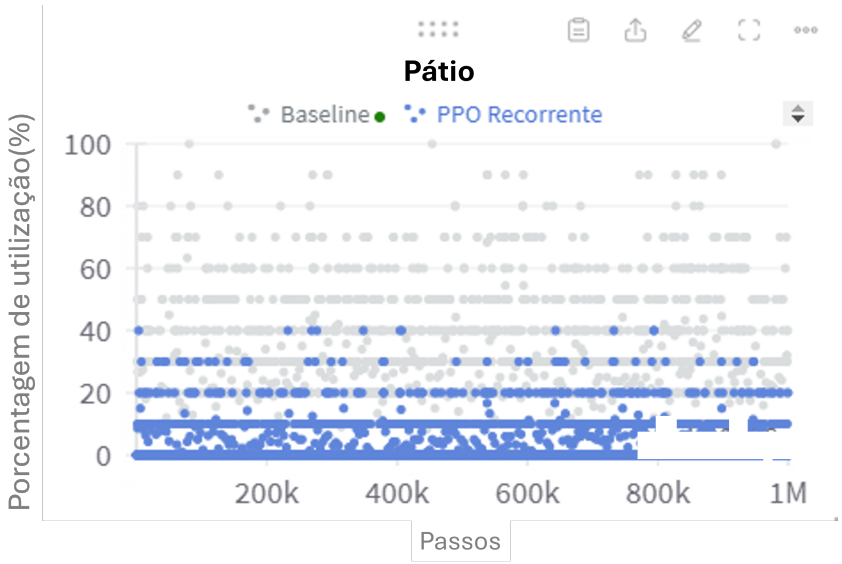


Fonte: (O autor, 2024)

Finalmente, a Figura 19 apresenta a comparação entre o agente PPO Recorrente e o

baseline randômico. O agente PPO Recorrente teve a menor utilização do pátio entre todos os agentes.

Figura 19 – Comparação da utilização do pátio pelo agente PPO Recorrente e o baseline randômico.



Fonte: (O autor, 2024)

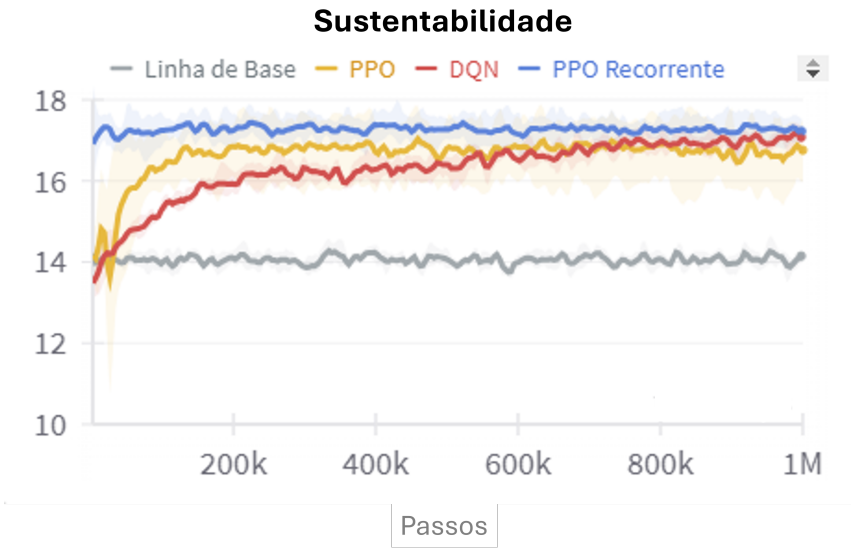
6.6.5 Análise da Sustentabilidade

A sustentabilidade foi medida para avaliar a eficiência dos agentes em buscar priorizar demandas com menor consumo de recursos e tempo de máquina, o que supostamente está associado à sustentabilidade da produção. O agente PPO Recorrente apresentou a maior média de sustentabilidade, 17,43, com um desvio padrão de 0,21, conforme apresentado na Tabela 11.

Tabela 11 – Métricas da sustentabilidade dos agentes.

Agente	Média	Desvio Padrão	Mínimo	Máximo
Randômico	14,06	2,38	11,82	15,24
PPO	16,37	0,44	13,02	17,46
DQN	16,31	0,72	13,38	17,26
PPO Recorrente	**17,43**	0,21	15,65	19,33

Figura 20 – Gráfico de linha da sustentabilidade dos agentes ao longo do tempo.



Fonte: (O autor, 2024)

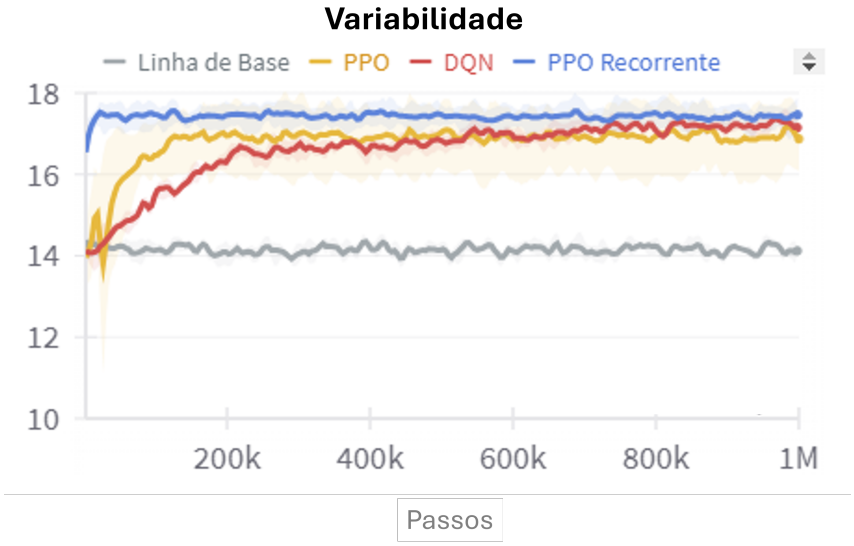
6.6.6 Análise da Variabilidade

A variabilidade, que mede a capacidade de priorizar demandas com maior personalização, foi maior para o PPO Recorrente, com média $17,59 \pm 0,15$. As métricas detalhadas estão na Tabela 12.

Tabela 12 – Métricas da variabilidade dos agentes.

Agente	Média	Desvio Padrão	Mínimo	Máximo
Randômico	15,26	2,64	14,33	16,53
PPO	16,52	0,41	13,29	17,49
DQN	16,62	0,75	13,92	17,41
PPO Recorrente	**17,59**	0,15	14,09	18,16

Figura 21 – Gráfico de linha da variabilidade dos agentes ao longo do tempo.



Fonte: (O autor, 2024)

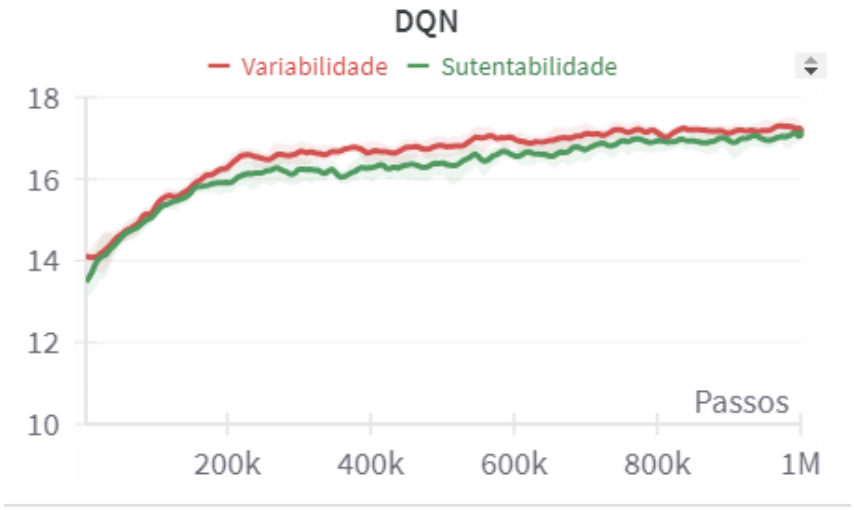
6.6.7 Análise Conjunta de Sustentabilidade e Variabilidade

Os agentes foram também avaliados quanto à sua habilidade de equilibrar sustentabilidade e variabilidade. A Figura 22 apresenta a relação de variabilidade e sustentabilidade para o agente DQN, enquanto as Figuras 23 e 24 apresentam as mesmas relações para os agentes PPO e PPO Recorrente, respectivamente. Os agentes PPO e PPO Recorrente conseguiram equilibrar sustentabilidade e variabilidade de maneira mais eficaz em comparação ao agente randômico. As métricas detalhadas estão na Tabela 13.

Tabela 13 – Métricas da sustentabilidade e variabilidade dos agentes.

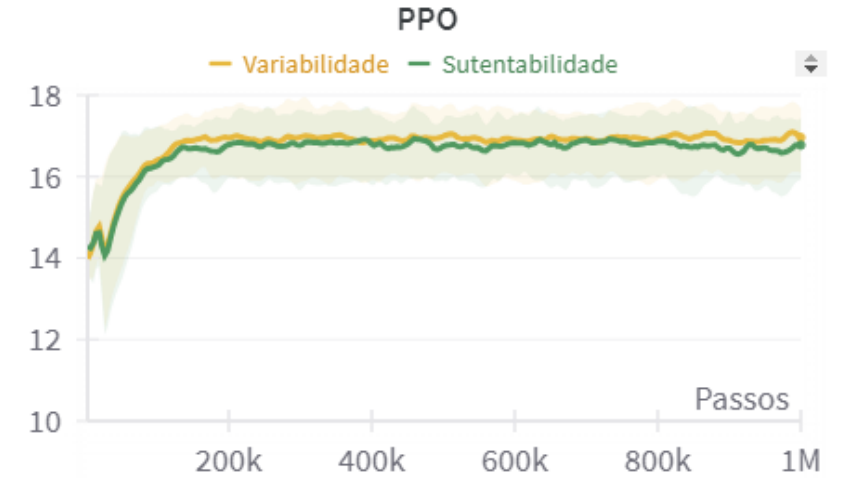
Métrica	Sustentabilidade Média	Sustentabilidade Desvio Padrão	Variabilidade Média	Variabilidade Desvio Padrão
Randômico	16,44	2,38	16,80	2,64
PPO	16,37	0,44	16,52	0,41
DQN	16,31	0,72	16,62	0,75
PPO Recorrente	17,43	0,21	17,59	0,15

Figura 22 – Comparação da sustentabilidade e variabilidade para o agente DQN.



Fonte: (O autor, 2024)

Figura 23 – Comparação da sustentabilidade e variabilidade para o agente PPO.



Fonte: (O autor, 2024)

Figura 24 – Comparação da sustentabilidade e variabilidade para o agente PPO Recorrente.

Fonte: (O autor, 2024)

Os resultados mostram que os agentes inteligentes apresentam variações significativas em suas métricas de desempenho. O PPO Recorrente destacou-se pela consistência de suas ações, penalidades mais baixas e maior sustentabilidade e variabilidade, evidenciando sua eficiência na priorização de demandas e aproximação da função atraso. Todos se saíram melhores que o agente randômico, que apresentou alta variabilidade nas ações e penalidades elevadas, refletindo a falta de uma política estruturada.

A análise comparativa dos agentes PPO, PPO Recorrente, DQN e randômico demonstrou que os agentes baseados em aprendizado por reforço, especialmente o PPO Recorrente, oferecem melhor desempenho em termos de sustentabilidade, variabilidade e lucro, com menores penalidades e melhor utilização do pátio.

6.7 Considerações Finais

Neste capítulo, foram apresentados os resultados da comparação entre diferentes agentes de aprendizado por reforço no contexto de uma unidade fabril. Três agentes inteligentes (PPO, PPO Recorrente e DQN) e um agente randômico (baseline) foram avaliados em termos de ações, lucro, penalidade, pátio, sustentabilidade e variabilidade. As métricas extraídas das execuções dos agentes fornecem uma análise detalhada sobre o desempenho de cada um em diferentes aspectos do problema.

Os resultados mostram que o agente PPO Recorrente obteve um desempenho superior em várias métricas. Especificamente, o PPO Recorrente apresentou a menor penalidade média 0,0216 e a menor utilização do pátio, da ordem de 0,1293, sugerindo uma gestão mais eficiente

das demandas e dos recursos da unidade fabril. Além disso, o PPO Recorrente conseguiu manter uma alta sustentabilidade (média de 17,43) e variabilidade (média de 17,59), demonstrando um bom equilíbrio entre esses dois aspectos essenciais para a produção sustentável e eficiente.

Comparando os agentes inteligentes com o agente randômico, observou-se que o PPO Recorrente reduziu a penalidade em aproximadamente 92,6% em relação ao agente randômico, que apresentou uma penalidade média de 0,2926. Além disso, o PPO Recorrente também reduziu a utilização do pátio em cerca de 98,6% em comparação com o agente randômico, que teve uma utilização média do pátio de 8,9740.

Os gráficos de linha de lucro mostram que o PPO Recorrente alcançou o maior lucro médio (643,81), seguido pelo DQN (627,12), PPO (616,00) e, por último, o agente randômico (673,18). A menor variabilidade nas ações do PPO Recorrente (desvio padrão de 4,995) sugere que esse agente foi mais consistente em suas decisões ao longo do tempo.

Além disso, as análises de sustentabilidade e variabilidade dos agentes mostram que os agentes PPO e PPO Recorrente conseguiram equilibrar esses dois aspectos de forma mais eficaz que o agente randômico. O agente PPO Recorrente destacou-se novamente com a maior sustentabilidade e variabilidade médias, indicando uma política bem estruturada que consegue atender às demandas de maneira equilibrada.

Em resumo, as análises e comparações realizadas neste capítulo fornecem uma base sólida para a conclusão de que os agentes de aprendizado por reforço, e em particular o PPO Recorrente, são ferramentas eficazes para a otimização de processos em unidades fabris. Os resultados apresentados encorajam extensões de implementações para incorporar novos aspectos de uma rede de manufatura, como o compartilhamento de informação entre múltiplos agentes, responsáveis pela tomada de decisão em outras unidades.

7 Conclusão

O contexto da produção personalizada em rede pode ser alcançado quando um conjunto de unidades fabris simuladas trabalha de forma independente e toma decisões com autonomia, apoiando também o engajamento local e o desempenho econômico global. O problema se assemelha a um job shop que depende de atrasos na coleta de matéria-prima e da capacidade produtiva já comprometida com entregas aceitas e em andamento. Vale ressaltar que os procedimentos *Order & Receive* e *Match* são afetados pela decisão central, e todo o desempenho do sistema depende da política aprendida e executada pelo agente. Abordagens de otimização tradicionais não contemplam tais aspectos.

Os primeiros resultados apontam para a validação do simulador de unidade fabril, conceitualmente responsável por interagir de forma mais próxima com fornecedores locais e polos consumidores regionais. Foram implementados três agentes inteligentes baseados em Aprendizagem por Reforço: agentes PPO, PPO Recorrente, e DQN; um agente de decisão aleatória foi utilizado como baseline para efeito de validação.

Para comparar o desempenho dos agentes, foram utilizadas métricas como ações tomadas, lucro, penalidade, utilização do pátio, sustentabilidade e variabilidade. Os agentes baseados em PPO apresentaram resultados superiores, com destaque para o PPO Recorrente que possui maior capacidade de estruturação de informação proveniente de episódios passados. Todos os agentes se mostraram treináveis e com desempenho superior ao *baseline*. Os resultados mostram que o agente PPO Recorrente obteve um desempenho superior em várias métricas, tais como penalidade média (0,0216), utilização do pátio (0,1293), sustentabilidade (média de 17,43) e variabilidade (média de 17,59). Comparando os agentes inteligentes com o agente randômico, observou-se que o PPO Recorrente reduziu a penalidade em aproximadamente 92,6% e a utilização do pátio em cerca de 98,6% em comparação com o agente randômico.

As principais limitações do sistema dizem respeito ao simulador desenhado para uma única unidade fabril, sem interação com outras unidades. O motor de recebimento de demandas e matéria prima não negocia prazos com fornecedores e clientes. O processo de negociação, por si, já poderia aceitar um agente decisor capaz de aproximar o atraso no recebimento de matéria prima e rejeitar uma demanda tão logo fosse recebida. Tal agente ainda habilitaria o encaminhamento de demandas para outros agentes, com estoques e *workloads* mais favoráveis ao cumprimento de prazos.

Dessa forma, uma principal extensão para o trabalho é avançar para um abordagem multi-agentes, possibilitando o compartilhamento de dados para a busca de um equilíbrio global em termos de lucro, sustentabilidade e variabilidade.

Outros aspecto que pode aproximar mais a proposta de casos reais consiste em aperfeiçoar

os métricas de sustentabilidade e variabilidade. Sustentabilidade, por exemplo, pode ser medida em termos de demandas não produzidas que geraram receitas provenientes de economia de energia, liberação de espaço de pátio, dentre outros. O *baseline* para esse tipo de aplicação pode ser baseado na versão offline dos problemas de jobshop, habilitando uma comparação mais precisa entre sistemas de decisão baseados em Aprendizagem por Reforço e baseados em Programação Matemática.

Referências

AISSANI, N.; BELDJILALI, B.; TRENTESAUX, D. Efficient and effective reactive scheduling of manufacturing system using sarsa-multi-objective agents. In: **MOSIM'08: 7th Conference Internationale de Modelisation et Simulation**. [S.l.: s.n.], 2008. p. 698–707. Citado na página 26.

AZARAKHSH, S.; SAHEBI, H.; HOSSEINI, S. M. S. Design of a sustainable integrated crude oil manufacturing network with risk cover and uncertainty considerations: a case study. **Journal of Ambient Intelligence and Humanized Computing**, Springer, p. 1–14, 2021. Citado na página 19.

BAER, S.; BAKAKEU, J.; MEYES, R.; MEISEN, T. Multi-agent reinforcement learning for job shop scheduling in flexible manufacturing systems. In: IEEE. **2019 Second International Conference on Artificial Intelligence for Industries (AI4I)**. [S.l.], 2019. p. 22–25. Citado na página 28.

BELLEMARE, M. G.; NADDAF, Y.; VENESS, J.; BOWLING, M. The Arcade Learning Environment: An Evaluation Platform for General Agents. **Journal of Artificial Intelligence Research**, v. 47, p. 253–279, jun. 2013. ISSN 1076-9757. Disponível em: <https://www.jair.org/index.php/jair/article/view/10819>. Citado 2 vezes nas páginas 39 e 42.

BERNSTEIN, D. S.; GIVAN, R.; IMMERMAN, N.; ZILBERSTEIN, S. The complexity of decentralized control of markov decision processes. **Mathematics of operations research**, INFORMS, v. 27, n. 4, p. 819–840, 2002. Citado na página 26.

BIEWALD, L. Experiment tracking with weights and biases, 2020. **Software available from wandb. com**, v. 2, n. 5, 2020. Citado na página 63.

BORTOLINI, M.; GALIZIA, F. G.; MORA, C. Reconfigurable manufacturing systems: Literature review and research trend. **Journal of manufacturing systems**, Elsevier, v. 49, p. 93–106, 2018. Citado na página 31.

BOUAZZA, W.; SALLEZ, Y.; BELDJILALI, B. A distributed approach solving partially flexible job-shop scheduling problem with a q-learning effect. **IFAC-PapersOnLine**, Elsevier, v. 50, n. 1, p. 15890–15895, 2017. Citado na página 27.

BRINTRUP, A.; PAK, J.; RATINEY, D.; PEARCE, T.; WICHMANN, P.; WOODALL, P.; MCFARLANE, D. Supply chain data analytics for predicting supplier disruptions: a case study in complex asset manufacturing. **International Journal of Production Research**, Taylor & Francis, v. 58, n. 11, p. 3330–3341, 2020. Citado na página 57.

BROCKMAN, G.; CHEUNG, V.; PETTERSSON, L.; SCHNEIDER, J.; SCHULMAN, J.; TANG, J.; ZAREMBA, W. Openai gym. **arXiv preprint arXiv:1606.01540**, 2016. Citado na página 63.

CHANG, J.; YU, D.; HU, Y.; HE, W.; YU, H. Deep reinforcement learning for dynamic flexible job shop scheduling with random job arrival. **Processes**, MDPI, v. 10, n. 4, p. 760, 2022. Citado na página 20.

CUNHA, B.; MADUREIRA, A. M.; FONSECA, B.; COELHO, D. Deep reinforcement learning as a job shop scheduling solver: A literature review. In: SPRINGER. **International Conference on Hybrid Intelligent Systems**. [S.l.], 2018. p. 350–359. Citado na página 20.

ESTRADA-JIMENEZ, L. A.; PULIKOTTIL, T.; NIKGHADAM-HOJJATI, S.; BARATA, J. Self-organization in smart manufacturing—background, systematic review, challenges and outlook. **Ieee Access**, IEEE, v. 11, p. 10107–10136, 2023. Citado na página 31.

GABEL, T.; RIEDMILLER, M. Distributed policy search reinforcement learning for job-shop scheduling tasks. **International Journal of production research**, Taylor & Francis, v. 50, n. 1, p. 41–61, 2012. Citado na página 27.

GOREN, S.; SABUNCUOGLU, I. Optimization of schedule robustness and stability under random machine breakdowns and processing time variability. **IIE Transactions**, Taylor & Francis, v. 42, n. 3, p. 203–220, 2009. Citado na página 23.

GUH, R.-S.; SHIUE, Y.-R.; TSENG, T.-Y. The study of real time scheduling by an intelligent multi-controller approach. **International Journal of Production Research**, Taylor & Francis, v. 49, n. 10, p. 2977–2997, 2011. Citado na página 25.

GUPTA, J. N.; MAJUMDER, A.; LAHA, D. Flowshop scheduling with artificial neural networks. **Journal of the Operational Research Society**, Taylor & Francis, v. 71, n. 10, p. 1619–1637, 2020. Citado na página 26.

HECKMANN, I.; COMES, T.; NICKEL, S. A critical review on supply chain risk–definition, measure and modeling. **Omega**, Elsevier, v. 52, p. 119–132, 2015. Citado na página 57.

HO, N. B.; TAY, J. C.; LAI, E. M.-K. An effective architecture for learning and evolving flexible job-shop schedules. **European Journal of Operational Research**, Elsevier, v. 179, n. 2, p. 316–333, 2007. Citado na página 25.

HUBBS, C. D.; LI, C.; SAHINIDIS, N. V.; GROSSMANN, I. E.; WASSICK, J. M. A deep reinforcement learning approach for chemical production scheduling. **Computers & Chemical Engineering**, Elsevier, v. 141, p. 106982, 2020. Citado na página 27.

HUBBS, C. D.; PEREZ, H. D.; SARWAR, O.; SAHINIDIS, N. V.; GROSSMANN, I. E.; WASSICK, J. M. Or-gym: A reinforcement learning library for operations research problem. **CoRR**, abs/2008.06319, 2020. Disponível em: <https://arxiv.org/abs/2008.06319>. Citado na página 20.

JOYCE, J. M. Kullback-Leibler Divergence. In: LOVRIC, M. (Ed.). **International Encyclopedia of Statistical Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. p. 720–722. ISBN 978-3-642-04898-2. Disponível em: <https://doi.org/10.1007/978-3-642-04898-2_327>. Citado na página 32.

JUN, S.; LEE, S.; CHUN, H. Learning dispatching rules using random forest in flexible job shop scheduling problems. **International Journal of Production Research**, Taylor & Francis, v. 57, n. 10, p. 3290–3310, 2019. Citado na página 26.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014. Citado na página 43.

KOCHENDERFER, M. J.; WHEELER, T. A.; WRAY, K. H. **Algorithms for decision making**. Cambridge: Massachusetts Institute of Technology, 2022. ISBN 978-0-262-04701-2. Citado 2 vezes nas páginas 32 e 33.

KUHNLE, A.; RÖHRIG, N.; LANZA, G. Autonomous order dispatching in the semiconductor industry using reinforcement learning. **Procedia Cirp**, Elsevier, v. 79, p. 391–396, 2019. Citado na página 27.

KULLBACK, S.; LEIBLER, R. A. On Information and Sufficiency. **The Annals of Mathematical Statistics**, v. 22, n. 1, p. 79 – 86, 1951. Publisher: Institute of Mathematical Statistics. Disponível em: <https://doi.org/10.1214/aoms/1177729694>. Citado na página 32.

LANG, S.; BEHRENDT, F.; LANZERATH, N.; REGGELIN, T.; MÜLLER, M. Integration of deep reinforcement learning and discrete-event simulation for real-time scheduling of a flexible job shop production. In: IEEE. **2020 Winter Simulation Conference (WSC)**. [S.l.], 2020. p. 3057–3068. Citado na página 27.

LI, X.; GAO, L.; LI, X.; GAO, L. A hybrid genetic algorithm and tabu search for multi-objective dynamic jsp. **Effective Methods for Integrated Process Planning and Scheduling**, Springer, p. 377–403, 2020. Citado na página 24.

LIN, C.-C.; DENG, D.-J.; CHIH, Y.-L.; CHIU, H.-T. Smart manufacturing scheduling with edge computing using multiclass deep q network. **IEEE Transactions on Industrial Informatics**, IEEE, v. 15, n. 7, p. 4276–4284, 2019. Citado na página 27.

LIU, C.-L.; CHANG, C.-C.; TSENG, C.-J. Actor-critic deep reinforcement learning for solving job shop scheduling problems. **Ieee Access**, IEEE, v. 8, p. 71752–71762, 2020. Citado na página 27.

LIU, F.; WANG, S.; HONG, Y.; YUE, X. On the robust and stable flowshop scheduling under stochastic and dynamic disruptions. **IEEE Transactions on Engineering Management**, IEEE, v. 64, n. 4, p. 539–553, 2017. Citado na página 24.

LU, Y.; XU, X.; WANG, L. Smart manufacturing process and system automation – a critical review of the standards and envisioned scenarios. **Journal of Manufacturing Systems**, v. 56, p. 312–325, 2020. ISSN 0278-6125. Disponível em: <https://www.sciencedirect.com/science/article/pii/S027861252030100X>. Citado na página 19.

LUO, P. C.; XIONG, H. Q.; ZHANG, B. W.; PENG, J. Y.; XIONG, Z. F. Multi-resource constrained dynamic workshop scheduling based on proximal policy optimisation. **International journal of production research**, Taylor & Francis, v. 60, n. 19, p. 5937–5955, 2022. Citado na página 27.

LUO, S. Dynamic scheduling for flexible job shop with new job insertions by deep reinforcement learning. **Applied Soft Computing**, Elsevier, v. 91, p. 106208, 2020. Citado na página 27.

MAGANHA, I.; SILVA, C.; FERREIRA, L. M. D. Understanding reconfigurability of manufacturing systems: An empirical analysis. **Journal of Manufacturing Systems**, Elsevier, v. 48, p. 120–130, 2018. Citado 2 vezes nas páginas 30 e 31.

MALUS, A.; KOZJEK, D. et al. Real-time order dispatching for a fleet of autonomous mobile robots using multi-agent reinforcement learning. **CIRP annals**, Elsevier, v. 69, n. 1, p. 397–400, 2020. Citado na página 27.

MNIH, V.; BADIA, A. P.; MIRZA, M.; GRAVES, A.; LILLICRAP, T. P.; HARLEY, T.; SILVER, D.; KAVUKCUOGLU, K. Asynchronous Methods for Deep Reinforcement Learning. **CoRR**, abs/1602.01783, 2016. ArXiv: 1602.01783. Disponível em: <http://arxiv.org/abs/1602.01783>. Citado na página 41.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; GRAVES, A.; ANTONOGLOU, I.; WIERSTRA, D.; RIEDMILLER, M. Playing atari with deep reinforcement learning. **arXiv preprint arXiv:1312.5602**, 2013. Citado na página 54.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G.; PETERSEN, S.; BEATTIE, C.; SADIK, A.; ANTONOGLOU, I.; KING, H.; KUMARAN, D.; WIERSTRA, D.; LEGG, S.; HASSABIS, D. Human-level control through deep reinforcement learning. **Nature**, v. 518, n. 7540, p. 529–533, fev. 2015. ISSN 1476-4687. Disponível em: <https://doi.org/10.1038/nature14236>. Citado 3 vezes nas páginas 13, 39 e 40.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G. et al. Human-level control through deep reinforcement learning. **nature**, Nature Publishing Group, v. 518, n. 7540, p. 529–533, 2015. Citado na página 20.

MOUELHI-CHIBANI, W.; PIERREVAL, H. Training a neural network to select dispatching rules in real time. **Computers & Industrial Engineering**, Elsevier, v. 58, n. 2, p. 249–256, 2010. Citado na página 25.

NACHTIGALL, T.; MIRONCIKA, S.; TOMICO, O.; FEIJS, L. Designing ultra-personalized product service systems. **CoDesign**, Taylor & Francis, v. 16, n. 4, p. 274–292, 2020. Citado na página 18.

NG, A. Y.; HARADA, D.; RUSSELL, S. Policy invariance under reward transformations: Theory and application to reward shaping. In: **Icml**. [S.l.: s.n.], 1999. v. 99, p. 278–287. Citado na página 34.

OGUNSAKIN, R.; MEHANDJIEV, N.; MARÍN, C. A. Bee-inspired self-organizing flexible manufacturing system for mass personalization. In: SPRINGER. **International conference on simulation of adaptive behavior**. [S.l.], 2018. p. 250–264. Citado na página 19.

OLAFSSON, S.; LI, X. Learning effective new single machine dispatching rules from optimal scheduling data. **International Journal of Production Economics**, Elsevier, v. 128, n. 1, p. 118–126, 2010. Citado na página 26.

OUELHADJ, D.; PETROVIC, S. A survey of dynamic scheduling in manufacturing systems. **Journal of scheduling**, Springer, v. 12, p. 417–431, 2009. Citado 2 vezes nas páginas 23 e 24.

PANZER, M.; BENDER, B. Deep reinforcement learning in production systems: a systematic literature review. **International Journal of Production Research**, Taylor & Francis, v. 60, n. 13, p. 4316–4341, 2022. Citado na página 54.

PARK, H.-S.; TRAN, N.-H.; PARK, J.-W. Biologically inspired techniques for autonomous shop floor control. **New Technologies-Trends, Innovations and Research, C. Volosencu, Ed. South Korea: InTech**, 2012. Citado 2 vezes nas páginas 30 e 31.

PARK, J.; CHUN, J.; KIM, S. H.; KIM, Y.; PARK, J. Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. **International Journal of Production Research**, Taylor & Francis, v. 59, n. 11, p. 3360–3377, 2021. Citado na página 27.

PICKARDT, C.; BRANKE, J.; HILDEBRANDT, T.; HEGER, J.; SCHOLZ-REITER, B. Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness. In: IEEE. **Proceedings of the 2010 winter simulation conference**. [S.l.], 2010. p. 2504–2515. Citado na página 25.

PRIORE, P.; PARREÑO, J.; PINO, R.; GÓMEZ, A.; PUENTE, J. Learning-based scheduling of flexible manufacturing systems using support vector machines. **Applied Artificial Intelligence**, Taylor & Francis, v. 24, n. 3, p. 194–209, 2010. Citado na página 26.

QIN, Z.; LU, Y. Self-organizing manufacturing network: A paradigm towards smart manufacturing in mass personalization. **Journal of Manufacturing Systems**, Elsevier, v. 60, p. 35–47, 2021. Citado 4 vezes nas páginas 19, 20, 23 e 31.

QU, S.; WANG, J.; JASPERNEITE, J. Dynamic scheduling in modern processing systems using expert-guided distributed reinforcement learning. In: IEEE. **2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)**. [S.l.], 2019. p. 459–466. Citado na página 27.

RAFFIN, A.; HILL, A.; GLEAVE, A.; KANERVISTO, A.; ERNESTUS, M.; DORMANN, N. Stable-baselines3: Reliable reinforcement learning implementations. **Journal of Machine Learning Research**, v. 22, n. 268, p. 1–8, 2021. Disponível em: <http://jmlr.org/papers/v22/20-1364.html>. Citado na página 63.

RAUCH, E.; DALLASEGA, P.; MATT, D. T. Distributed manufacturing network models of smart and agile mini-factories. **International Journal of Agile Systems and Management**, Inderscience Publishers (IEL), v. 10, n. 3-4, p. 185–205, 2017. Citado na página 18.

REN, W.; YAN, Y.; HU, Y.; GUAN, Y. Joint optimisation for dynamic flexible job-shop scheduling problem with transportation time and resource constraints. **International Journal of Production Research**, Taylor & Francis, v. 60, n. 18, p. 5675–5696, 2022. Citado na página 24.

SCHULMAN, J.; MORITZ, P.; LEVINE, S.; JORDAN, M.; ABBEEL, P. **High-Dimensional Continuous Control Using Generalized Advantage Estimation**. arXiv, 2018. ArXiv:1506.02438 [cs]. Disponível em: <http://arxiv.org/abs/1506.02438>. Citado 2 vezes nas páginas 41 e 42.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. **Proximal Policy Optimization Algorithms**. arXiv, 2017. ArXiv:1707.06347 [cs]. Disponível em: <http://arxiv.org/abs/1707.06347>. Citado 4 vezes nas páginas 13, 41, 42 e 43.

SCHULMAN, J.; WOLSKI, F.; DHARIWAL, P.; RADFORD, A.; KLIMOV, O. Proximal policy optimization algorithms. **arXiv preprint arXiv:1707.06347**, 2017. Citado na página 54.

SELS, V.; GHEYSEN, N.; VANHOUCKE, M. A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. **International Journal of Production Research**, Taylor & Francis, v. 50, n. 15, p. 4255–4270, 2012. Citado na página 24.

SHI, D.; FAN, W.; XIAO, Y.; LIN, T.; XING, C. Intelligent scheduling of discrete automated production line via deep reinforcement learning. **International journal of production research**, Taylor & Francis, v. 58, n. 11, p. 3362–3380, 2020. Citado na página 27.

SHIUE, Y.-R. Data-mining-based dynamic dispatching rule selection mechanism for shop floor control systems using a support vector machine approach. **International Journal of Production Research**, Taylor & Francis, v. 47, n. 13, p. 3669–3690, 2009. Citado na página 26.

SILBERMAYR, L.; MINNER, S. A multiple sourcing inventory model under disruption risk. **International Journal of Production Economics**, v. 149, p. 37–46, 2014. ISSN 0925-5273. The Economics of Industrial Production. Citado na página 57.

SILVER, D.; HUBERT, T.; SCHRITTWIESER, J.; ANTONOGLOU, I.; LAI, M.; GUEZ, A.; LANCTOT, M.; SIFRE, L.; KUMARAN, D.; GRAEPEL, T. et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. **arXiv preprint arXiv:1712.01815**, 2017. Citado na página 53.

STRØM, M. A.; REHN, C. F.; PETTERSEN, S. S.; ERIKSTAD, S. O.; ASBJØRNSLETT, B. E.; BRETT, P. O. Combining design and strategy in offshore shipping. In: **Marine Design XIII**. [S.l.]: CRC Press, 2018. p. 147–162. Citado na página 53.

SUTTON, R. S.; BARTO, A. G. **Reinforcement learning: an introduction**. Second edition. Cambridge, Massachusetts: The MIT Press, 2018. (Adaptive computation and machine learning series). ISBN 978-0-262-03924-6. Disponível em: <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>. Citado 11 vezes nas páginas 32, 33, 34, 35, 36, 37, 38, 39, 40, 41 e 54.

TORN, I.; VANEKER, T. Mass personalization with industry 4.0 by smes: a concept for collaborative networks. **Procedia Manufacturing**, v. 28, p. 135–141, 2019. ISSN 2351-9789. 7th International conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV2018). Citado na página 18.

ĐURASEVIĆ, M.; JAKOBOVIĆ, D. A survey of dispatching rules for the dynamic unrelated machines environment. **Expert Systems with Applications**, Elsevier, v. 113, p. 555–569, 2018. Citado na página 24.

WANG, H.; CHENG, J.; LIU, C.; ZHANG, Y.; HU, S.; CHEN, L. Multi-objective reinforcement learning framework for dynamic flexible job shop scheduling problem with uncertain events. **Applied Soft Computing**, v. 131, p. 109717, 2022. ISSN 1568-4946. Citado na página 20.

WANG, H.; SARKER, B. R.; LI, J.; LI, J. Adaptive scheduling for assembly job shop with uncertain assembly times based on dual q-learning. **International Journal of Production Research**, Taylor & Francis, v. 59, n. 19, p. 5867–5883, 2021. Citado na página 26.

WANG, X.; JIN, Y.; SCHMITT, S.; OLHOFER, M. Recent advances in bayesian optimization. **ACM Computing Surveys**, ACM New York, NY, v. 55, n. 13s, p. 1–36, 2023. Citado na página 66.

WANG, Y.-C.; USHER, J. M. Application of reinforcement learning for agent-based production scheduling. **Engineering applications of artificial intelligence**, Elsevier, v. 18, n. 1, p. 73–82, 2005. Citado na página 26.

WANG, Y.-F. Adaptive job shop scheduling strategy based on weighted q-learning algorithm. **Journal of Intelligent Manufacturing**, Springer, v. 31, n. 2, p. 417–432, 2020. Citado na página 27.

WASCHNECK, B.; REICHSTALLER, A.; BELZNER, L.; ALTENMÜLLER, T.; BAUERNHANSL, T.; KNAPP, A.; KYEK, A. Optimization of global production scheduling with deep reinforcement learning. **Procedia Cirp**, Elsevier, v. 72, p. 1264–1269, 2018. Citado na página 27.

WECKMAN, G. R.; GANDURI, C. V.; KOONCE, D. A. A neural network job-shop scheduler. **Journal of Intelligent Manufacturing**, Springer, v. 19, p. 191–201, 2008. Citado na página 26.

WILLIAMS, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. **Reinforcement learning**, p. 5–32, 1992. Publisher: Springer. Citado na página 41.

XIONG, H.; FAN, H.; JIANG, G.; LI, G. A simulation-based study of dispatching rules in a dynamic job shop scheduling problem with batch release and extended technical precedence constraints. **European Journal of Operational Research**, Elsevier, v. 257, n. 1, p. 13–24, 2017. Citado na página 24.

YANG, S.; XU, Z. Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. **International Journal of Production Research**, Taylor & Francis, v. 60, n. 16, p. 4936–4953, 2022. Citado na página 27.

YSKA, D.; MEI, Y.; ZHANG, M. Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. In: SPRINGER. **Genetic Programming: 21st European Conference, EuroGP 2018, Parma, Italy, April 4-6, 2018, Proceedings 21**. [S.l.], 2018. p. 306–321. Citado na página 25.

ZANG, Z.; WANG, W.; SONG, Y.; LU, L.; LI, W.; WANG, Y.; ZHAO, Y. et al. Hybrid deep neural network scheduler for job-shop problem based on convolution two-dimensional transformation. **Computational intelligence and neuroscience**, Hindawi, v. 2019, 2019. Citado na página 26.

ZHANG, C.; SONG, W.; CAO, Z.; ZHANG, J.; TAN, P. S.; CHI, X. Learning to dispatch for job shop scheduling via deep reinforcement learning. **Advances in Neural Information Processing Systems**, v. 33, p. 1621–1632, 2020. Citado na página 20.

ZHANG, F.; MEI, Y.; NGUYEN, S.; ZHANG, M. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. **ieee transactions on cybernetics**, IEEE, v. 51, n. 4, p. 1797–1811, 2020. Citado na página 25.

ZHANG, F.; MEI, Y.; ZHANG, M. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In: SPRINGER. **AI 2018: Advances in Artificial Intelligence: 31st Australasian Joint Conference, Wellington, New Zealand, December 11-14, 2018, Proceedings 31**. [S.l.], 2018. p. 472–484. Citado na página 25.

ZHANG, F.; MEI, Y.; ZHANG, M. A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling. In: **Proceedings of the Genetic and Evolutionary Computation Conference**. [S.l.: s.n.], 2019. p. 347–355. Citado na página 25.

ZHANG, R.; SONG, S.; WU, C. Robust scheduling of hot rolling production by local search enhanced ant colony optimization algorithm. **IEEE Transactions on Industrial Informatics**, IEEE, v. 16, n. 4, p. 2809–2819, 2020. Citado na página 24.

ZHANG, T.; XIE, S.; ROSE, O. Real-time job shop scheduling based on simulation and markov decision processes. In: IEEE. **2017 Winter Simulation Conference (WSC)**. [S.l.], 2017. p. 3899–3907. Citado na página 26.

ZHANG, Y.; ZHU, H.; TANG, D.; ZHOU, T.; GUI, Y. Dynamic job shop scheduling based on deep reinforcement learning for multi-agent manufacturing systems. **Robotics and Computer-Integrated Manufacturing**, Elsevier, v. 78, p. 102412, 2022. Citado na página 20.

ZHENG, P.; XIA, L.; LI, C.; LI, X.; LIU, B. Towards self-x cognitive manufacturing network: An industrial knowledge graph-based multi-agent reinforcement learning approach. **Journal of Manufacturing Systems**, v. 61, p. 16–26, 2021. ISSN 0278-6125. Citado na página 18.

ZHOU, M. **Petri nets in flexible and agile automation**. [S.l.]: Springer Science & Business Media, 2012. v. 310. Citado na página 28.