UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

LUCAS VOLKMER HENDGES

DEEP Q-LEARNING PARA O CONTROLE SUPERVISÓRIO FLEXÍVEL DE SISTEMAS A EVENTOS DISCRETOS EM LARGA ESCALA

PATO BRANCO

2022

LUCAS VOLKMER HENDGES

DEEP Q-LEARNING PARA O CONTROLE SUPERVISÓRIO FLEXÍVEL DE SISTEMAS A EVENTOS DISCRETOS EM LARGA ESCALA

Deep Q-Learning on Supervisory Flexible Control of Discrete-Event Systems

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Orientador: Prof. Dr. Dalcimar Casanova
Coorientador: Prof. Dr. Marcelo Teixeira

PATO BRANCO

2022

LUCAS VOLKMER HENDGES

DEEP Q-LEARNING PARA O CONTROLE SUPERVISÓRIO FLEXÍVEL DE SISTEMAS A EVENTOS DISCRETOS EM LARGA ESCALA

Trabalho de Conclusão de Curso apresentado como requisito para obtenção do título de Bacharel em Engenharia de Computação da Universidade Tecnológica Federal do Paraná (UTFPR).

Data de Aprovação: 08 de Abril de 2022.

Prof. Dr. Dalcimar Casanova
Universidade Tecnológica Federal do Paraná

Prof. Dr. Marco Antônio de Castro Barbosa
Universidade Tecnológica Federal do Paraná

Prof. Dr. Rafael Cardoso
Universidade Tecnológica Federal do Paraná

PATO BRANCO

2022

Dedico este trabalho a todos os meus amigos e amigas que me acompanharam na jornada até aqui. Vocês que me fizeram acreditar que era possível mesmo quando eu não acreditava.

Mais que tudo, dedico-o também à minha mãe e irmã que estiveram sempre presentes nos momentos difíceis da jornada até aqui, e foram compreensivas quando precisei de espaço.

AGRADECIMENTOS

Agradeço ao meu orientador Dalcimar Casanova e a meu coorientador Marcelo Teixeira por me ajudarem na definição do escopo do trabalho e na sua escrita, e também ao Kallil Zielinski pela oportunidade de integrar o projeto que rendeu o tema para este trabalho.

De forma especial, agradeço a meus amigos Bruno Duarte, Fábio Kurpel, Juliana Sanguanini e Lucas Caldeira, que dispuseram de seus tempos para ler o texto diversas vezes, e me ajudaram a melhorar sua qualidade, trazendo críticas sob pontos de vista diferentes.

RESUMO

Sistemas industriais, como os de manufatura, podem ser tipicamente associados a um perfil de sistemas que evoluem a eventos que ocorrem assincronamente, em pontos discretos no tempo, aos quais dá-se o nome de Sistemas a Eventos Discretos (SEDs). Basicamente, um SED absorve a ideia de que o tempo tomado entre uma ação e outra (eventos) no sistema pode ser desprezado em detrimento à razão lógica como os eventos ocorrem. Ou seja, um SED prioriza o mapeamento lógico da ocorrência de eventos no sistema e as operações sobre SEDs consistem tão somente em processar saídas lógicas que orquestram com precisão o arranjo físico do sistema. Do ponto de vista de controle, uma das abordagens que se dedica a calcular sequências lógicas ótimas para SEDs, levando em conta propriedades como controlabilidade e ausência de *deadlocks*, é a Teoria do Controle Supervisório (TCS), que é processada sobre modelos de SEDs expressos como Máquinas de Estados Finitos (MEFs). Como a TCS estrutura um método formal pautado na precisão do cálculo da lógica de controle, ela consequentemente beneficia propriedades como segurança e robustez da ação de controle, o que é positivo em muitos ambientes industriais. Em contrapartida, a TCS se mostra limitada quando o objetivo é flexibilizar a lógica de controle, total ou parcialmente, e processar eventos que não necessariamente tenham uma natureza exata de ocorrência. Para esses casos, ou seja, quando um SED possui alguns eventos com natureza probabilística, outros não, uma alternativa é integrar a ação exata, calculada via TCS, com processamento inteligente. Nesse caso, é fundamental que a inteligência artificial se aplique a apenas certos eventos do sistema, sob pena de comprometer os aspectos exatos dos eventos que são de interesse preservar como tais. Este trabalho, portanto, propõe o particionamento do conjunto de eventos de um SED em determinísticos e probabilísticos, a fim de processá-los distintamente. Enquanto os determinísticos são processados via TCS, como usual, os demais são tratados a partir da conversão de um modelo (MEF) controlado de um SED em um Processo Decisório de Markov (PDM), sobre o qual aplica-se uma abordagem de *Deep Q-Learning*. A escolha de *Deep Q-Learning* é, sobretudo, por seu potencial para processar quantidades consideráveis de estados, o que se aproxima do perfil de aplicações reais de SEDs. O resultado dessa integração é uma abordagem mais flexível de controle, em comparação à TCS clássica, que se mostra capaz de maximizar recompensas de certas ações do sistema ao passo em que preserva o rigor formal do controlador.

Palavras-chave: Controle; Automação; Deep-Q Learning; Modelagem; Manufatura.

ABSTRACT

Industrial Systems, as the manufacturing ones, can be tipically associated with a system profile that evolves with asynchronous events, in discrete time instants, which we name Discrete Event Systems (DES). Basically, a DES synthesizes the idea that the time between two actions (or events) in the system can be despised due to the way the events occur. That means that DES prioritizes the logical mapping of the occurence of events in the system, and the operations on them consist only in processing logical outputs that orchestrate precisely the system's physical structure. From the control perspective, one of the approaches that is dedicated on calculating optimal logical sequences to DES, taking into account properties such as the controllability and deadlock absence, is the Supervisory Control Theory (SCT), that is processed over DES models, expressed as Finite State Machines (FSM). As the SCT structures a formal method focused in the precision of the control logic's computation, as a consequence, it benefits propertites such as safety and robustness of the control action, which is positive in many industrial environments. On the other hand, SCT fails on its limitation on totally or partially flexibilizing this logic, by processing events that are not necessarily exact. For these cases, i.e., when a DES has some events with probabilistic nature, intelligent processing approaches can be more suitable. For that, it is fundamental that the artificial perception be applied to certain events, but not to all of them, with the risk of compromising the exact control imposed by the SCT, which we want to preserve. Thus, this work proposes the partition of the event set of a DES on the deterministic and probabilistic subsets, in order to process them distinctly. While the deterministic ones are processed through SCT as usual, the ones that remain are treated through the conversion of a controlled model (FSM) of a DES in a Markov's Decision Process (MDP), over which a *Deep Q-Learning* approach is applied. The choice of *Deep Q-Learning* is, over all, for its potential on processing big state sets, which is closer to the real applications of DES. The result of this integration is a more flexible control approach, in comparison to the classical SCT, making it possible to maximize certain system rewards while preserving the formality of the controller.

Keywords: Control; Automation; Deep$_Q Learning$; $Modelling$; $Manufacturing$.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

SIGLAS

UTFPR Universidade Tecnológica Federal do Paraná

SUMÁRIO

1 INTRODUÇÃO

Sistemas de manufatura buscam transformar a matéria-prima em bens de consumo, integrando pessoas, equipamentos e tecnologias (GROOVER, 2011; MANU *et al.*, 2018). Para isso, a engenharia cumpre um papel decisivo ao prover métodos, modelos e ferramentas que, juntos, integram o meio físico ao lógico e permitem responder ao ambiente automaticamente, dentro do tempo e com a precisão esperada.

Do ponto de vista de engenharia, um sistema de manufatura pode ser visto conforme a sua natureza de evolução no tempo. Essa evolução, quando observada com certo grau de abstração, ocorre de forma assíncrona no tempo, a partir da ocorrência de *eventos*, que modelam a ocorrência de sinais elétricos no sistema e desencadeiam transições entre os estados. Aos sistemas que compartilham dessa dinâmica, dá-se o nome de *Sistemas a Eventos Discretos* (SEDs) e essa classe envolve, por exemplo, sistemas de tráfego, logística e robótica.

Em um SED, os eventos que guiam o sistema podem ser caracterizados conforme o nível de determinismo como ocorrem na malha física. Quando tais eventos apresentam uma natureza determinística, eles podem ser modelados matematicamente através de *Máquinas de Estados Finitos* (MEFs) (SIPSER, 2013), e controlados a partir de mecanismos formais de síntese, como a *Teoria do Controle Supervisório* (TCS) (P. J. G. RAMADGE, 1989). A essência da TCS é aplicar uma metodologia exata, ideal para eventos determinísticos, para sintetizar controladores ótimos, ou seja, maximamente permissivos, controláveis e não bloqueantes, recebendo como entrada modelos que representam os componentes do sistemas e suas especificações de controle.

Apesar de a TCS ser reconhecida como um método eficaz para a obtenção de controladores para SEDs, essa teoria vem sendo desafiada ao longo dos anos a diversas adaptações para lidar com o dinamismo que emerge dos sistemas modernos de produção. De fato, ambientes fabris modernos tendem a adotar modelos flexíveis de operação, gerando diferentes produtos na mesma malha física e fazendo uso de modos distintos de atuação (PARK; FEBRIANI, 2019; HARRISON; VERA; AHMAD, 2016). Essa característica força o controlador a reconhecer diferentes contextos para associá-los ao esquema de atuação, englobando decisões de negócio como eventos do sistema, o que remove parte da natureza determinística de como os eventos são elegíveis. O perfil estocástico associado à ocorrência de eventos é uma particularidade que não converge diretamente ao *framework* original da TCS, requerendo complementos tangentes à infraestrutura de modelagem, síntese e implementação de controladores dessa natureza.

Algumas abordagens na literatura buscam enriquecer a TCS com níveis de flexibilidade,

tais como o uso de sensores distinguidores (CURY, J. E. *et al.*, 2015), a adoção de modelos com variáveis de estado (TEIXEIRA; HERING DE QUEIROZ; CURY, J., 2015), a aplicação de modelos reconhecedores de contextos (LUCAS SILVA; RIBEIRO; TEIXEIRA, 2017), a simplificação via tratamento computacional modular (TEIXEIRA; CURY, J.; QUEIROZ, 2018), etc. Tais abordagens agregam-se ao perfil determinístico da TCS e permitem manipular contextos ao mesmo tempo em que preservam o teor formal exato, tão importante do ponto de vista de precisão da ação de controle. No entanto, elas exploram apenas a natureza determinística da ocorrência de eventos, por meio de estratégias lógicas, também determinísticas. Ou seja, elas consistem em instanciar as ocorrências de eventos determinísticos e garantir a semântica de contexto de modo exato, não considerando a natureza estocástica de alguns eventos que podem ser agregados a tais sistemas, e.g. a inclusão da modelagem de decisões de negócio como eventos intrísecos a ele, tampouco suas mudanças dinâmicas de significado ao longo da evolução de um SED.

Buscando melhorar o poder semântico de modelos de SEDs, a literatura ainda apresenta soluções de controle como por meio de lógica *fuzzy* (WANG *et al.*, 2021), Redes de Petri (MURATA, 1989a) e associações com outros métodos, como as Simheurísticas (LATORRE-BIEL *et al.*, 2021) e Metaheurísticas (OSAKI; HOSOE; HAGIWARA, 2020). Também a área de Inteligência Artificial e *Machine Learning* (RUSSELL; NORVIG, 2002) apresenta soluções complementares ao controle estocástico de SEDs, como por meio de algoritmos de clusterização *K-Nearest Neighbour* (KNN) (HARROU; ZEROUAL; SUN, 2020), árvores de decisão (GONON; BIMBOT; GRIBONVAL, 2009), *Naive Bayes* (SINGH *et al.*, 2020) e Aprendizagem por Reforço (AR). Métodos de AR, em particular, são executados sobre um *ambiente* de aprendizado. Os ambientes de aprendizado partem de um Processo Decisório de Markov (PDM) (SUTTON; BARTO, 2018), que é uma estrutura capaz de mapear estados e ações do ambiente.

Por ser possível converter um SED em um PDM, pode-se lidar com SEDs utilizando AR. Isso é feito separando os eventos determinísticos dos SEDs para terem sua estrutura de controle processada por meio da TCS, ao passo que os eventos probabilísticos, por sua vez, são tratados por algoritmos como o *State-Action-Reward-State-Action* (SARSA) e sua variante de múltiplos passos *N-Step State-Action-Reward-State-Action* (*N-step* SARSA), conforme feito em Zielinski.

Para sistemas cuja modelagem tem um número reduzido de estados, métodos como o de Kallil M.C. Zielinski *et al.* (2021) são viáveis, do ponto de vista de custo computacional. Porém, essa abordagem não escala na mesma ordem ao se manipular espaços de estados muito grandes.

Isso ocorre pois os algoritmos SARSA e *N-step* SARSA são tabulares. A chamada *Q-table* (Tabela Q), necessária nessas abordagens, deve ser armazenada e processada sistematicamente, guardando valores de recompensas atuais para cada uma das tuplas $(estado, evento)$. Em espaços de estados maiores, compatíveis SEDs reais, o armazenamento de tabelas Q pode limitar o tratamento computacional, em termos de memória e de tempo de execução, não raramente inviabilizando o tratamento do problema.

Uma alternativa a (ZIELINSKI, K. M. *et al.*, 2021) vem do campo do *Deep Learning*, cujos métodos baseados em redes neurais artificiais se mostram adequados para lidar com espaços de estados extensos (ZHANG *et al.*, 2021). De interesse particular deste trabalho, a classe de aprendizado profundo por reforço, *Deep Q-Learning*, consiste em uma rede neural capaz de prever os valores da Tabela Q indiretamente, utilizando aproximações, dispensando assim a necessidade de armazená-la na memória. Essa abordagem pode ser integrada ao tratamento de eventos probabilísticos de SEDs cujo espaço de estados é amplo, o que é feito neste trabalho.

Desta forma, definem-se os objetivos do trabalho.

1.1 OBJETIVOS

Objetivo Geral: Estruturar um método de *Deep Q-Learning* para o tratamento de eventos probabilísticos de um SED integrando a abordagem ao controle supervisório responsável pelo tratamento dos eventos não probabilísticos do sistema, testando a abordagem em um exemplo de sistema de manufatura simulado.

Objetivos Específicos:

- Definir um exemplo de processo a ser modelado e obter seu modelo em MEF;
- Obter o controlador para o sistema a partir da TCS;
- Implementar a solução inteligente de *Deep Q-Learning* para o sistema;
- Comparar o desempenho do sistema com a solução proposta ao resultado obtido com o algoritmo *N-step* SARSA em trabalho anterior, em um sistema com espaço de estados reduzido (ZIELINSKI, K. M. *et al.*, 2021).
- Comparar o desempenho com e sem o método desenvolvido ao aplicá-lo em um sistema com espaço de estados maior.

2 REFERENCIAL TEÓRICO

Os sistemas industriais de manufatura, em sua maioria, têm um comportamento que pode ser abstraído considerando apenas seus *estados*, *transições* e *eventos*. Um estado representa uma situação bem definida dentro da planta industrial, como a operação de uma máquina: se esta está operando ou está parada. Já as transições são funções que levam o sistema de um estado a outro, ao passo que os eventos são acontecimentos que rotulam e desencadeiam as transições. Por exemplo, o acionamento de um botão ou a leitura de passagem de uma peça por um sensor são eventos que podem implicar em transições entre estados, como um braço robótico iniciar sua operação de retirada de uma peça de uma plataforma. O Exemplo 1 ilustra uma situação dessa natureza.

Exemplo 1 *Para fins ilustrativos, considera-se a versão parcial de um sistema industrial, composto por duas plataformas,* A *e* B*, e uma esteira* C*. É definido que cada uma dessas plataformas pode comportar uma única peça. Existem dois braços robóticos no sistema, sendo que o primeiro deles, à esquerda, ao detectar a presença de uma peça em* A *por meio de um sensor, a move para* B*. O segundo braço robótico, à direita, detecta a entrada da peça na plataforma* B*, também por meio de um sensor, e a retira posteriormente, a encaminhando para a esteira* C*. Considera-se aqui que a peça é retirada do sistema quando entra na esteira* C*. Esse comportamento pode ser observado na Figura 1.*

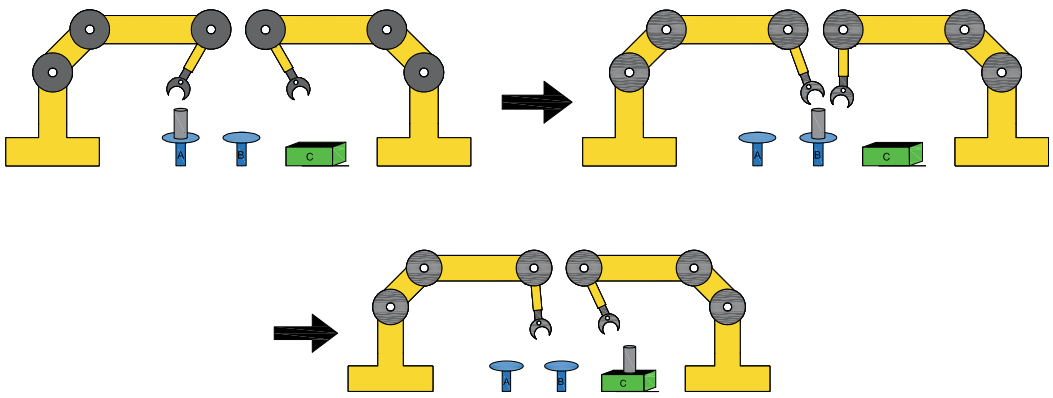


Figura 1 – Ilustração do exemplo 1.

Para o sistema ilustrado, considera-se que o braço robótico esquerdo acaba de terminar sua operação de colocar uma peça na plataforma B quando ocorre a chegada de uma segunda peça na plataforma A. Neste caso, é impossível que a peça em A seja movida para B, sem que

antes o braço robótico da direita inicie sua operação para retirar a peça em B, deslocando-a para a esteira C, retirando-a do sistema. Ou seja, existe uma restrição quanto à ordem em que as transições precisam ocorrer. A situação encontra-se ilustrada na Figura 2.

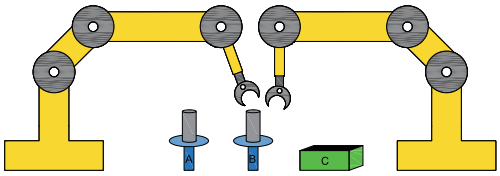


Figura 2 – Ilustração de situação problemática para o sistema do exemplo 1

Outro problema possível seria se o sistema estivesse suscetível a falhas, como o caso de uma máquina quebrar: isso deve ser considerado na modelagem. Ainda, se o sistema contiver um fluxo que permita a passagem de uma mesma peça pelo setor múltiplas vezes, mas que após um número limite ela seja descartada, isso também precisa ser considerado.

Uma vez que os sistemas industriais, do mais simples ao mais complexo, podem ser abstraídos ao nível de estados, eventos e transições, será definido, nas seções subsequentes, o formalismo matemático que possibilita esta modelagem.

2.1 MÁQUINAS DE ESTADOS FINITOS E O CONTROLE SUPERVISÓRIO

Um autômato ou máquina de estados finitos (MEF) (MENEZES, 1998) é um formalismo matemático que se define como uma tupla

$$G = \langle \Sigma, Q, q^{\circ}, Q^{\omega}, \rightarrow \rangle,$$

em que G representa o nome da máquina, Σ é um conjunto finito de *eventos*, também chamado de alfabeto; Q é um conjunto finito de *estados*; $q^{\circ} \in Q$ representa o *estado inicial* da máquina, sendo este, único; $Q^{\omega} \subseteq Q$ é o conjunto de *estados marcados*, geralmente representando estados em que alguma tarefa foi concluída, ou estados finais; e $\rightarrow \subseteq Q \times \Sigma \times Q$ representa a *função parcial de transição*.

Eventos retirados de Σ podem ser concatenados, compondo o que se chama de *strings* de eventos. O conjunto Σ^* inclui todas as *strings* finitas de eventos que se pode formar utilizando os eventos de Σ, incluindo a *string* vazia ε. A transição entre dois eventos $q, q' \in Q$ decorrente da ocorrência do evento $\sigma \in \Sigma$ é representada por $q \xrightarrow{\sigma} q'$. Quando o evento σ encontra-se desabilitado, a transição não é possível, o que é representado por $q \xrightarrow{\sigma} q'$. Partido-se do estado

inicial da máquina, a ocorrência de uma *string* s de eventos que levam a máquina ao estado q é representa por $G \xrightarrow{s} q$.

SEDs em geral contêm vários componentes independentes, sendo que cada um pode ser modelado por uma MEF. Nesse caso, é conveniente relacionar dois ou mais modelos de componentes do SED em uma única representação que sincroniza seus comportamentos, a *composição síncrona* (denotada por $\|$).

Dadas duas MEFs, $G_1 = \langle \Sigma_1, Q_1, q_1^\circ, Q_1^\omega, \rightarrow_1 \rangle$ e $G_2 = \langle \Sigma_2, Q_2, q_2^\circ, Q_2^\omega, \rightarrow_2 \rangle$, define-se $G_1 \| G_2 = \langle \Sigma_1 \cup \Sigma_2, Q_1 \times Q_2, (q_1^\circ, q_2^\circ), Q_1^\omega \times Q_2^\omega, \rightarrow \rangle$, em que o conjunto de transições $\rightarrow$ é construído conforme abaixo:

- $(q_1, q_2) \xrightarrow{\sigma} (q_1', q_2')$ se $\sigma \in \Sigma_1 \cap \Sigma_2, q_1 \xrightarrow{\sigma} q_1', q_2 \xrightarrow{\sigma} q_2'$;
- $(q_1, q_2) \xrightarrow{\sigma} (q_1', q_2)$ se $\sigma \in \Sigma_1 \setminus \Sigma_2, q_1 \xrightarrow{\sigma} q_1'$;
- $(q_1, q_2) \xrightarrow{\sigma} (q_1, q_2')$ se $\sigma \in \Sigma_2 \setminus \Sigma_1, q_2 \xrightarrow{\sigma} q_2'$;
- indefinido.

Em resumo, a operação de composição síncrona de duas ou mais MEFs, combina os comportamentos individuais de todas as máquinas compostas, criando um modelo equivalente. Os eventos compartilhados por cada planta G_j são sincronizados, e os demais são intercalados, tornando possível a totalidade de sequências de eventos que o sistema pode apresentar.

A composição do sistema é representada por

$$G = \|_{j \in J} G_j$$

em que $J = \{1, \cdots, m\}$ representa o conjunto de componentes do sistema.

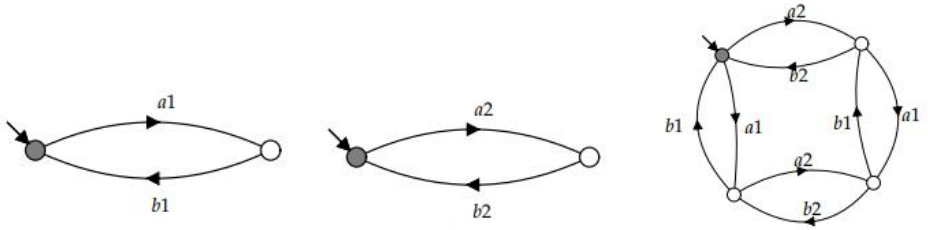
A planta após a composição mapeia como os componentes do sistema evoluem em paralelo, sem restrições. Na prática, espera-se que a planta seja coordenada, para que seus componentes interajam da maneira desejada.

Considerando novamente o Exemplo 1, a Figura 3 representa a modelagem de cada um dos braços e a composição das plantas.

A Figura 3(a) representa a modelagem do comportamento do braço robótico da esquerda, que move a peça de A para B, sendo esta planta chamada de $G1$. O evento $a1$ representa o início da operação deste braço, e o evento $b1$ representa seu fim. De forma análoga, a Figura 3(b) representa a mesma modelagem para o segundo braço robótico, chamado de $G2$, que move a peça de B para C, em que o evento $a2$ representa o início da operação do braço e o evento $b2$ representa seu fim. Por fim, a Figura 3(c) representa a composição entre $G1$ e $G2$.

Os estados inicias de todas as plantas $G1$, $G2$ e $G1||G2$ são aqueles em que há uma

seta que não provém de nenhum outro estado anterior, e, neste caso, são estados marcados.



(a) Planta $G1$ (b) Planta $G2$ (c) Composição das plantas $(G1||G2)$

Figura 3 – Modelagem em MEFs **do sistema do Exemplo 1.**

Os SEDs sempre apresentam restrições de eventos em determinados estados, e após feito o modelo dos componentes da planta, deve-se coordenar esses componentes, impondo a ordem lógica que se espera. Para isso, usa-se um modelo complementar, chamado *especificação*, sendo este responsável por restringir o comportamento do sistema para apenas o desejado. Isso é feito através da composição da planta com a especificação.

Uma especificação geral é representada por

$$E = \|_{i \in I} \, E_i \, ,$$

com $i \in I = \{1, \cdots, n\}$, em que n é o número de especificações individuais, e E é a composição síncrona delas.

Pensando na planta apresentada, considere que se deseje controlá-la, a fim de obter seu comportamento restrito à ordem lógica em que os eventos devem ocorrer. Isto é apresentado na Figura 4.

Partindo-se do estado inicial, é possível perceber que um evento $a2$ está desabilitado sem que antes ocorra um evento $b1$. Em termos práticos, isso significa que o braço robótico da direita não pode iniciar sua operação enquanto o braço robótico da esquerda não tiver concluído o transporte da peça de A a B.

De forma similar, a especificação apresentada impede que um evento $b1$ ocorra antes que o evento $a2$ tenha sido concluído, o que implica que o braço robótico da esquerda só pode colocar uma nova peça na plataforma B após a peça anterior ter sido retirada.

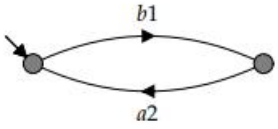


Figura 4 – Modelagem da especificação.

Quando se projeta uma estrutura de controle utilizando especificações sobre uma planta modelada com MEFs, pode-se expressar o comportamento do sistema a partir da composição da G com as especificações E, denotado

$$K = G \| E.$$

K representa o comportamento controlado do sistema, tal como foi projetado via engenharia. Entretanto, para que forme definitivamente o sistema de controle, K deve deter algumas propriedades funcionais, dentre elas o fato de ser não-bloqueante e de reconhecer a natureza dos eventos quanto à controlabilidade. Para isso, define-se:

Definição 1: Uma MEF $G = \langle \Sigma_G, Q_G, q_G^\circ, Q_G^\omega, \rightarrow_G \rangle$ é não-bloqueante se, para toda e qualquer *string* de eventos $s \in \Sigma_G^*$, $G \xrightarrow{s} q$ implica que $\exists\, t \in \Sigma_G^*$ tal que $q \xrightarrow{t} q'$, para algum $q' \in Q_G^\omega$.

Isso equivale dizer que, partindo do estado inicial q°, é possível alcançar algum estado marcado.

Já o conceito de *controlabilidade* refere-se à possibilidade em desabilitar, diretamente, alguns eventos do sistema, e à impossibilidade de desabilitar outros, devido à natureza intrínseca desses. Desta forma, os eventos que se pode desabilitar, são chamados controláveis, ao passo que os que não se pode desabilitar, são chamados não-controláveis. Por exemplo, um evento que envolve um sensor fazer a detecção da passagem de uma peça em uma esteira costuma ser visto como não-controlável por não ser possível desabilitar diretamente sua ocorrência, ao passo que a ação de início de operação de uma máquina pode ser vista como controlável. Na TCS, a controlabilidade é reconhecida a partir do particionamento do conjunto de eventos Σ_G em $\Sigma_G = \Sigma_G^{\mathrm{c}} \cup \Sigma_G^{\mathrm{u}}$, em que Σ_G^{c} e Σ_G^{u}, que representam o subconjunto dos eventos controláveis e não-controláveis, respectivamente. Sendo assim, a controlabilidade pode ser definida como a seguir.

Definição 2: Sejam $G = \langle \Sigma_G, Q_G, q_G^\circ, Q_G^\omega, \rightarrow_G \rangle$ e $K = \langle \Sigma_K, Q_K, q_K^\circ, Q_K^\omega, \rightarrow_K \rangle$ duas MEFs tal que $K \subset G$. K é dito controlável em relação a G se, para toda e qualquer $s \in \Sigma_G^*$ e todo $\mu \in \Sigma_G^{\mathrm{u}}$, se

$$K \xrightarrow{s} q_K \qquad \text{e} \qquad G \xrightarrow{s} q_G \xrightarrow{\mu} q'_G$$

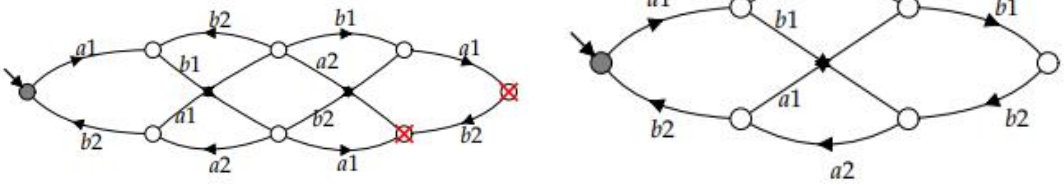
então existe $q'_K \in Q_K$, tal que

$$K \xrightarrow{s} q_K \xrightarrow{\mu} q'_K.$$

Ou seja, se após uma *string* s, um evento não-controlável μ for possível na planta G, então este evento também deve ser possível na especificação K.

A partir desses conceitos, a *Teoria do Controle Supervisório* (P. J. G. RAMADGE, 1989) estabelece o cálculo matemático de um controlador, ou *supervisor*, máximo denotado por sup$C(K,G)$. Esse supervisor é o maior submodelo de K, sendo assim maximamente permissivo (ou seja, bloqueia tão somente os eventos que as especificações restringem em cada estado), que adicionalmente detém a propriedade de ser satisfazer as especificações impostas por K de forma controlável. Se sup$C(K,G)$ for adicionalmente não-bloqueante, é chamado de um controlador ótimo.

A Figura 5(a) apresenta o controlador K, que é a composição das plantas do sistema com as especificações, e a Figura 5(b) apresenta o supervisor sup$C(K,G)$ para o Exemplo 1. Note que o supervisor elimina os estados de K que são considerados maus estados, ou seja, estados que não se deseja atingir, ou por violarem propriedades de controlabilidade, permitindo a ocorrência de eventos que as especificações proibem, ou por serem bloqueantes, resultando no sistema terminar em um estado, não marcado, do qual não consegue sair. Ao eliminá-los, é obtido o comportamento maximamente permissivo do sistema.



(a) Controlador K para o Exemplo 1. (b) Supervisor sup$C(K,G)$ para o Exemplo 1.

Figura 5 – Comparação de K e sup$C(K,G)$.

A abordagem inspirada em TCS é adequada para sistemas cujos eventos possuem natureza exata, devido às MEFs serem determinísticas. Em sistemas industriais, no entanto, muitas vezes existem eventos com caráter probabilístico de ocorrência, o que limita a aplicação ordinária de controle supervisório. Na Figura 6, foi incluído ao exemplo 1 um dispositivo acima da esteira C capaz de pintar de vermelho a peça que está sobre a esteira. No entanto, para este exemplo, assume-se que nem todas as peças serão pintadas: pintar ou não a peça é uma decisão externa ao sistema, que não compete ao controlador – o que se faz é a modelagem dessa decisão como um evento da planta.

O evento probabilístico, i.e., não determinístico, que a imagem aborda é o evento de início da operação do dispositivo que faz a pintura da peça. Neste caso, a TCS não será suficiente para tratar do problema, quando ele é descrito a partir desse prisma. Alternativamente, sua

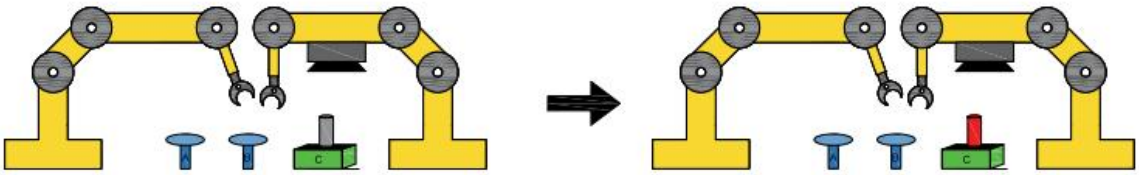


Figura 6 – Inclusão de evento probabilístico na planta.

integração com abordagens inteligentes, como a aprendizagem por reforço, pode auxiliar no tratamento destes eventos.

2.2 APRENDIZADO POR REFORÇO

O *aprendizado por reforço* (AR) é um paradigma no qual um *agente* busca melhorar seu desempenho baseado nas recompensas que recebe ao interagir com um *ambiente*, aprendendo uma *política* ou sequência de *ações* (KAELBLING; LITTMAN; MOORE, 1996; SUTTON; BARTO, 2018).

A cada passo de tempo t, este agente observa seu estado atual $s_t \in S$ e escolhe uma ação $a_t \in A$ para ser executada conforme a política π, que leva o agente ao estado s_{t+1}. A cada par estado-ação (s_t, a_t), o agente recebe um sinal de reforço $R(s_t, a_t) \rightarrow \mathbb{R}$ do ambiente, como recompensa por executar a ação a_t no estado s_t.

Formalmente, o método de AR pode ser exposto como um *Processo Decisório de Markov* (PDM) (PUTERMAN, 1994), definido como a quíntupla $M = \langle S,A,T,R,\gamma \rangle$, onde:

- $S = \{s_1, \cdots, s_n\}$ é o conjunto finito de estados do ambiente;
- $A = \{a_1, \cdots, a_n\}$ é o conjunto finito de ações que o agente pode executar;
- $T : S \times A \rightarrow \Pi(S)$ é uma função de transição entre estados onde: $\Pi(S)$ é uma distribuição de probabilidade sobre o conjunto de estados S; e $T(s_{t+1}, s_t | a_t)$ é a probabilidade de uma transição de s_t para s_{t+1} ocorrer com uma ação a_t.
- $R : S \times A \rightarrow \mathbb{R}$ é uma função que define a recompensa recebida pelo agente por selecionar uma ação a_t em um estado s_t num instante t;
- $\gamma \in [0,1]$ é um fator de desconto que determina quão longe no futuro um agente de AR procura por recompensas em comparação com seu futuro imediato.

Um episódio e_{MDP} pode ser descrito como uma sequência de estados, ações e recompensas (i.e. a experiência) adquirida por um agente a partir do estado inicial do ambiente até um estado terminal, e pode ser definida como:

$$e_{MDP} = s_0,a_0,r_0,s_1,a_1,r_1,\cdots,s_{n-1},a_{n-1},r_{n-1},sn\,, \tag{1}$$

em que s_i representa o i^o estado, a_i a i^a ação, e r_i a i^a recompensa.

Então, a quantidade de recompensa futura em qualquer instante t de tempo é dada por:

$$R_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots + \gamma^{n-t} r_n\,. \tag{2}$$

Lembrete 1 *O fator de desconto* $\gamma \in [0,1]$ *expressa quão seriamente o agente leva suas futuras ações em conta. Valores próximos de* 0 *representam uma estratégia que foca em recompensas em estados próximos, levando menos em consideração valores de recompensa em estados mais distantes. Se o ambiente é determinístico,* γ *pode ser escolhido como* 1*, pois as mesmas ações sempre resultam nas mesmas recompensas (LAPAN, 2018).*

Um agente de AR implementa, em cada passo de tempo, uma função que mapeia uma ação para cada estado no ambiente. Essa função é chamada de *política* do agente, que é denotada por π. Existem diversas políticas possíveis para um agente seguir, mas um caso particular é a *política ótima*, denotada por π^*. Este é um modelo em que as ações obtidas pelos agentes são melhores ou iguais às demais políticas π (SUTTON; BARTO, 2018).

Métodos como o *State-Action-Reward-State-Action* (SARSA) estimam uma assim chamada *função Q* para a política π no estado atual e em todos os demais estados e ações do ambiente. O procedimento consiste em determinar $Q_t^\pi(s_t,a_t)$, ou seja, em um determinado instante de tempo t, deseja-se obter o valor Q da recompensa geral obtida estando no estado s e tomando a ação a. O agente interage com o ambiente a fim de melhorar a aproximação dos valores Q. O resultado obtido é a chamada tabela Q, que mapeia qual a melhor ação a ser tomada em cada estado do ambiente, baseado na recompensa que se obtém ao tomá-la. Este é um método reconhecido por sua simplicidade para ser implementado ao mesmo tempo que é eficaz. À classe de métodos como o SARSA, que realizam uma busca exaustiva pelo ambiente de forma a obter uma *política ótima* π^*, se dá o nome de "Diferença Temporal".

No entanto, ao tentar lidar com sistemas com espaços de estados grandes, métodos tabulares como este podem não ser eficazes pois a tabela de valores a ser armazenada torna-se maior, o que implica em uma complexidade de espaço diretamente proporcional ao tamanho do espaço de estados da planta modelada. Por isso, abordagens que utilizam redes neurais podem ser mais adequadas.

2.3 DEEP LEARNING

Métodos de *deep learning* são baseados em *redes neurais* e têm como base o conceito de aprendizado de representação (ZHANG *et al.*, 2021).

Os neurônios desempenham o papel de calcular as probabilidades de uma entrada e gerar cada uma das possíveis saídas do sistema, e, baseado na etapa de treinamento com dados rotulados, a rede neural define os pesos que cada um dos neurônios tem neste processo, interagindo uns com os outros através das múltiplas camadas que a rede neural pode ter (ZHANG *et al.*, 2021).

A principal diferença prática entre métodos clássicos de aprendizado de máquina e *deep learning* é a forma com que são apresentados e representados os dados. No primeiro caso, é necessário obter diversas características sobre o problema (RASCHKA, 2015), citando a clássica base Iris, em que foram medidas largura e altura de pétalas e sépalas de flores do Gênero Iris, para então rotular os dados de suas espécies (FISHER, 1936). No *deep learning* a etapa de extração de características não é necessária, é feita apenas a rotulagem (ZHANG *et al.*, 2021). Em contrapartida, as bases de dados *deep learning* costumam ser muito maiores que as bases de dados de aprendizagem de máquina (DE ALMEIDA PEREIRA *et al.*, 2021), pois é preciso realizar um treinamento mais exaustivo para calibrar corretamente o peso que os neurônios atribuem para cada uma das saídas possíveis.

Considerando o problema em que um computador recebe como entrada uma imagem que se sabe que é de um algarismo de 0 a 9 com grafia humana. O objetivo do computador é identificar qual algarismo é esse. Na fase de *treinamento*, pode-se obter uma grande quantidade de imagens de algarismos, escritos por diversas pessoas, e fazer a rotulagem para todas as imagens coletadas. Em outras palavras, está sendo dito ao computador qual saída é esperada para cada entrada (ZHANG *et al.*, 2021; GOODFELLOW; BENGIO; COURVILLE, 2016). A partir disso, passa-se para o modelo as entradas (imagens) e as saídas esperadas para cada entrada (rótulo).

Na sequência, após um treinamento com diversas imagens, espera-se que o modelo seja capaz de identificar o algarismo de novas imagens desconhecidas. Para isso, a fase de *teste* consiste em justamente apresentar ao modelo imagens novas, não rotuladas, e medir seu desempenho ou acurácia, que consiste em avaliar o percentual de imagens desconhecidas que tiveram seus algarismos identificados corretamente.

2.3.1 Deep Q-Learning

O *Deep Q-Learning* (DQL) é um paradigma que junta AR com o *deep learning*. A demanda para essa junção se dá pela existência de ambientes de aprendizado que possuem um espaço de estados grande, sendo assim, métodos tradicionais de AR como o SARSA e o *N-step* SARSA podem se tornar ineficazes, pois é necessário armazenar uma tabela Q muito extensa.

O DQL pode ser visto como uma junção dos métodos citados por haver uma modificação na forma com que se calcula a função Q, pois isso passa a ser feito por meio de uma rede neural, mesmo que o ambiente de aprendizado ainda seja mapeado em um PDM, como no AR.

O funcionamento do método se baseia em interações sucessivas do agente com o ambiente de aprendizado, da mesma forma que métodos de AR baseados em diferença temporal. Para isso, em cada instante de tempo t, o agente realiza uma iteração, obtendo a quádrupla

$$(s_t, a_t, R_t, s_{t+1})$$

em que s_t representa o estado no instante t, a_t representa a ação tomada, R_t representa a recompensa recebida do agente pelo ambiente, e s_{t+1} representa o estado em que o agente termina, após tomada a ação a_t (FAN *et al.*, 2020).

Esta quádrupla é então armazenada na chamada memória de *replay*, da qual são extraídas *minibatches*, i.e., pequenas amostras de iterações, que são utilizadas para treinar a rede neural, e melhorar as aproximações de $Q_t(s_t,a_t)$, ou seja, as aproximações dos valores Q de cada par estado-ação, sem que haja a necessidade de armazenar este valor para cada par, dado que apenas torna-se necessário o armazenamento da rede neural, cujo tamanho varia em função do número de ações do agente, e dos parâmetros fixados inicialmente. O funcionamento básico de um método de DQL é ilustrado na Figura 7.

Este método, bem como os métodos de AR baseados em diferença temporal, pode ser aplicado em SEDs com comportamento flexível, ou seja, que contêm um conjunto de eventos probabilísticos, como os sistemas industriais reais. Estes em específico podem apresentar modelagens com conjuntos de estados muito grandes, fazendo com que se torne computacionalmente intratável a utilização de métodos tabulares.

No entanto, por utilizarem redes neurais artificiais, os métodos de *Deep Q-Learning* têm como vantagem sobre os métodos de AR de diferença temporal, a capacidade de processar justamente tais sistemas com um número elevado de estados, pelo fato de não ser necessário

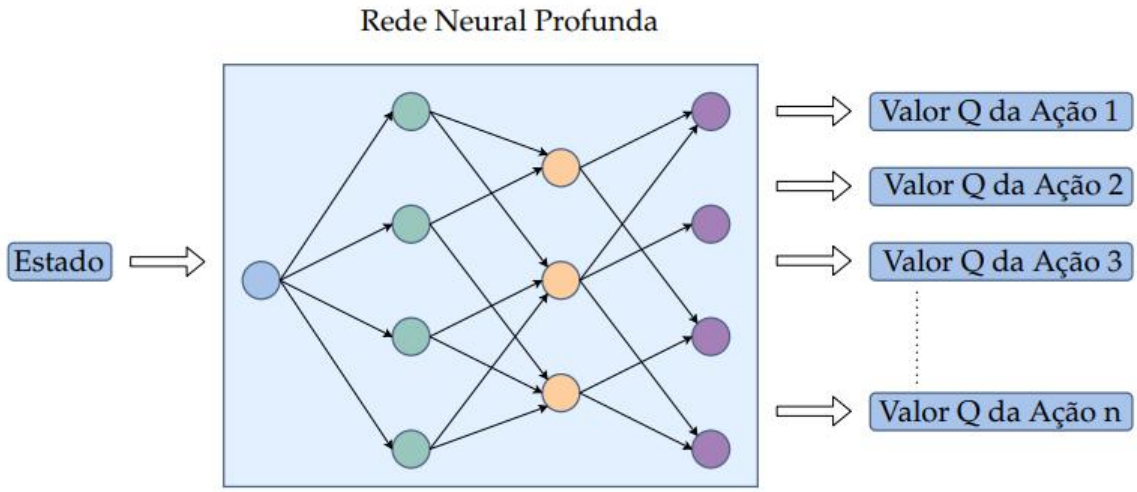


Figura 7 – Ilustração do funcionamento de um método de *Deep Q-Learning*. A entrada da rede, representada em azul, simboliza o estado atual do agente; as camadas centrais, representadas em verde e amarelo, simbolizam as camadas ocultas da rede; e a última camada, representada em roxo, simboliza os valores Q calculados pela rede, para cada uma das ações que o agente pode tomar no estado em que se encontra.

armazenar a tabela Q, e apenas a rede neural.

3 TRABALHOS RELACIONADOS

A Figura 8 apresenta como diferentes abordagens da literatura se aderem aos diferentes perfis de sistemas industriais, que podem ser vistos como SEDs, tratando os eventos probabilísticos com estruturas de redes de Petri (MURATA, 1989b), metaheurísticas (OSAKI; HOSOE; HAGIWARA, 2020), lógica *fuzzy* (WANG *et al.*, 2021) e ainda *Machine Learning*, por exemplo, mostrando também o objetivo deste trabalho, que é tratá-los utilizando o DQL.

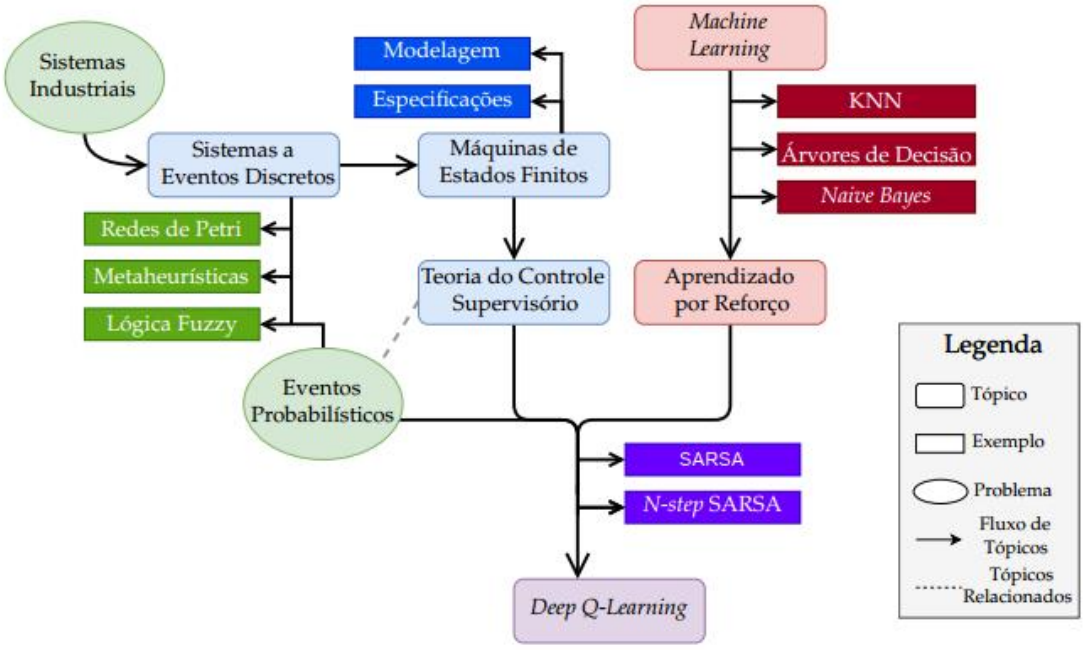


Figura 8 – Diferentes abordagens utilizadas para lidar com sistemas industriais que apresentam eventos probabilísticos.

Inicialmente, assumem-se dois grandes grupos de processos industriais: aqueles que evoluem a eventos (SEDs) (CASSANDRAS; LAFORTUNE, 2010) e os que não. Sistemas que demandam uma coordenação lógica, em certo grau de abstração, como os sistemas de manufatura por exemplo, são classicamente vistos como SEDs, portanto o foco da discussão deste trabalho é neste grupo, apesar de ser possível tratá-los no domínio de tempo contínuo (ROMAN *et al.*, 2019; TURNIP; PANGGABEAN, 2020; PRECUP *et al.*, 2020).

Quando um sistema é visto como SED, conforme citado acima, é possível usar métodos específicos para tratar este tipo de sistema, e os trabalhos aqui listados apresentam algumas maneiras de fazer isso, dependendo do nível de flexibilidade que o sistema agrega. Uma abordagem bastante conhecida é a modelagem formal por métodos como as Redes de Petri (MURATA, 1989b), como em Murata (1989a), tal como sua variante temporizada estocástica, as GSPNs (XUE; KIECKHAFER; CHOOBINEH, 1998). Outra maneira de lidar com SEDs é por meio de

Simheurísticas, às quais as Redes de Petri podem ser agregadas (LATORRE-BIEL *et al.*, 2021), ou então Metaheurísticas (OSAKI; HOSOE; HAGIWARA, 2020). Outra possibilidade existente é o campo relativamente novo da teoria de filas (TIWARI; AL-ASWADI; GAURAV, 2021), ou, de forma mais conservadora, as próprias MEFs (P. J. G. RAMADGE, 1989).

Uma possibilidade para identificar qual a abordagem mais adequada para lidar com SEDs é observando-se e particionando-se o seu conjunto de eventos conforme a natureza de ocorrência: os eventos probabilísticos e os não-probabilísticos. No geral, os eventos probabilísticos podem ser tratados por versões temporizadas das Redes de Petri ou Autômatos, ou expostos como Processos Decisórios de Markov (PDMs) (PUTERMAN, 1994), para serem futuramente processados com Lógica *Fuzzy* (GOMES; MADEIRA; BARBOSA, 2021; WANG *et al.*, 2021; LI; XIONG, 2021), Aprendizado por Reforço (AR) (SUTTON; BARTO, 2018), integração de AR com *Deep Learning* (MNIH *et al.*, 2015), também chamado de *Deep Q-Learning*, método também utilizado em (CHEN; ULMER; THOMAS, 2021), além de outros métodos similares. Diferentemente, eventos não-probabilísticos são preferencialmente modelados como MEFs para facilitar a síntese do supervisor (MOHAJERANI *et al.*, 2021; P. J. G. RAMADGE, 1989), e processo de verificação (MALIK; WARE, 2020).

No presente trabalho, o conjunto de eventos não probabilísticos tem seus eventos modelados utilizando MEFs, e seu supervisor é processado pela sintetização da TCS, que garante o controle ótimo (P. J. G. RAMADGE, 1989) para esse conjunto de eventos, plantas e especificações. Na sequência, o controlador resultante deste processo e o conjunto de eventos probabilísticos são juntados e convertidos em representações em PDMs, que na sequência são processados pelos algoritmos de DQL, variação com redes neurais do AR baseado em diferença temporal. O modelo resultante é capaz de decidir quais eventos e sequências de eventos do sistema proporcionam uma recompensa total maior a ele.

É possível aplicar outros algoritmos de aprendizado de máquina além do DQL (DANKWA; ZHENG, 2019), ou do AR (AHMED *et al.*, 2019), mas isso requer um conjunto de dados preexistente para o treinamento, o que implica na diminuição da flexibilidade que é trazida pela simulação do agente aprendendo através do ambiente. Também é possível lidar com a TCS e o AR separadamente. Por exemplo, a TCS pode ser usada para otimização de produção em sistemas de manufatura, mesmo que parcialmente controláveis (PUTTEN *et al.*, 2020), reconhecimento de contextos sobre modelos em MEFs (LUCAS SILVA; RIBEIRO; TEIXEIRA, 2017), linguagens com preservação de ordem (NOORULDEEN; SCHMIDT, 2020) e sintetização de ataques de *hackers* (MATSUI; LAFORTUNE, 2021). Já no campo do AR, existem estudos

focados na organização de sistemas de produção (WASCHNECK *et al.*, 2018), despacho de pedidos (STRICKER *et al.*, 2018) e otimização de precificação (KRASHENINNIKOVA *et al.*, 2019). O DQL engloba problemas como de alocação de recursos em redes de radiocomunicação (GIANG; THANH; KOO, 2021), otimização de tempo de entrega de produtos com drones (CHEN; ULMER; THOMAS, 2021), dentre outros. Mesmo estas sendo todas áreas muito difundidas, a aplicação de métodos de DQL em ambientes modelados como SEDs, que englobam eventos probabilísticos, não é bastante explorada, apesar de haverem iniciativas, como (RABE *et al.*, 2017).

A separação dos conjuntos de eventos por natureza de ocorrência, e a seleção de métodos específicos para seu processamento, conforme explicado acima, implica que cada estudo retorna benefícios distintos para as práticas de controle e levam a métricas de performance variadas. Por exemplo, a TCS é conhecida por sua eficiência para segurança, mas não por sua flexibilidade, ao passo que o AR e o DQL são apropriados para lidar com ambientes dinâmicos, mas não garantem segurança. Como cada abordagem traz vantagens específicas ao processo de controle e elas se estruturam sobre *frameworks* distintos, a combinação delas não é direta, o que é solucionado neste trabalho, coletando as melhores características de cada uma e suportando o controle seguro e flexível de processos industriais.

4 PROJETO

Este trabalho parte de resultados obtidos em trabalho anterior, que estão disponíveis em Kallil M. C. Zielinski *et al.* (2021), e são reproduzidos nas Seções 4.3 e 4.4. A partir destes esforços iniciais, é desenvolvido um método alternativo ao utilizado em Kallil M.C. Zielinski *et al.* (2021), aplicando *Deep Q-Learning*, cujos resultados são comparados com os disponíveis em Kallil M. C. Zielinski *et al.* (2021), para casos de teste similares aos expostos.

4.1 METODOLOGIA

O trabalho foi desenvolvido conforme os passos da Figura 9.

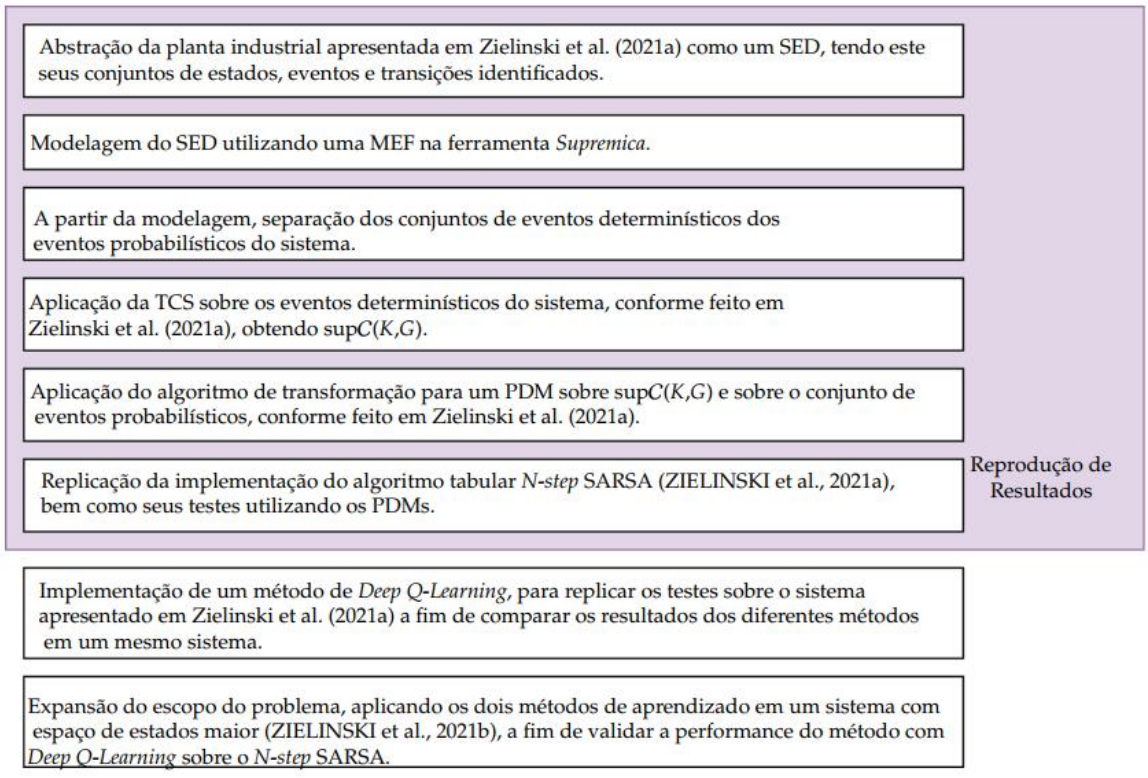


Figura 9 – Atividades desenvolvidas no projeto.

4.2 APRESENTAÇÃO, MODELAGEM E SÍNTESE DE UMA PLANTA INDUSTRIAL

Este trabalho parte do exemplo explorado em Kallil M.C. Zielinski *et al.* (2021) (Figura 10), envolvendo a versão parcial de um sistema de manufatura automotiva. O intuito inicial é de reproduzir aqueles resultados, em seguida aplicar o método de DQL proposto neste trabalho e,

por fim, comparar as duas abordagens. Essa comparação leva a outro resultado deste trabalho, que é estender esse exemplo para mostrar que essa extensão seria inviável de ser processada com a abordagem de Kallil M.C. Zielinski *et al.* (2021).

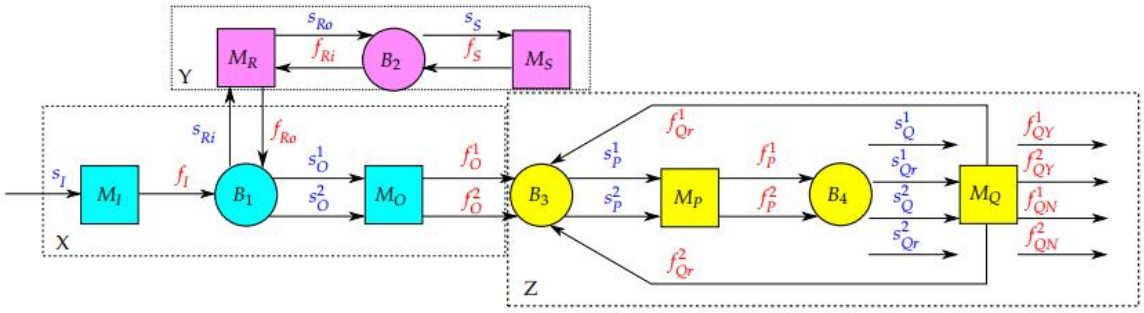


Figura 10 – Versão parcial de um sistema flexível de manufatura automotiva. Quadrados representam máquinas, círculos representam *buffers*, setas representam o fluxo do sistema. Os rótulos em azul representam eventos controláveis, ao passo que os em vermelho representam os não-controláveis.

O exemplo se divide em três blocos: X, Y, e Z, e cada um deles é responsável por uma tarefa específica no processo de produção. Os veículos entram no sistema pelo bloco X, e no *buffer* B_1 é tomada uma decisão: o veículo pode ser escolhido para receber uma customização (e.g., uma pintura diferenciada, ar-condicionado ou dispositivos de segurança), que é aplicada ao veículo no bloco Y, pelo evento s_{Ri}, antes de o veículo ser direcionado ao bloco Z; após a customização, o veículo passa a ser um veículo de *tipo 1*. A opção alternativa é quando o veículo é dirigido do bloco X diretamente ao bloco Z sem receber nenhuma customização, sendo estes os veículos de *tipo 2*. Isso leva à primeira questão que busca-se responder.

Questão 1 *Entrar ou não no bloco Y é uma decisão que não depende do processo de controle, mas sim no que é melhor para a produção e pelo lucro esperado. O controlador deve então permitir ambos os caminhos, portanto, como o controlador pode providenciar tal decisão?*

Os dois tipos de veículos podem então entrar no bloco Z, onde o restante da manufatura é feito. Quando é atingida a estação B_4, pode-se obter duas saídas: (i) o veículo deixa o sistema, independentemente de ter sido ou não aprovado no teste de qualidade (evento f_{QY}^i (SIM) e f_{QN}^i (NÃO) para os tipos $i = 1,2$); ou (ii) o carro é retrabalhado no bloco Z em caso de rejeição no teste de qualidade (eventos f_{Qr}^i, para os tipo $i = 1,2$). Isso leva à segunda questão a ser respondida.

Questão 2 *Quando um veículo falha no teste de qualidade (máquina M_Q), o sistema de controle deve optar entre: (i) um retrabalho, tentando complementar sua customização e esperar aumentar suas chances de aprovação no próximo teste de qualidade; ou (ii) removê-lo da linha de produção.*

Esta é também uma decisão que não depende do processo de controle, mas expectativas de lucro da empresa. Do controlador se espera que apenas permita ambas as possibilidades para que um agente externo tome a decisão. Como esta decisão pode ser fornecida pelo controlador?

As questões 1 e 2 alinham-se à solução inteligente proposta neste trabalho, integrada à solução lógica resultante da TCS. Considerando as questões apontadas, a problemática de controle sobre o sistema, é apresentada na subseção seguinte.

4.2.1 Problemática de controle

No exemplo da Figura 10, considera-se que as múltiplas máquinas executam operações distintas, sobre veículos distintos passando pela linha de produção. Diferentes modelos de veículos são montados utilizando a mesma planta, mas cada veículo requer componentes intrínsecos àquele modelo e, portanto, suas montagens são distintas, o que implica em fluxos de montagem diferentes. Por isso, sistemas de controle como estes são, em geral, associados com tarefas de engenharia complexas, que envolvem tomada de decisões que muitas vezes extrapolam a ordem lógica.

Ainda que o *framework* da TCS ofereça opções que simplificam a modelagem lógica, a síntese, e a programação por meio da geração automática de código, ela não oferece opções de tomada de decisões inteligentes. De fato, o caráter maximamente permissivo de um controlador obtido via TCS remete meramente ao fato de que nenhum evento é proibido sem que isso seja estritamente necessário. Isso difere, por exemplo, da perspectiva de selecionar eventos dentre aqueles habilitados pelo controlador. Na prática, isso significa que a TCS prima pela segurança das ações a serem executadas pelo sistema, sem competir diretamente por produtividade dessas ações. Esta limitação pode ser contornada por meio da aplicação de AR para otimizar as tomadas de decisão, complementando assim a lógica de controle com percepção artificial.

Para isso, é razoável assumir que a unidade de teste (máquina M_Q) possui uma probabilidade associada com o quão esperado é que cada veículo passe no teste. Veículos do tipo 2 são mais prováveis de serem aprovados, pois incluem menos equipamentos a serem testados em comparação com os de tipo 1. A inclusão de mais dispositivos nos veículos de tipo 1 é, portanto, a decisão que mais impacta no valor adicionado ao veículo, em troca de ele ter mais chances de falhar no teste de qualidade. A abordagem aqui utilizada trata dessas decisões de dois parâmetros: (i) a probabilidade de aprovação depois do teste de qualidade, que é associada com a

sua transição na matriz de transições; e (ii) a estrutura de recompensas R que reflete o lucro na manufatura de cada veículo.

A seguir, apresentam-se as etapas de modelagem e de síntese para o exemplo da Figura 10 para, em seguida, incorporar a abordagem inteligente.

4.2.2 Modelagem e Síntese

Cada componente da Figura 10 é modelado como uma MEF na Figura 11.

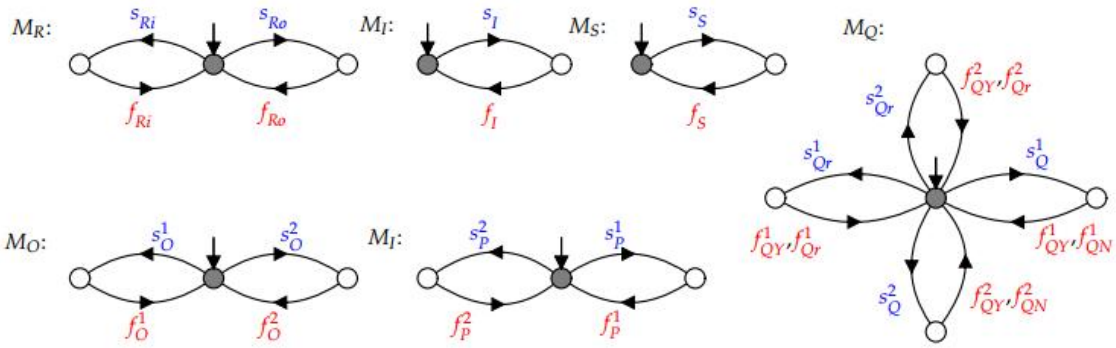


Figura 11 – Planta do sistema de manufatura automotivo.

Os modelos M_I e M_S representam as máquinas de mesmo nome com 2 estados cada, que envolvem transições de início e fim de operação. Os modelos M_O, M_P e M_R representam as máquinas que trabalham em dois modos, um para cada tipo de veículo (1 e 2). Finalmente, MEF M_Q expressa se os veículos de tipo 1 e 2 vão deixar o sistema ou retornar à linha de produção para serem retrabalhados. A composição $G = M_I \parallel M_O \parallel M_S \parallel M_P \parallel M_Q$, com 540 estados, é feita na ferramenta *Supremica*, e corresponde ao comportamento da planta sem restrições. As restrições impostas são representadas pelo conjunto de especificações detalhado na Tabela 1.

Especificação	Descrição
B_i, $i = 1,\cdots,4$	Evitar *overflow* e *underflow* no *buffer* B_i
C_j, $j = O,P,Q$	Controlar os tipos de veículos com que a máquina M_j lida
O_{Rewk}	Limitar o número de ciclos de retrabalho a serem permitidos para cada veículo
O_{Cust}	Decidir sobre a adição ou não de uma customização em cada veículo

Tabela 1 – Especificações a serem consideradas no exemplo.

É possível notar que as especificações B_i e C_j são diretas e não envolvem nenhum tipo de rastreamento de memória, apenas desabilitando alguns eventos para impor o fluxo correto dos veículos na linha de produção. No entanto, a literatura mostra (CURY, J. E. *et al.*, 2015) que a modelagem de O_{Rewk} e O_{Cust} como MEFs pode ser mais complicada, e pode requerer a construção manual de um amplo e complexo modelo de estados, devido aos seguintes motivos:

(i) Ambas têm um fator dinâmico de custo associado, que requer uma abordagem mais flexível de controle que as MEFs.

(ii) A modelagem de O_{Rewk} como MEFs fixa um limite determinístico de ciclos de retrabalho permitidos. Qualquer seja esse limite, ele implica que seja modelada toda a memória que precede a ação de controle, o que aumenta exponencialmente com o número de ciclos e não é prático para casos reais (TEIXEIRA; CURY, J.; QUEIROZ, 2018; TEIXEIRA; HERING DE QUEIROZ; CURY, J., 2015; CURY, J. E. *et al.*, 2015).

(iii) MEFs impõem uma definição estática a O_{Rewk} e O_{Cust}, que não é compatível com a especificação do problema abordado neste trabalho.

Por hora, as possibilidades de um veículo adentrar ao bloco Y, bem como de ser reciclado no bloco Z (especificações O_{Rewk} e O_{Cust}), são mantidas livres pela síntese do controle supervisório. Essas decisões serão tomadas pela camada de *software* baseada em AR, e serão providas automaticamente ao controlador. As MEFs que modelam B_i e C_j são representadas na Figura 12.

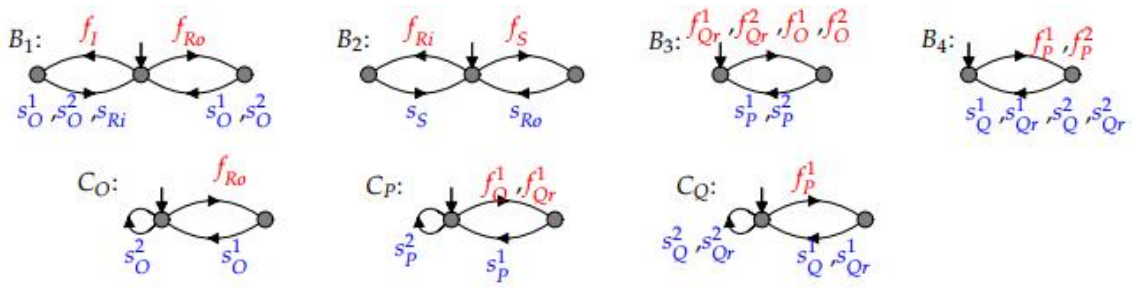


Figura 12 – Restrições de *overflow* e *underflow*.

Em B_i, o *overflow* é controlado desabilitando eventos de chegada de veículos no *buffer* quando este está cheio; por outro lado, o *underflow* é controlado desabilitando os eventos de início de máquinas no estado inicial, ou seja, quando o *buffer* está vazio. Em adição ao controle do *buffer*, é necessário também manter a memória do tipo a ser processado, para que a atuação possa ser compatível. Isto é reforçado pelas especificações C_j. Por exemplo, C_O desabilita a operação da máquina M_O para veículos de tipo 1 (evento s_O^1) no estado inicial, e apenas o habilita se o caminho pelo bloco Y, que culmina no evento f_{Ro}, é escolhido. C_Q trabalha da mesma maneira e o resultado é o rastreamento completo para cada tipo de veículo através do sistema.

A composição $E = B_1 \parallel B_2 \parallel B_3 \parallel B_4 \parallel C_O \parallel C_P \parallel C_Q$ tem 81 estados. Quando integrado à planta, forma o comportamento esperado pelo sistema de manufatura controlado, modelado por

$K = G \parallel E$ com 18630 estados. A síntese envolve o processamento do supervisor $\sup C(K,G)$, este com 1309 estados e 4878 transições, também obtido utilizando a ferramenta *Supremica*.

A partir disso, é feita a tradução do modelo $\sup C(K,G)$ modelado por uma MEF em um PDM, conforme o algoritmo descrito de Kallil M.C. Zielinski *et al.* (2021).

4.3 TRADUÇÃO DE UMA MEF EM UM PDM

Para uma MEF $\langle \Sigma, Q, q^{\circ}, Q^{\omega}, \rightarrow \rangle$ ser transformada em um PDM $\langle S, A, T, R, \gamma \rangle$, inicia-se considerando as equivalências entre os conjuntos de estados $S \equiv Q$, o conjunto de eventos e ações $A \equiv \Sigma$, e as relações de transições determinísticas $T \equiv \rightarrow$. É necessário também adicionar a função de recompensa R e o fator de desconto γ para completar o processo de conversão. Tudo isso é feito a nível de código.

R é criada alocando-se recompensas para cada $a \in A$, ou seja, para cada evento do SED da Figura 10 (que coincidem com as ações do PDM após o processo de tradução), são alocadas recompensas para mapear o benefício obtido com cada ação. Por exemplo, o evento f^1_{QY} retorna uma recompensa *positiva* alta, ao passo que o evento f^1_{QN} retorna uma recompensa *negativa* alta. Ações básicas das máquinas também devem ter uma pequena recompensa negativa associada a elas, referente a consumo de energia, manutenção, etc. Por fim, carros do tipo 1, com customização, devem trazer mais lucro ao sistema.

Para criar o ambiente de aprendizado por reforço, deve-se, finalmente, adicionar probabilidades às transações do PDM determinístico obtido até então. Assume-se que tais probabilidades estejam bem definidas do ponto de vista da engenharia. Para o exemplo da Figura 10, considera-se apenas que há probabilidades de aprovação e rejeição dos veículos após o teste de qualidade, com as demais transações sendo determinísticas. Também assume-se que veículos de tipo 2 (não customizados) têm igual ou maior chance de aprovação que os veículos de tipo 1, porque passam por menos máquinas para serem produzidos.

As instâncias de teste utilizadas para testar o ambiente de AR possuíam todas 9 casos de de teste cada. Para uma dada instância, altera-se algum dos seus parâmetros 9 vezes: uma para cada caso de teste na instância, de forma crescente ou decrescente, a fim de observar a resposta do comportamento do agente mediante esta variação.

Exemplificando, na primeira instância, uma das ações probabilísticas do agente teve sua probabilidade de ocorrência alterada entre 10% e 90%. A primeira das instâncias de teste é conforme definida abaixo.

Instância de Teste 1 : *A ação de* ***aprovação de veículos do tipo 1*** *(com customização) é iniciada em 90% no primeiro caso de teste, e é reduzida em 10% em cada caso subsequente, até atingir 10% no caso 9; a probabilidade de rejeição deste mesmo tipo de veículo é complementar à de aprovação, e cresce de 10% a 90%; a probabilidade de retrabalho do veículo está atrelada à probabilidade de rejeição; as demais probabilidades permanecem fixas em todos os casos. É importante notar as probabilidades fixas de aprovação e rejeição de veículos do tipo 2, que são respectivamente 90% e 10%, bem como a recompensa fixa de -10 para as ações que não possuem probabilidade atrelada a elas. A Tabela 2 apresenta todas as probabilidades das ações do agente.*

Ação:	s_I, $s_O^{1,2}$, $s_P^{1,2}$, $s_Q^{1,2}$, $s_{Qr}^{1,2}$, s_{Ri}, s_{Ro}, s_S, f_I, $f_O^{1,2}$, $f_P^{1,2}$, f_{Ri}, f_{Ro}, f_S		
Recomp.	-10		
Prob. (%)	Não considerada		
Ação:	**Recomp.**	**Prob. (%)**	**Descrição:**
f_{QN}^1	-30000	10 a 90	Rejeição de veículo do tipo 1
f_{QN}^2	-20000	10	Rejeição de veículo do tipo 2
f_{QY}^1	30000	90 a 10	Aprovação de veículo de tipo 1
f_{QY}^2	20000	90	Aprovação de veículo de tipo 2
f_{Qr}^1	-1000	10 a 90	Retrabalho de veículo de tipo 1
f_{Qr}^2	-1000	10	Retrabalho de veículo de tipo 2

Tabela 2 – Recompensas e probabilidades assumidas para o exemplo.

Após a conversão da MEF em um PDM, este é utilizado como entrada para os algoritmos de AR. Neste trabalho, especificamente, foi feita inicialmente a reprodução dos resultados previamente obtidos por Kallil M.C. Zielinski *et al.* (2021), usando o algoritmo *N-step* SARSA para lidar com as ações probabilísticas e, na sequência, foi implementada a solução de proposta.

Após a modelagem e conversão da MEF que representa a planta industrial em um PDM, e a partir da implementação dos algoritmos de AR, foi feito um comparativo das recompensas médias obtidas utilizando o algoritmo inteligente, e sem utilizá-lo, para fins de comparação de desempenho; também foi gerado um gráfico ilustrando as ações do sistema que foram priorizadas em cada casa.

4.4 *N-step* SARSA

Primeiramente, com o intuito de reproduzir os resultados apresentados em Kallil M.C. Zielinski *et al.* (2021), a implementação do algoritmo *N-step* SARSA foi replicada, tendo essa se

mostrado suficiente para lidar com os eventos probabilísticos do sistema, sendo capaz de otimizar as recompensas obtidas pelo agente de aprendizado, quando compara-se tais recompensas ao sistema controlado apenas utilizando-se a TCS.

Na Figura 13 são apresentadas as recompensas obtidas pelo agente utilizando o algoritmo *N-step* SARSA e sem o utilizar, ou seja, apenas com a TCS. É válido reiterar que a TCS sem nenhum tipo de otimização inteligente não almeja agregar resultado no tratamento dos eventos probabilísticos, por deixar aleatória a escolha do agente entre tomar ou não a ação que pode representar algum risco. Logo, a linha do gráfico em que se representa a TCS na Figura 13, o controlador deixou a escolha sobre os eventos probabilísticos totalmente aleatória.

Para esta instância de teste, a entrada para o agente foi o PDM obtido na Seção 4.3, o que significa que no eixo horizontal do gráfico está representada a probabilidade de rejeição dos veículos de tipo 1, que cresce de 10% a 90%. Pode-se observar que conforme esta probabilidade aumenta, a recompensa do agente diminui, o que se deve ao fato de que aprovar o veículo do tipo 1 dá ao agente uma recompensa alta, e ao passo que aumenta-se a probabilidade de rejeição do veículo deste tipo, a probabilidade de aprovação diminui, o que faz com que o agente priorize ações que trazem recompensas menores.

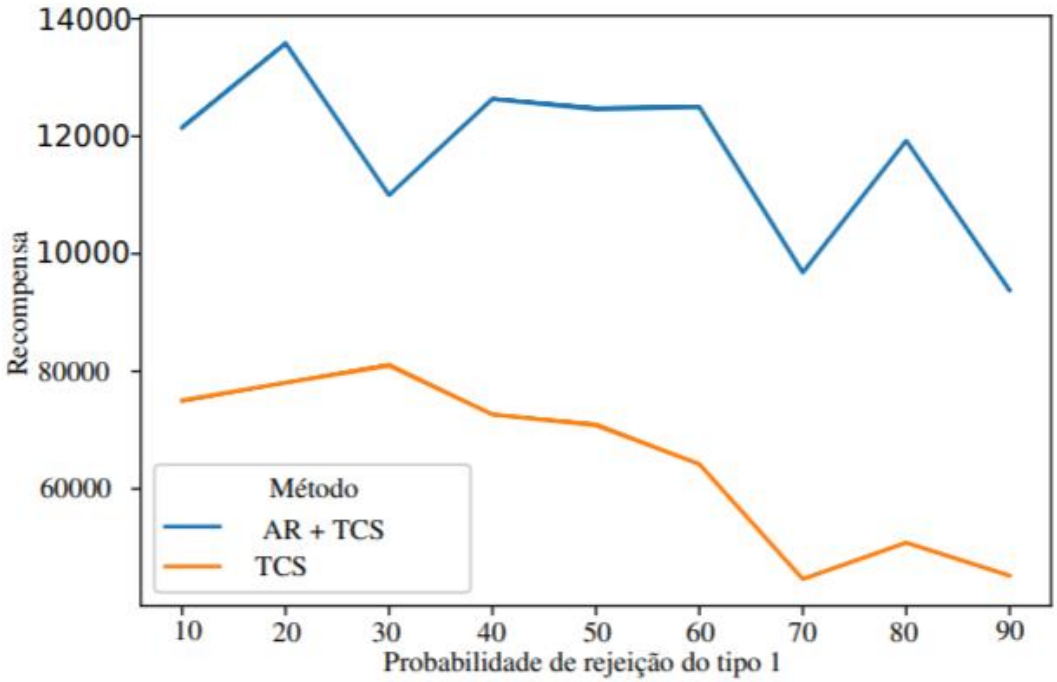


Figura 13 – Variação das recompensas médias com a variação da probabilidade de rejeição de veículo de tipo 1

Na Figura 14 estão representadas as ações que o agente tomou com mais frequência em cada caso de teste sem o uso do algoritmo *N-step* SARSA, ou seja, apenas com o controle

supervisório. Novamente, o eixo horizontal do gráfico representa a probabilidade de rejeição dos veículos de tipo 1.

É possível perceber que a ocorrência de aprovação de veículos do tipo 2 permaneceu praticamente a mesma durante todos os casos, dado que sua probabilidade foi constante; ao mesmo tempo houve um aumento no número de rejeições dos veículos de tipo 1, o que reforça a diminuição na recompensa obtida pelo agente apenas controlado por TCS apresentadas na Figura 13, porque a rejeição do veículo de tipo 1 retorna uma recompensa negativa para o agente.

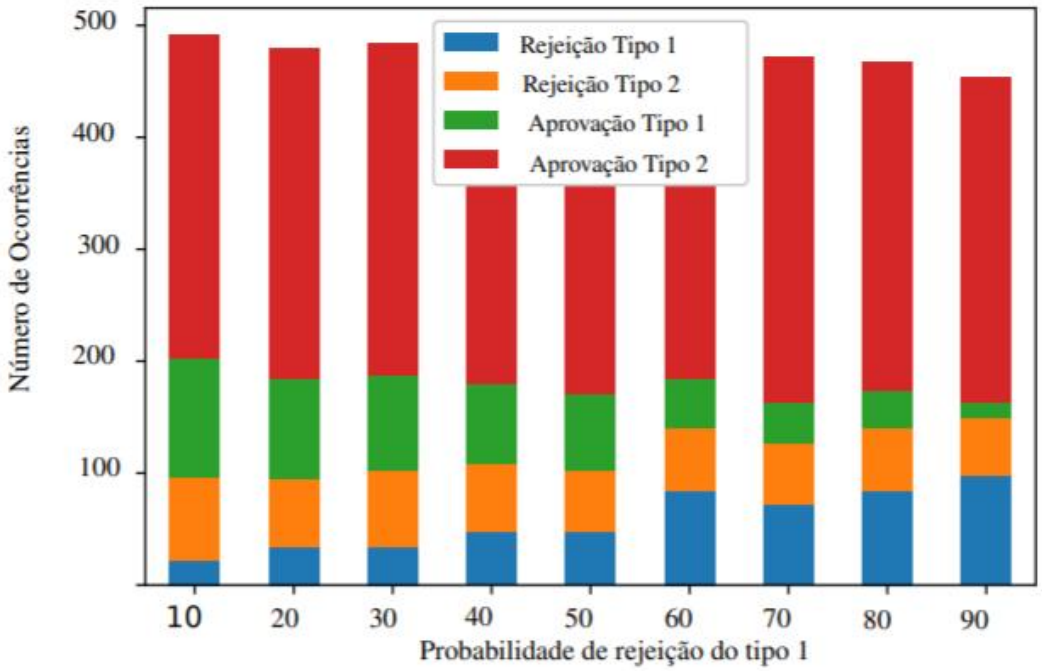


Figura 14 – Ações mais tomadas *sem* o uso do algoritmo *N-step* SARSA.

Já na Figura 15, da mesma forma, são apresentadas as ações tomadas pelo agente utilizando o algoritmo inteligente *N-step* SARSA. Neste caso, pode-se perceber que desde o primeiro caso de teste, as ações de aprovação de veículos de ambos os tipos foram as que mais ocorreram, sendo que nos piores casos, o agente agiu mais vezes a fim de produzir veículos de tipo 1 que para tentar produzir veículos de tipo 2 e acabar por rejeitá-los devido à alta probabilidade desse evento.

Esta característica de buscar maximizar as recompensas positivas e minimizar as recompensas negativas, o que ocorreu com o uso do algoritmo *N-step* SARSA, explica por que o desempenho do agente foi melhor fazendo o uso do método do que sem usá-lo.

Já na Figura 16 são mostrados o número de vezes que o agente optou por ações de retrabalho dos veículos com e sem o método *N-step* SARSA. Pode-se perceber que conforme a probabilidade de rejeição do veículo customizado aumenta, o agente inteligente opta por

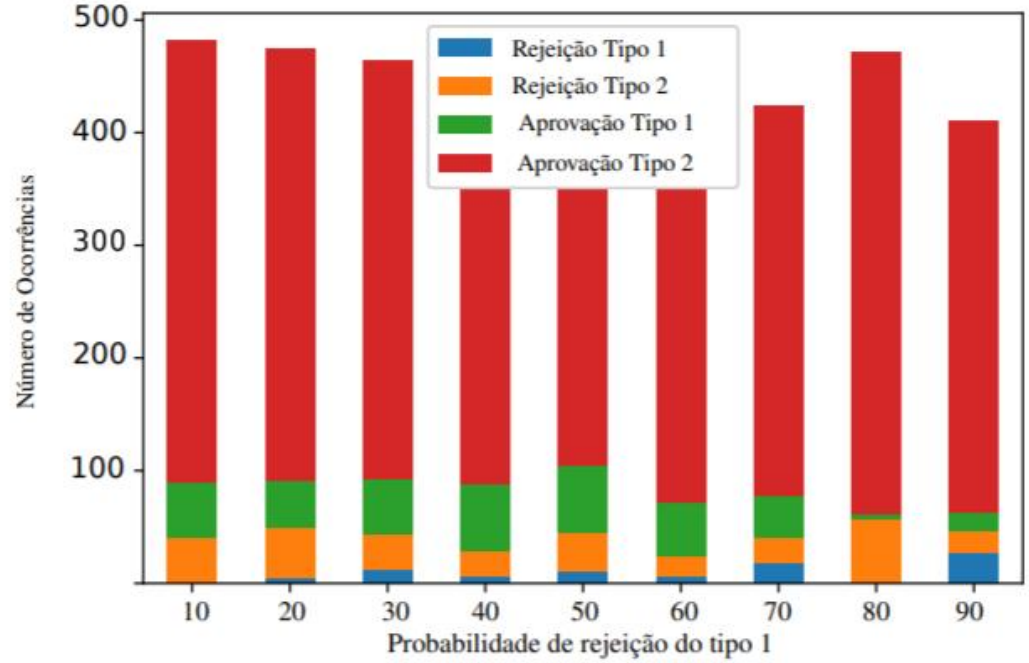


Figura 15 – Ações que o agente mais tomou *com* o uso do algoritmo *N-step* SARSA.

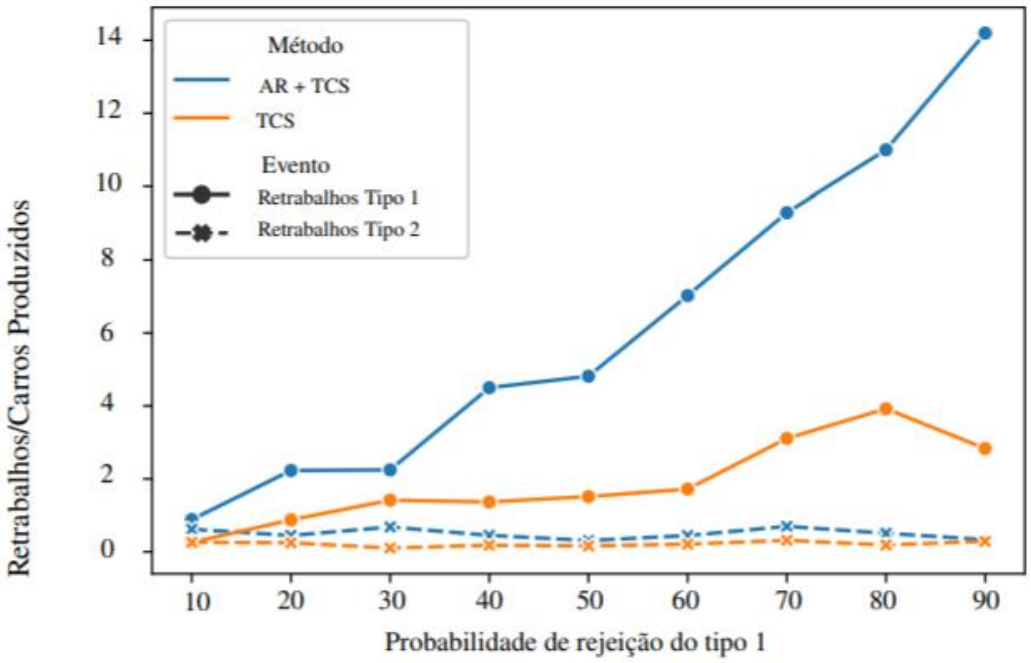


Figura 16 – Razão entre o número de ciclos de retrabalho e o número de carros produzidos para o agente com e sem o método *N-step* SARSA.

retrabalhar mais vezes os veículos sem customização, já que a probabilidade de aprovação destes segue inalterada, e assim, ele maximiza sua recompensa.

4.5 MÉTODO DE *DEEP Q-LEARNING*

Após essa reprodução de resultados utilizando o método *N-step* SARSA, que foi descrita na seção anterior, foi implementado o método de *Deep Q-Learning* buscando comparar o desempenho dos agentes tendo seus comportamentos controlados pelos dois diferentes métodos, o que é abordado na presente seção.

Primeiramente foi implementada a rede neural, em que são definidos os parâmetros de taxa de aprendizado, número de dimensões de entrada da rede, o número de nós das camadas ocultas e o número de ações do agente; junto a isso, é definido o método *forward* da rede, responsável por propagar os valores de entrada até as camadas de saída, calculando o erro para treinar a rede. Todas estas etapas são implementadas utilizando módulos de AR das bibliotecas *torch* do *Python*.

A classe do agente engloba também as implementações dos métodos de armazenar a transição do agente, escolher transição e aprender, que são responsáveis pelas etapas de aprendizado.

A Figura 17 apresenta as recompensas médias do agente fazendo uso dos dois métodos inteligentes – *N-step* SARSA e *Deep Q-Learning*– comparados ao uso da TCS sem o suporte inteligente. É justo ressaltar que a TCS não é um método de otimização e que, portanto, não busca melhorar o comportamento do agente. Logo, mesmo o método *N-step* SARSA, que é tabular, já apresenta resultados melhores nas recompensas médias obtidas pelo agente devido a esse já ser capaz de aprender quais comportamentos lhe dão recompensas mais altas. Porém, em comparação com o DQL, esse obteve resultados ainda superiores ao *N-step* SARSA, em todos os casos de teste, o que quantifica a superioridade do método de aprendizagem utilizando redes neurais.

Já na Figura 18 estão expostas as ações de aprovação e rejeição que o agente de DQL mais tomou. Em comparação com as Figuras 14 e 15, pode-se perceber que o agente foi muito mais incisivo na escolha das ações, dado que em todos os casos de teste aprovou-se muito mais veículos de tipo 2 que de tipo 1, ou seja, optou-se num geral por produzir veículos sem customização, pois a chance de obter uma recompensa alta foram maiores. No entanto, pode-se perceber que para os casos de teste em que as probabilidades de aprovação dos veículos customizados foi mais alta (mais à esquerda na figura), houve uma tentativa de priorizar sua aprovação, mas ao passo que a probabilidade de rejeição dos veículos customizados aumentou (mais à direita na figura), esta ação foi minimizada. Em um contexto geral, isso indica que o agente optou majoritariamente

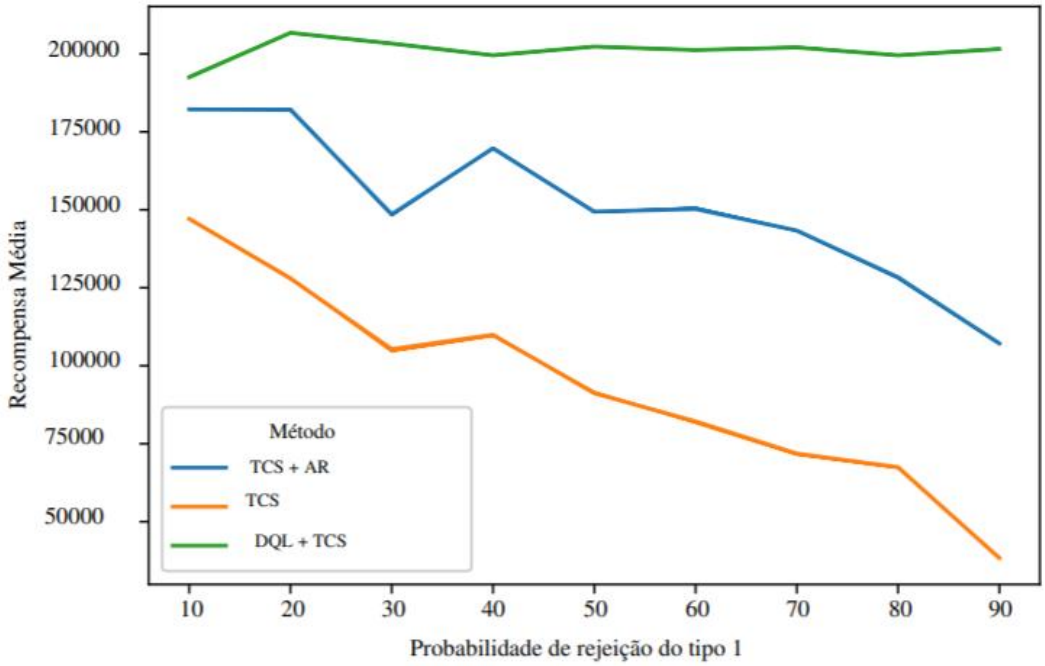


Figura 17 – Variação da recompensa média com a variação da probabilidade de rejeição de veículo de tipo 1.

por produzir o veículo que não teve sua probabilidade de aprovação alterada.

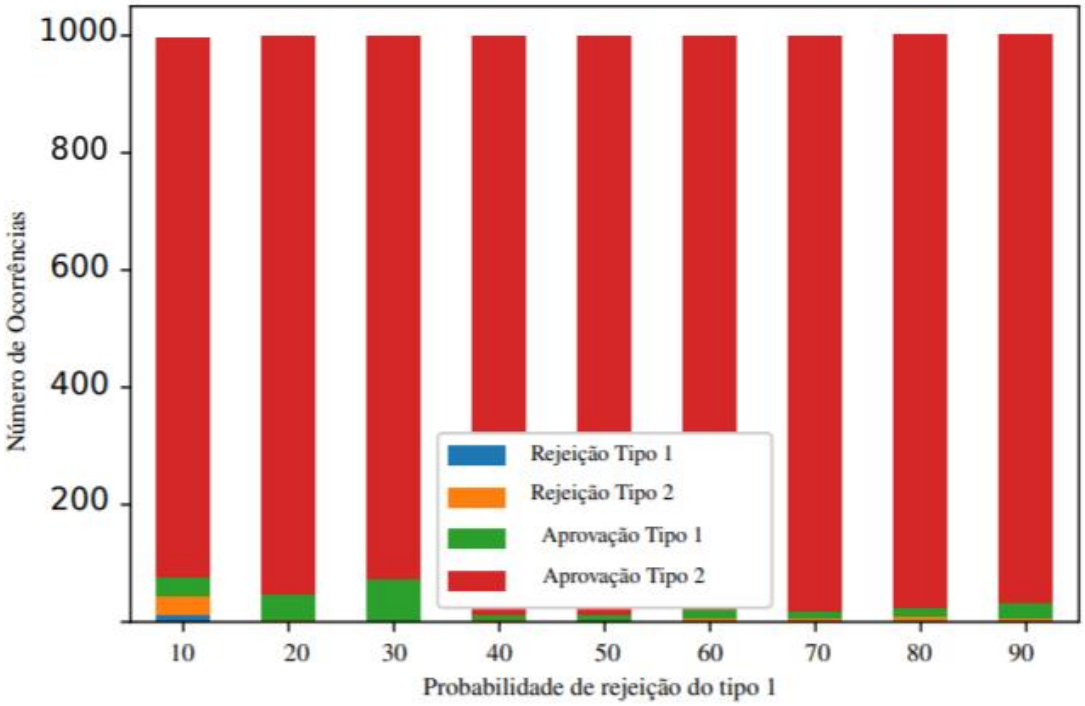


Figura 18 – Ações que o agente mais tomou *com* o uso do algoritmo de *Deep Q-Learning*.

Por fim, ainda tratando-se da primeira instância de teste, a Figura 19 representa o número de retrabalhos que o agente de DQL fez, em comparação aos outros métodos.

É possível perceber que o agente de DQL realizou um número muito baixo de retrabalhos em comparação com o agente implementado pelo *N-step* SARSA. A interpretação mais provável

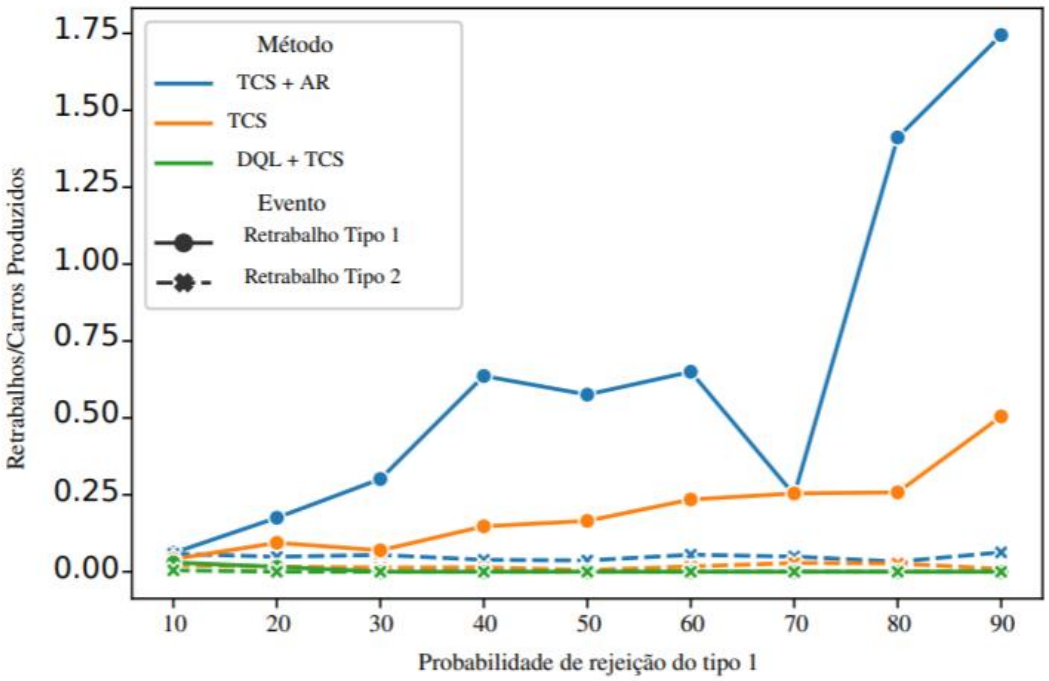


Figura 19 – Razão entre o número de ciclos de retrabalho e o número de veículos produzidos do agente com o método *N-step* SARSA, com o método de DQL e apenas com o controle supervisório.

de tal situação é devido ao fato de que o DQL optou por produzir muito mais veículos de tipo 2, ou seja, sem customização, que tiveram um percentual de rejeição menor que o veículo customizado, sendo assim, retrabalhando menos os veículos, e obtendo as recompensas negativas atreladas ao retrabalho menos vezes.

Na sequência, foi criada uma segunda instância de teste para os algoritmos.

Instância de Teste 2 : *A* ***recompensa*** *de* ***reprovação de veículos do tipo 1*** *(com customização) é iniciada em -20000 no primeiro caso de teste, e é diminuída em 2500 em cada caso subsequente, até atingir -40000 no caso 9, ao passo que a recompensa de aprovação deste tipo de veículo é fixa em 30000; para o veículo de tipo 2, suas recompensas de aprovação e rejeição são fixas em 20000 e -20000, respectivamente. As probabilidades de rejeição e de aprovação do veículo de tipo 1 são fixas em 30% e 70%, respectivamente, e para o veículo de tipo 2, são de 10% e 90%, respectivamente. A Tabela 3 apresenta todas as demais probabilidades e recompensas das ações do agente.*

Para estes casos de teste, os resultados são apresentados nas Figuras 20 a 24. A primeira delas, Figura 20, apresenta as recompensas médias obtidas utilizando ambos os métodos inteligentes *N-step* SARSA e DQL, e sem utilizá-los, ou seja, apenas com a TCS. Novamente, é possível perceber, como esperado, a superioridade dos métodos inteligentes nas tomadas de decisões, em comparação à aleatoriedade da TCS, sendo o DQL o que apresentou o melhor

Ação:	$s_I, s_O^{1,2}, s_P^{1,2}, s_Q^{1,2}, s_{Qr}^{1,2}, s_{Ri}, s_{Ro}, s_S, f_I, f_O^{1,2}, f_P^{1,2}, f_{Ri}, f_{Ro}, f_S$		
Recomp.	-10		
Prob. (%)	Não considerada		
Ação:	**Recomp.**	**Prob. (%)**	**Descrição:**
f_{QN}^1	-20000 a -40000	30	Rejeição de veículo do tipo 1
f_{QN}^2	-20000	10	Rejeição de veículo do tipo 2
f_{QY}^1	30000	70	Aprovação de veículo de tipo 1
f_{QY}^2	20000	90	Aprovação de veículo de tipo 2
f_{Qr}^1	-1000	10	Retrabalho de veículo de tipo 1
f_{Qr}^2	-1000	10	Retrabalho de veículo de tipo 2

Tabela 3 – Representação das recompensas e probabilidades da Instância de Teste 4.5.

resultado em termos de recompensa geral obtida pelo agente, para essa instância de teste.

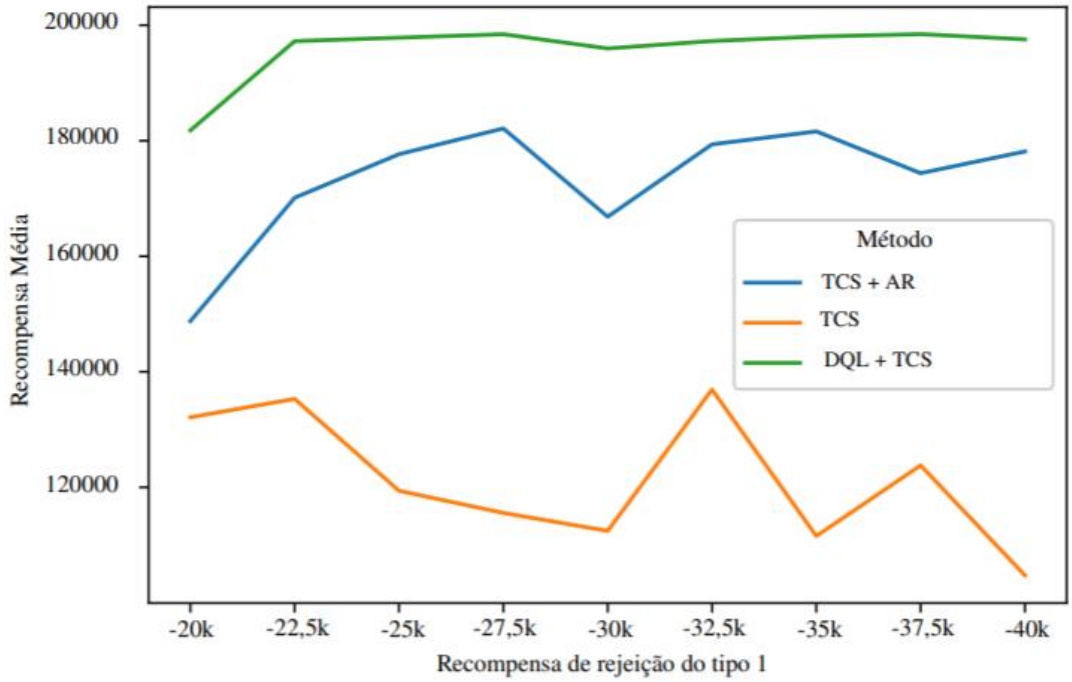


Figura 20 – Recompensas variando a recompensa de rejeição de veículo do tipo 1.

Já as Figuras 21, 22 e 23, apresentam as ações priorizadas pelos agentes implementados com o método de DQL, com o algoritmo SARSA e utilizando apenas a TCS, respectivamente. Primeiramente, na Figura 21, é possível perceber novamente que o agente de DQL optou por aprovar o veículo sem customização, que por sua vez não teve sua recompensa negativa associada à rejeição alterada, e sim permaneceu constante em -20000.

Na Figura 22, por sua vez, o agente utilizando o *N-step* SARSA também priorizou aprovar veículos de tipo 2, apesar de ter tentado explorar com mais constância a aprovação dos veículos customizados, mas a recompensa negativa menor dos casos de rejeição de veículos de tipo 1 acabou pesando mais nas decisões do agente.

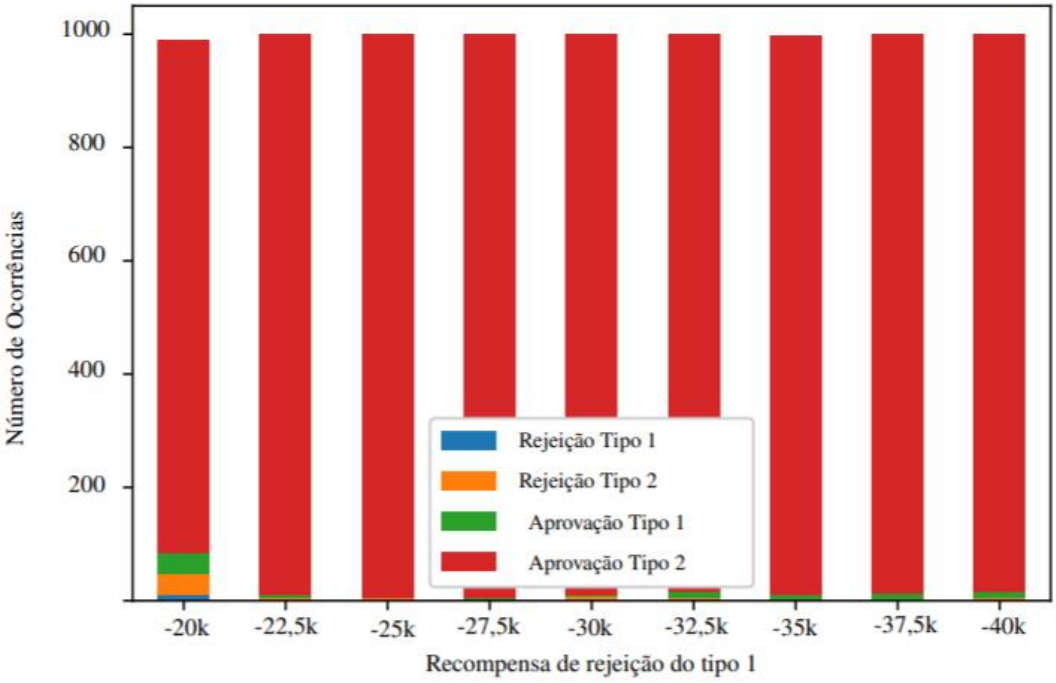


Figura 21 – Ações tomadas com o DQL **variando a recompensa de rejeição de veículo do tipo 1.**

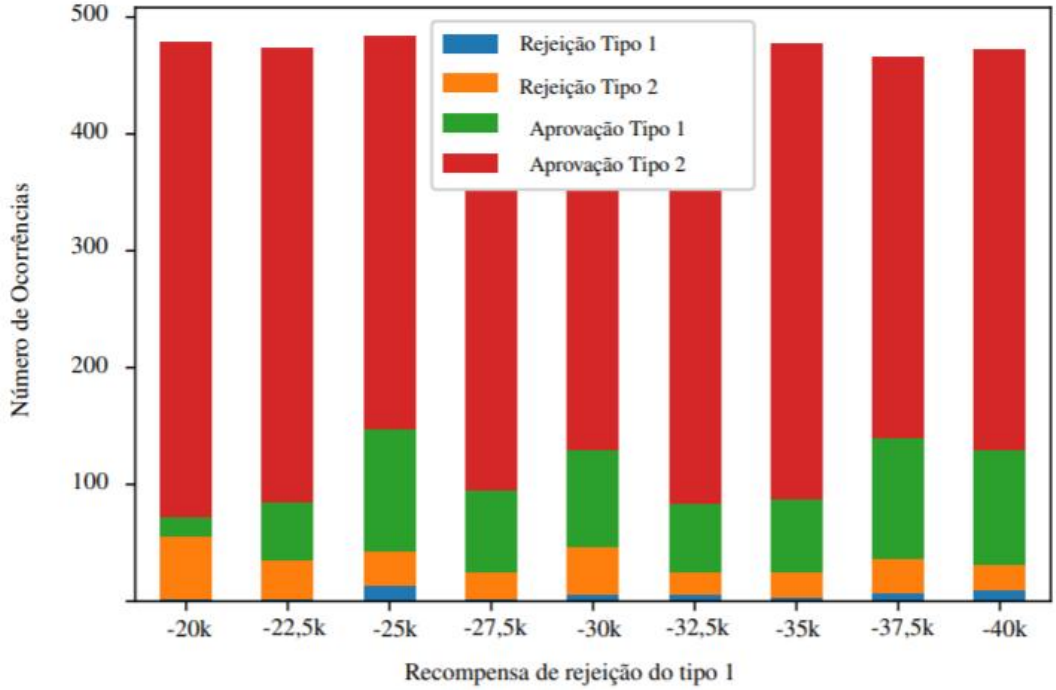


Figura 22 – Ações tomadas com o *N-step* SARSA variando a recompensa de rejeição de veículo do tipo 1.

Já na Figura 23, pode-se perceber que utilizando apenas a TCS sem nenhuma inteligência implementada, o agente manteve-se constante na escolha de ações, sem apresentar nenhuma variação conforme mudou-se os valores de recompensas na instância de teste.

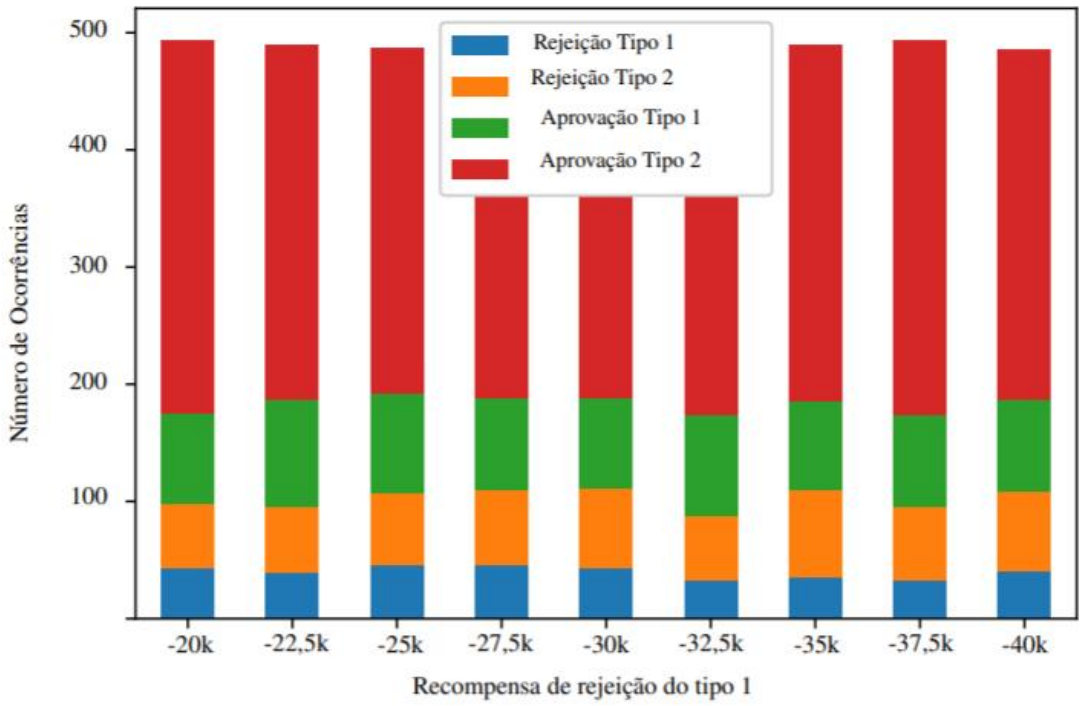


Figura 23 – Ações tomadas com a TCS **variando a recompensa de rejeição de veículo do tipo 1.**

Finalmente, a Figura 24 apresenta a proporção de retrabalhos feita por cada abordagem; novamente pode-se perceber que o agente de DQL optou por minimizar as ações de retrabalho, ao passo que o método *N-step* SARSA e a TCS pura ainda buscaram retrabalhar veículos com mais frequência.

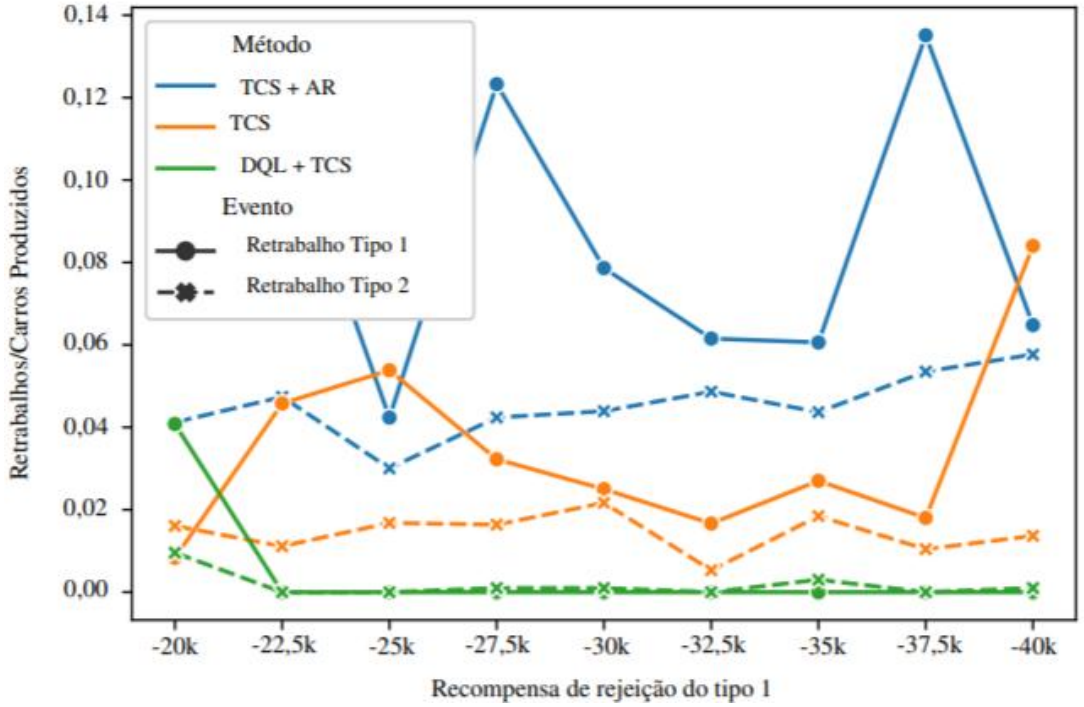


Figura 24 – Retrabalhos variando a recompensa de rejeição de veículo do tipo 1.

4.6 EXPANSÃO DO PROBLEMA

Na seção anterior, foi possível perceber que o *Deep Q-Learning* apresentou resultados superiores aos do método *N-step* SARSA para os casos de teste tratados. No entanto, a eficiência agregada por métodos que utilizam redes neurais não se resume a apenas melhorar as recompensas para casos de teste tratáveis. Adicionalmente, esses métodos também possuem reflexos no tratamento de problemas de grande porte, cujos espaços de estados são significativamente maiores. Para estes casos, o processamento computacional por meio de métodos tabulares, tais como o *N-step* SARSA, pode se tornar custoso ou, em alguns casos, intratável.

Por isso, a fim de validar a eficácia do método de DQL para instâncias de teste com espaços de estados grandes, foi incluída uma plataforma giratória no final da planta do exemplo apresentado na Figura 10, resultando no sistema ilustrado na Figura 25.

O objetivo desta plataforma é acomodar os veículos que saem da linha de montagem e proceder com as etapas de teste, a fim de aprová-los ou descartá-los. Os veículos entram na plataforma na posição P1, e após a rotação da plataforma, quando se encontram em P2, a máquina MD é responsável por descartar os veículos que se sabe serem defeituosos. Na sequência, após outra rotação, na posição P3, a máquina MT testa os veículos para verificar se podem ser aprovados, e após uma última rotação, a máquina MA, por fim, retira os veículos aprovados da

linha de produção na posição P4.

Embora essa pequena extensão do processo seja uma etapa usual na rotina fabril, ela causa um enorme efeito sobre o espaço de estados resultante da nova versão do modelo do sistema.

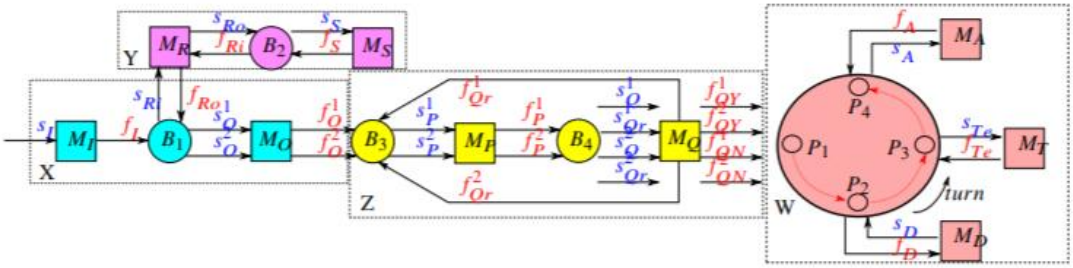


Figura 25 – Versão parcial do sistema flexível de manufatura automotiva com a plataforma giratória no bloco W.

Assim como no caso anterior, apresentado nas Figuras 10 e 11, novamente o exemplo é modelado por MEFs, culminando em uma nova versão G' do modelo da Planta G, que possui 8460 estados, em comparação aos 540 estados da planta sem a plataforma giratória.

O conjunto de especificações E para controlar a planta G', permanece com as mesmas regras apresentadas na Tabela 1, restringindo *overflow* e *underflow* dos blocos X, Y e Z do sistema, porém, faz-se necessário agregar as especificações impostas pelo bloco W, bloco este que é composto pela plataforma giratória. O novo conjunto de especificações é apresentado na Tabela 4. Após feita a modelagem de todas as especificações, o modelo das especificações sincronizadas tem um total de 41310 estados.

Especificação	Descrição
R1	Impedir a rotação da plataforma ao descartar um veículo
R2	Impedir a rotação da plataforma quando o braço robótico está transferindo um veículo
R3	Impedir a rotação da plataforma quando há um veículo sendo testado
R4	Controlar o fluxo de veículos aprovados entre P1 e P2
R5	Controlar o fluxo de veículos aprovados entre P2 e P3
R6	Controlar o fluxo de veículos aprovados entre P3 e P4
R7	Controlar o overflow em P1
R8	Controlar o fluxo de veículos rejeitados entre P1 e P2
R9	Impedir a rotação da plataforma quando não há nenhum veículo

Tabela 4 – Especificações a serem consideradas na plataforma giratória.

O comportamento síncrono K' entre a planta e as especificações é obtido com a composição entre G' e E', ou seja, $K' = G'$ e E'. A partir de K', a síntese do controlador com a plataforma giratória é processada na ferramenta *Supremica*, resultando num supervisor

sup$C'(K',G')$ de 153289 estados e 741736 transições após a síntese. Para fins de comparação, o controlador sup$C'(K',G')$ para o exemplo sem a plataforma giratória possuía apenas 1309 estados e 4878 transições; a adição da plataforma plataforma giratória no final da planta representou um aumento de 11610% no número de estados do controlador após a síntese.

Na sequência, para o ambiente obtido desta planta, foram feitas 100 instâncias de teste com seus parâmetros variados aleatoriamente, a fim de comprovar o desempenho dos algoritmos inteligentes em comparação à ação da TCS sem o suporte inteligente. Os parâmetros foram variados de acordo com a Tabela 5.

Ação:	s_I, $s_O^{1,2}$, $s_P^{1,2}$, $s_Q^{1,2}$, $s_{Qr}^{1,2}$, s_{Ri}, s_{Ro}, s_S, f_I, $f_O^{1,2}$, $f_P^{1,2}$, f_{Ri}, f_{Ro}, f_S		
Recomp.	-10 a -800		
Prob. (%):	Não considerada		
Ação:	**Recomp.**	**Prob. (%)**	**Descrição:**
f_{QN}^1	-20000 a -40000	10 a 90	Rejeição de veículo do tipo 1
f_{QN}^2	-20000 a -40000	10 a 90	Rejeição de veículo do tipo 2
f_{QY}^1	20000 a 40000	10 a 90	Aprovação de veículo de tipo 1
f_{QY}^2	20000 a 40000	10 a 90	Aprovação de veículo de tipo 2
f_{Qr}^1	-1000 a -9000	10 a 90	Retrabalho de veículo de tipo 1
f_{Qr}^2	-1000 a -9000	10 a 90	Retrabalho de veículo de tipo 2

Tabela 5 – Casos de teste aleatórios para o ambiente com a plataforma giratória.

Os eventos de rejeição e de retrabalho de veículos de ambos os tipos tiveram recompensas sempre negativas atreladas a eles, ao passo que os de aprovação tiveram sempre recompensa positiva. As probabilidades de ocorrência dos eventos por sua vez foram variadas na mesma faixa de valores, independentemente do evento ser de aprovação, rejeição ou retrabalho.

O desempenho geral do agente, no sistema modelado pode ser observado na Figura 26, que ilustra a recompensa média obtida pelos agentes de aprendizado por reforço do *N-step* SARSA e pelo de *Deep Q-Learning*, bem como a recompensa obtida sem o uso de métodos inteligentes, apenas com o controle via TCS.

A recompensa média foi escolhida como critério de avaliação de desempenho para comparar os métodos por representar de maneira geral quão bem o agente foi na escolha de priorizar ações que podem lhe trazer um ganho geral positivo.

Primeiramente, a curva em azul expõe a recompensa média obtida utilizando-se apenas a TCS. Pode-se considerar o uso da TCS sem nenhum tipo de otimização como um gráfico de controle para ser possível mensurar a eficácia dos métodos inteligentes. Espera-se que métodos que envolvam aprendizado por reforço apresentem desempenho superior à TCS para problemas que não são totalmente determinísticos, justamente pela TCS não ser capaz de lidar com eventos

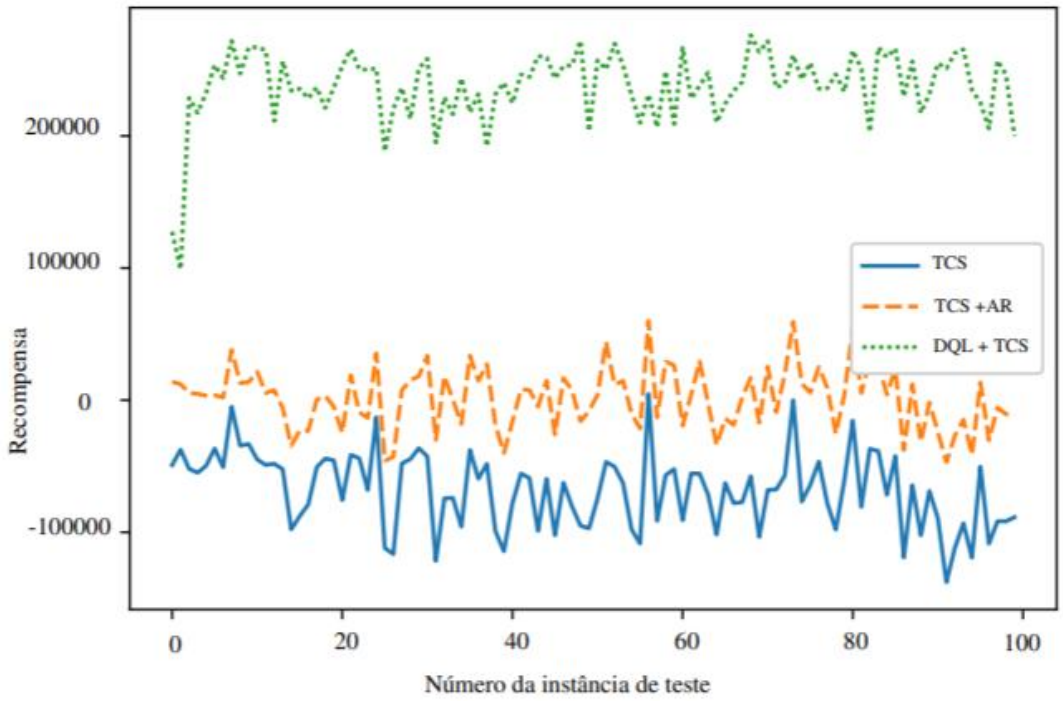


Figura 26 – Teste dos dois algoritmos em comparação com a TCS **pura para 100 instâncias aleatórias.**

probabilísticos. Se a aprovação ou rejeição de veículos tivesse probabilidade de 0% ou 100% de ocorrer quando o agente atinge determinado estado, a TCS seria capaz de lidar com o sistema de forma eficaz, fazendo com que o agente não atingisse os estados que se tem certeza que irão resultar em recompensas negativas; como não trata-se de um caso assim em nenhuma das instâncias de teste — por não haver certeza atrelada aos eventos, e sim uma probabilidade — é esperado que a inclusão de métodos inteligentes deva sempre retornar um ganho de desempenho, em maior ou menor grau.

Observando a Figura 26, de fato pode-se perceber que as recompensas do agente utilizando apenas a TCS permaneceram quase sempre negativas, indicando que, em um sistema probabilístico real, em que pode-se ter influências de questões de mercado para maximizar as recompensas positivas, a ação da TCS sem o suporte de métodos de aprendizado provavelmente não seja uma alternativa eficaz, justamente por ela ser ideal apenas para tratar de sistemas totalmente determinísticos.

Porém, o seu comportamento após integrada ao aprendizado por reforço, com o algoritmo *N-step* SARSA, mostrou-se bastante superior, conforme ilustra a Figura 26, pela linha tracejada em laranja. Pode-se perceber que as recompensas totais foram superiores às do agente utilizando apenas a TCS para todos os casos de teste. No entanto, em números absolutos, estas recompensas ficaram próximas de 0 para a maioria dos casos de teste.

Este comportamento é contrastante com os casos de teste apresentados anteriormente,

antes da incorporação da plataforma giratória ao final da planta. Na Figura 20, por exemplo, pôde-se observar que a TCS juntamente com o *N-step* SARSA tiveram um desempenho muito mais satisfatório, apresentando recompensas acima de 140 mil unidades para todos os casos de teste; na Figura 17, o *N-step* SARSA também apresentou resultados mais sólidos que para os casos de teste com a plataforma giratória ao final da planta.

Acredita-se que este comportamento é justificável exatamente pela inclusão da plataforma giratória no sistema, pois isso fez com que o número de estados de sup$C(K,G)$ aumentasse mais de 117 vezes, o que pode acarretar na perda de performance do *N-step* SARSA, por ser um método tabular.

Por fim, ainda na Figura 26, a linha em verde expõe o comportamento da TCS utilizando DQL para lidar com os eventos probabilísticos. Pode-se constatar que a recompensa média do agente utilizando esta combinação permaneceu alta para todos os casos. A faixa de valores observados como recompensa para o agente implementando DQL foi sempre próxima dos 200 mil, não destoando dos valores obtidos nos exemplos anteriores, como nas Figuras 17 e 20, o que mostra que o grande aumento no espaço de estados, agregado pela inclusão da plataforma giratória, não impactou na performance do método. Isso sugere que a solução aqui proposta se mostra escalável na medida em que a dimensão do sistema tratado aumenta.

Outro ganho de desempenho obtido através da implementação de uma solução utilizando redes neurais trata-se da diminuição do tamanho do espaço utilizado para a execução do treinamento do algoritmo, em comparação com o método *N-step* SARSA, que é tabular. Isso se dá pelo fato da tabela armazenada por este aumentar conforme lida-se com espaços de estados maiores, ao passo que naquele, não há variação de memória.

Conforme abordado, o tamanho da Tabela Q armazenada pelo algoritmo *N-step* SARSA é diretamente proporcional ao número de estados em que o agente pode estar e ações que o agente pode tomar. Mais especificamente, o espaço em memória ocupado por esta tabela é dado por

$$\text{número de estados} * \text{número de ações} * \text{tamanho da unidade de dado (bits)}.$$

Após a síntese do supervisor para o exemplo sem a plataforma giratória agregada no final do sistema, obteve-se sup$C(K,G)$ com 1309 estados; olhando isoladamente para a tabela composta pelos pares de estados e ações, obtêm-se uma tabela de aproximadamente $132KB$ para o tipo de dado *float* de 32 *bits*, que foi utilizado.

Com a inclusão da plataforma giratória, ainda para o *N-step* SARSA, este valor passa a ser de $20466KB$, o que representa uma tabela com um aumento de espaço de 15294%.

A principal vantagem incorporada pelo método com redes neurais é que a memória

ocupada pela rede neural é muito menos variante. Isso se deve ao fato que o único valor que é alterado na contabilização do tamanho da rede é o número de ações do agente, ou seja, a última camada da rede; este tamanho também está muito mais atrelado à arquitetura da rede neural projetada. Ou seja, conforme o tamanho do espaço de estados cresce, e os números de *estados* e *transições* aumentam, o tamanho da rede permanece fixo, exceto nos casos em que novos *eventos* são incluídos.

Para o exemplo sem a plataforma giratória no final, que possui 26 eventos diferentes no sistema, representando 26 ações para o agente, o tamanho da rede neural é de $285KB$, tamanho este que não está ligado ao tamanho do espaço de estados, e unicamente ao número de *eventos* do sistema.

Já para o exemplo com a plataforma giratória, o aumento do tamanho da rede neural é muito menor, justamente por não estar atrelado ao tamanho do espaço de estados, e só ao número de eventos, que passa de 26 a 35. Nesse caso, a rede neural tem $294KB$ apenas, representando um aumento de apenas 3,17%.

Os valores discriminados podem ser observados na Tabela 6.

	Memória sem a plataforma giratória (KB)	**Memória com a plataforma giratória (KB)**	**Aumento (%)**
N-step SARSA	132	20466	15294
Deep Q-Learning	285	294	3,17

Tabela 6 – Comparação da complexidade de espaço entre os métodos, para as diferentes plantas testadas.

5 CONCLUSÕES

A partir das métricas empíricas que foram geradas neste trabalho, apresentadas nos gráficos, os resultados obtidos foram satisfatórios e mostraram duas características importantes de se ressaltar.

A primeira delas é que nos casos de teste iniciais, onde o espaço de estados era menor, a TCS apoiada pelo *Deep Q-Learning* foi mais assertiva em priorizar ações que maximizassem a recompensa do agente, em comparação à TCS apoiada pelo *N-step* SARSA. Isso se concluiu baseando-se nos gráficos das Figuras 20, 21, 22 e 23, por exemplo.

A segunda característica que pôde-se constatar é que para os casos do exemplo expandido, apresentado na Figura 25, onde o espaço de estados tornou-se consideravelmente maior em comparação ao exemplo sem a plataforma giratória, o método apoiado pelo *Deep Q-Learning* manteve seu desempenho satisfatório, obtendo recompensas na mesma faixa de valores que as apresentadas no exemplo sem a plataforma giratória, ao passo que o método apoiado pelo *N-step* SARSA apresentou uma queda de performance. Isso se constatou na Figura 26, para as 100 instâncias de teste aleatórias, que mostra que o *Deep Q-Learning* é um método mais consistente.

Por fim, os resultados analíticos, apresentados na Tabela 6 mostram que a complexidade de espaço aumenta em uma proporção muito maior no método apoiado pelo *N-step* SARSA em comparação ao método apoiado pelo *Deep Q-Learning*, o que justifica a perda de desempenho naquele, em comparação a este.

REFERÊNCIAS

AHMED, M. U. *et al.* A Machine Learning Approach to Classify Pedestrians' Event based on IMU and GPS. **International Journal of Artificial Intelligence**, v. 17, p. 164–167, 2019.

CASSANDRAS, C.; LAFORTUNE, S. **Introduction to Discrete Event Systems**. [*S. l.: s. n.*], jan. 2010. P. 800. ISBN 1441941193. DOI: 10.1007/978-0-387-68612-7.

CHEN, X.; ULMER, M. W.; THOMAS, B. W. Deep Q-learning for same-day delivery with vehicles and drones. **European Journal of Operational Research**, 2021. ISSN 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.06.021. Disponível em: https://www.sciencedirect.com/science/article/pii/S0377221721005361.

CURY, J. E. *et al.* Supervisory control of discrete event systems with distinguishers. **Automatica**, v. 56, p. 93–104, 2015. ISSN 0005-1098. DOI: https://doi.org/10.1016/j.automatica.2015.03.025. Disponível em: https://www.sciencedirect.com/science/article/pii/S0005109815001363.

DANKWA, S.; ZHENG, W. Twin-Delayed DDPG: A Deep Reinforcement Learning Technique to Model a Continuous Movement of an Intelligent Robot Agent, p. 1–5, ago. 2019. DOI: 10.1145/3387168.3387199.

DE ALMEIDA PEREIRA, G. H. *et al.* Active fire detection in Landsat-8 imagery: A large-scale dataset and a deep-learning study. **ISPRS Journal of Photogrammetry and Remote Sensing**, v. 178, p. 171–186, 2021. ISSN 0924-2716. DOI: https://doi.org/10.1016/j.isprsjprs.2021.06.002. Disponível em: https://www.sciencedirect.com/science/article/pii/S092427162100160X.

FAN, J. *et al.* A Theoretical Analysis of Deep Q-Learning. In: BAYEN, A. M. *et al.* (Ed.). **Proceedings of the 2nd Conference on Learning for Dynamics and Control**. The Cloud: PMLR, out. 2020. v. 120. (Proceedings of Machine Learning Research), p. 486–489. Disponível em: http://proceedings.mlr.press/v120/yang20a.html.

FISHER, R. A. THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS. **Annals of Eugenics**, v. 7, n. 2, p. 179–188, 1936. DOI: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1469-1809.1936.tb02137.x. Disponível em: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1469-1809.1936.tb02137.x.

GIANG, H. T. H.; THANH, P. D.; KOO, I. Deep Q-learning-based resource allocation for solar-powered users in cognitive radio networks. **ICT Express**, v. 7, n. 1, p. 49–59, 2021. ISSN 2405-9595. DOI: https://doi.org/10.1016/j.icte.2021.01.008. Disponível em: https://www.sciencedirect.com/science/article/pii/S2405959521000084.

GOMES, L.; MADEIRA, A.; BARBOSA, L. S. A semantics and a logic for Fuzzy Arden Syntax. **Soft Computing**, v. 25, p. 6789–6805, mai. 2021.

GONON, G.; BIMBOT, F.; GRIBONVAL, R. Probabilistic scoring using decision trees for fast and scalable speaker recognition. **Speech Communication**, v. 51, n. 11, p. 1065–1081, 2009. ISSN 0167-6393. DOI: https://doi.org/10.1016/j.specom.2009.02.007. Disponível em: https://www.sciencedirect.com/science/article/pii/S0167639309000247.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [*S. l.*]: MIT Press, 2016. http://www.deeplearningbook.org.

GROOVER, M. **Introduction to Manufacturing Processes**. [*S. l.*]: Wiley, 2011. ISBN 9781118213629.

HARRISON, R.; VERA, D.; AHMAD, B. Engineering Methods and Tools for Cyber–Physical Automation Systems. **Proceedings of the IEEE**, v. 104, n. 5, p. 973–985, 2016. DOI: 10.1109/JPROC.2015.2510665.

HARROU, F.; ZEROUAL, A.; SUN, Y. Traffic congestion monitoring using an improved kNN strategy. **Measurement**, v. 156, p. 107534, 2020. ISSN 0263-2241. DOI: https://doi.org/10.1016/j.measurement.2020.107534. Disponível em: https://www.sciencedirect.com/science/article/pii/S0263224120300713.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. **Journal of Artificial Intelligence Research**, v. 4, p. 237–285, 1996.

KRASHENINNIKOVA, E. *et al.* Reinforcement learning for pricing strategy optimization in the insurance industry. **Engineering Applications of Artificial Intelligence**, v. 80, p. 8–19, abr. 2019. DOI: 10.1016/j.engappai.2019.01.010.

LAPAN, M. **Deep reinforcement learning hands-on : apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more**. Birmingham, UK: Packt Publishing, 2018. ISBN 9781788834247.

LATORRE-BIEL, J. I. *et al.* Combining simheuristics with Petri nets for solving the stochastic vehicle routing problem with correlated demands. **Expert Systems with Applications**, v. 168, p. 114240, 2021. ISSN 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2020.114240. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417420309581.

LI, Z.-M.; XIONG, J. Event-triggered fuzzy filtering for nonlinear networked systems with dynamic quantization and stochastic cyber attacks. **ISA Transactions**, 2021. ISSN 0019-0578. DOI: https://doi.org/10.1016/j.isatra.2021.03.034. Disponível em: https://www.sciencedirect.com/science/article/pii/S0019057821001750.

LUCAS SILVA, A.; RIBEIRO, R.; TEIXEIRA, M. Modeling and control of flexible context-dependent manufacturing systems. **Information Sciences**, v. 421, p. 1–14, 2017. ISSN 0020-0255. DOI: https://doi.org/10.1016/j.ins.2017.08.084. Disponível em: https://www.sciencedirect.com/science/article/pii/S0020025516307836.

MALIK, R.; WARE, S. On the computation of counterexamples in compositional nonblocking verification. **Discrete Event Dynamic Systems**, v. 30, p. 301–334, 2020.

MANU, G. *et al.* Flexible Manufacturing Systems (FMS), A Review. **International Journal of Mechanical and Production Engineering Research and Development**, v. 8, p. 323–336, abr. 2018.

MATSUI, S.; LAFORTUNE, S. **Synthesis of Winning Attacks on Communication Protocols using Supervisory Control Theory**. [*S. l.: s. n.*], 2021. arXiv: 2102.06028 [cs.CR].

MENEZES, P. B. **Linguagens formais e autômatos**. [*S. l.*]: Sagra-Dcluzzato, 1998.

MNIH, V. *et al.* Human-level control through deep reinforcement learning. **Nature**, Springer Science e Business Media LLC, v. 518, n. 7540, p. 529–533, fev. 2015.

MOHAJERANI, S. *et al.* Divergent stutter bisimulation abstraction for controller synthesis with linear temporal logic specifications. **Automatica**, v. 130, p. 109723, ago. 2021. DOI: 10.1016/j.automatica.2021.109723.

MURATA, T. Petri nets: Properties, analysis and applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541–580, 1989. DOI: 10.1109/5.24143.

MURATA, T. Petri Nets: Properties, Analysis and Applications. **Procedings of the IEEE**, v. 77, p. 541–580, 1989.

NOORULDEEN, A.; SCHMIDT, K. W. Order-Preserving Languages for the Supervisory Control of Automated Manufacturing Systems. **IEEE Access**, v. 8, p. 131901–131919, 2020.

OSAKI, K.; HOSOE, Y.; HAGIWARA, T. Metaheuristics-Based Approximation of Two-Dimensional Probability Distributions for Stochastic Systems Control. **IFAC-PapersOnLine**, v. 53, n. 2, p. 4998–5003, 2020. 21th IFAC World Congress. ISSN 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1095. Disponível em: https://www.sciencedirect.com/science/article/pii/S2405896320314713.

P. J. G. RAMADGE, W. M. W. The control of discrete event systems. **Proceedings of the IEEE**, 1989.

PARK, H.-S.; FEBRIANI, R. A. Modelling a Platform for Smart Manufacturing System. **Procedia Manufacturing**, v. 38, p. 1660–1667, 2019. 29th International Conference on Flexible Automation and Intelligent Manufacturing (FAIM 2019), June 24-28, 2019, Limerick, Ireland, Beyond Industry 4.0: Industrial Advances, Engineering Education and Intelligent Manufacturing. ISSN 2351-9789. DOI: https://doi.org/10.1016/j.promfg.2020.01.118. Disponível em: https://www.sciencedirect.com/science/article/pii/S2351978920301190.

PRECUP, R.-E. *et al.* Model-Free Control of Finger Dynamics in Prosthetic Hand Myoelectric-based Control Systems. **Studies in Informatics and Control**, v. 29, p. 399–410, dez. 2020.

PUTERMAN, M. L. **Markov Decision Processes: Discrete Stochastic Dynamic Programming**. 1st. USA: John Wiley & Sons, Inc., 1994. ISBN 0471619779.

PUTTEN, B. *et al.* Supervisor synthesis and throughput optimization of partially-controllable manufacturing systems. **Discrete Event Dynamic Systems**, p. 1–33, nov. 2020. DOI: 10.1007/s10626-020-00325-x.

RABE, M. *et al.* Combining a discrete-event simulation model of a logistics network with deep reinforcement learning. In: PROCEEDINGS of the MIC and MAEB 2017 Conference. [*S. l.: s. n.*], 2017. P. 765–774.

RASCHKA, S. **Python Machine Learning**. [*S. l.*]: Packt Publishing, 2015. ISBN 1783555130.

ROMAN, R.-C. *et al.* Combined Model-Free Adaptive Control with Fuzzy Component by Virtual Reference Feedback Tuning for Tower Crane Systems. **Procedia Computer Science**, v. 162, p. 267–274, 2019. ISSN 1877-0509.

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach, 2002.

SINGH, M. *et al.* Performance of bernoulli's naive bayes classifier in the detection of fake news. **Materials Today: Proceedings**, 2020. ISSN 2214-7853. DOI: https://doi.org/10.1016/j.matpr.2020.10.896. Disponível em: https://www.sciencedirect.com/science/article/pii/S2214785320385333.

SIPSER, M. **Introduction of the Theory of Computation**. Third. [*S. l.*]: CENGAGE Learning, 2013.

STRICKER, N. *et al.* Reinforcement learning for adaptive order dispatching in the semiconductor industry. **CIRP Annals**, v. 67, n. 1, p. 511–514, 2018. ISSN 0007-8506.

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

TEIXEIRA, M.; CURY, J.; QUEIROZ, M. H. de. Exploiting Distinguishers in Local Modular Control of Discrete-Event Systems. **IEEE Transactions on Automation Science and Engineering**, v. 15, p. 1431–1437, 2018.

TEIXEIRA, M.; HERING DE QUEIROZ, M.; CURY, J. Supervisory Control of DES With Extended Finite-State Machines and Variable Abstraction. **IEEE Transactions on Automatic Control**, PP, p. 1–1, jul. 2015. DOI: 10.1109/TAC.2014.2337411.

TIWARI, S.; AL-ASWADI, F. N.; GAURAV, D. Recent trends in knowledge graphs: theory and practice. **Soft Computing**, Springer Science e Business Media LLC, p. 1–9, abr. 2021.

TURNIP, A.; PANGGABEAN, J. H. Hybrid Controller Design based Magneto-rheological Damper Lookup Table for Quarter Car Suspension. **International Journal of Artificial Intelligence**, v. 18, p. 193–206, 2020.

WANG, J. *et al.* A novel fuzzy control with filter-based event-triggered mechanism for nonlinear uncertain stochastic systems suffered input hysteresis. **Fuzzy Sets and Systems**, 2021. ISSN 0165-0114. DOI: https://doi.org/10.1016/j.fss.2021.06.012. Disponível em: https://www.sciencedirect.com/science/article/pii/S0165011421002384.

WASCHNECK, B. *et al.* Optimization of global production scheduling with deep reinforcement learning. **Procedia CIRP**, v. 72, p. 1264–1269, 2018. CIRP Conference on Manufacturing Systems. ISSN 2212-8271.

XUE, Y.; KIECKHAFER, R.; CHOOBINEH, F. Automated construction of GSPN models for flexible manufacturing systems. **Computers in Industry**, v. 37, n. 1, p. 17–25, 1998. ISSN 0166-3615. DOI: https://doi.org/10.1016/S0166-3615(98)00083-9. Disponível em: https://www.sciencedirect.com/science/article/pii/S0166361598000839.

ZHANG, A. *et al.* Dive into Deep Learning. **arXiv preprint arXiv:2106.11342**, 2021.

ZIELINSKI, K. M. C. *et al.* **Suplementary Material**. [*S. l.: s. n.*], 2021. Disponível em: https://bit.ly/3h0KL9m.

ZIELINSKI, K. M. *et al.* Flexible control of Discrete Event Systems using environment simulation and Reinforcement Learning. **Applied Soft Computing**, v. 111, p. 107714, 2021. ISSN 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2021.107714. Disponível em: https://www.sciencedirect.com/science/article/pii/S1568494621006359.

ÍNDICE REMISSIVO