UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Integrando *o Machine Learning* a um sistema de simulação baseada em agentes

Anderson Lino de Paula Martins

Maio de 2022
Itajubá – MG

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Anderson Lino de Paula Martins

Integrando *o Machine Learning* a um sistema de simulação baseada em agentes

Trabalho submetido ao Programa de Pós-Graduação em Engenharia de Produção como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia de Produção.
Área de concentração: Engenharia de Produção
Orientador: Prof. Dr. Alexandre Ferreira de Pinho

Maio de 2022
Itajubá – MG

DEDICATÓRIA

Dedico esse título à minha família, meus pais Almir e Márcia e minha irmã Aline. Também aos meus amigos mais próximos.

AGRADECIMENTOS

Gostaria de primeiramente agradecer à Deus por ter me dado forças durante toda essa trajetória quando eu mais precisei.

Um agradecimento ao meu orientador Alexandre Pinho, que sempre esteve presente e me auxiliou da melhor forma possível ao longo dessa caminhada, sempre disposto a ajudar!

À minha família que sempre esteve tão presente em toda minha vida, me apoiando de todas as formas possíveis. Tanto meus pais Almir e Márcia, como também minha irmã Aline que sempre me deram suporte nos momentos de dificuldades.

Aos meus amigos de pesquisa que sempre me ajudaram e ouviram os inúmeros momentos de desabafo. Como o Alyson, Gustavo, Thais e Bruno, que me acompanharam nessa trajetória.

Aos demais amigos próximos, como o Pedro, o Piuzana e o Patrick, que sempre estiveram ao meu lado, mesmo que não fisicamente, mas sempre dispostos a ajudar e me incentivando.

E um agradecimento que vai mais especial é ao meu amigo Kitty, que quando mais precisei, se mostrou extremamente solicito ao ajudar na pesquisa, até mesmo desprendendo de um final de semana no Rio de Janeiro só para me ajudar!

Mas também aos demais que estiveram presentes e me auxiliaram de outras formas, fazendo que fosse possível chegar hoje onde estou.

RESUMO

A competitividade industrial vem aumentando cada vez mais nos últimos anos e uma das alternativas para enfrentar essa competitividade é a utilização das tecnologias da Indústria 4.0, e entre elas, se encontram a Simulação e o Big Data. O Big Data envolve uma grande geração de dados que necessitam interpretação, no qual pode ser interpretado utilizando algoritmos de aprendizado de máquina a partir do aprendizado por reforço, que pode ser em conjunto com a simulação. A simulação computacional é a incorporação do mundo real em um sistema virtual, absorvendo as características fundamentais, e um dos métodos de simulação é a Simulação Baseada em Agentes, no qual o agente é o foco do sistema. Nesse contexto, esse trabalho propõe explicar como é possível integrar o aprendizado de máquina a um sistema de simulação baseada em agentes. Sendo assim, será utilizada uma ferramenta de auxílio ao modelador, mostrando duas formas para realizar essa implementação no *software* AnyLogic®. A primeira forma será utilizando uma ferramenta externa, o Pathmind, e para isso será criado um sistema que gera caixas de três cores diferentes (vermelho, verde e azul), representado por vetores de forma aleatória. O sistema deverá ser capaz de identificar qual a cor da caixa, sendo o foco na descrição das etapas a serem seguidas para realizar a implementação utilizando essa ferramenta. O teste de eficiência da ferramenta foi dado a partir do número de acertos que a máquina é capaz de realizar, e o resultado encontrado mostrou uma alta eficiência por parte dessa ferramenta. Uma vez que antes da implementação do aprendizado de máquina, o sistema agia de forma aleatória, acertando as cores seguindo a probabilidade estatística de aleatoriedade prevista para esse problema, que era de 12,5%, e após a implementação, o sistema alcançou uma taxa de acerto de 100%, fica evidente a eficiência da ferramenta. A segunda forma será de forma direta no *software* AnyLogic®, utilizando linguagem de programação Java por meio do algoritmo de aprendizagem por reforço *Q-learning*, que foi desenvolvido nessa pesquisa. Para isso será utilizado a mesma base do modelo computacional anterior, porém, para essa aplicação serão criadas caixas de cinco cores diferentes (vermelho, verde, azul, branco e preto), e serão representadas por meios de *strings*, no qual o sistema busca acertar a cor da caixa a partir do aprendizado de máquina utilizando o algoritmo *Q-learning* e utilizando a matriz de resultado Q. E assim como na forma utilizando a ferramenta externa, a ênfase será na demonstração de todas as etapas a serem seguidas para concluir essa implementação. O sistema novamente se mostrou eficiente sendo capaz de identificar de forma correta em todas as tentativas. Então esse trabalho conseguiu mostrar duas formas eficientes de implementar o aprendizado por reforço no software AnyLogic®, utilizando uma ferramenta externa e de forma direta, no qual a primeira necessita um nível inferior de conhecimento de aprendizado de máquina e programação, se mostrando mais simples, porém, é *black box*, enquanto da segunda forma é o contrário, exigindo um nível alto de conhecimento de aprendizado de máquina e programação, porém com código aberto.

Palavras-chave: Simulação computacional, simulação baseada em agentes, aprendizado de máquina, aprendizado por reforço, algoritmo *Q-learning*.

ABSTRACT

Industrial competitiveness has been increasing more and more in recent years and one of the alternatives to face this competitiveness is the use of Industry 4.0 technologies, and among them are Simulation and Big Data. Big Data involves a large generation of data that needs interpretation, which can be interpreted using machine learning algorithms from reinforcement learning, which can be in conjunction with simulation. Computer simulation is the incorporation of the real world into a virtual system, absorbing the fundamental characteristics, and one of the simulation methods is Agent-Based Simulation, in which the agent is the focus of the system. In this context, this work proposes to explain how it is possible to integrate machine learning to an agent-based simulation system. Being a tool to aid the modeler, showing two ways to carry out this implementation in AnyLogic® software. The first way will be using an external tool, the Pathmind, for that, a system will be created that generates boxes of three different colors (red, green and blue), represented by vectors, at random. The system must be able to identify the color of the box, focusing on the description of the steps to be followed to carry out the implementation using this tool. The tool efficiency test was given based on the number of adjustments that the machine is capable of performing, and the result found showed a high efficiency by this tool. Since before the implementation of machine learning, the system acted randomly, matching the colors following the statistical probability of randomness predicted for this problem, which was 12.5%, and after the implementation, the system reached a rate 100% hit. The second way will be directly in the AnyLogic® software, using Java programming language through the Q-learning reinforcement learning algorithm, which was developed in this research. For this, the same basis as the previous computational model will be used, however, for this application, boxes of five different colors will be created (red, green, blue, white and black), and will be represented through strings, in which the system seeks to hit the right box color from machine learning using the Q-learning algorithm and using the Q result matrix. And as with the form using the external tool, the emphasis will be on demonstrating all the steps to be followed to complete this implementation. The system again proved to be efficient, being able to correctly identify in all attempts. So this work was able to show two efficient ways to implement reinforcement learning in AnyLogic® software, using an external tool and in a direct way, in which the first one needs a lower level of knowledge of machine learning and programming, proving to be simpler, however, it is black box, while the second way is the opposite, requiring a high level of knowledge of machine learning and programming, but with open source.

Key words: Computer simulation, agent-based simulation, machine learning, reinforcement learning, Q-learning algorithm.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

A3C	*Asynchronus Actor-Critic Algorithm*
DDPG	*Deep Deterministic Policy Gradient*
DQN	*Deep Q-Network*
GPU	*Graphics Processing Unit*
I4.0	Indústria 4.0
IA	Inteligência Artificial
ML	*Machine Learning*
MLE	Mercados Locais de Energia
NAF	*Q-learning with Normalized Advantages Functions*
PLE	*Personal Learning Edition*
PPO	*Proximal Policy Optimization*
RGB	*Red Green and Blue*
RL	*Reinforcement Learning*
SaaS	*Software as a Service*
SBA	Simulação Baseada em Agentes
SBPO	Simpósio Brasileiro de Pesquisa Operacional
SD	Simulação Dinâmica
SED	Simulação a Eventos Discretos
SEICR	*Susceptible-Exposed-Infectious-Confirmed-Recovered*
SEIR	*Susceptible-Exposed-Infected-Recovered*
SGD	*Stochastic Gradient Descent*
TRPO	*Trust Region Policy Optimization*

SUMÁRIO

1. INTRODUÇÃO

1.1 Contextualização e questões de pesquisa

O setor industrial atualmente alterou sua visão geral por conta da recessão global, fazendo com que haja uma perspectiva focada para o seu real valor agregado criado (ALCÁCER e CRUZ-MACHADO, 2019). O termo Indústria 4.0 (I4.0) surgiu na Alemanha, na Feira de Hannover em 2011, e atualmente está ocorrendo o auge dessa quarta revolução industrial, uma vez que vem sendo altamente utilizada nos dias atuais (XU, XU e LI, 2018). Segundo Liao *et al*. (2017), essa revolução tem ganhado cada vez mais força e se tornando mais presente em todo o mundo.

Alcácer e Cruz-Machado (2019) citam as diversas tecnologias e características presentes em uma I4.0 e entre elas estão a Simulação e o *Big Data*. O *Big Data* está presente há algumas décadas, sendo que teve um crescimento exponencial nesta última, momento em que a sociedade está passando por uma rápida revolução digital (SONG e ZHU, 2018). Yang *et al*. (2022) afirmam que o *Big Data* beneficia na visibilidade das informações através de sua automação, onde por meio de suas previsões é capaz de auxiliar no processo industrial, podendo fazer com que aumente a produção, vida útil e os ciclos de um produto, e que o volume de dados presente é muito grande, podendo ser *Terabytes, Exabyte* ou *Zettabytes* de dados.

O Aprendizado de Máquina (*Machine Learning* - ML) possui diversas aplicações, porém, recentemente uma dessas tem ganhado um grande destaque, que é na área do *Big Data*. O ML tem se mostrado muito eficiente no auxílio da interpretação da grande quantidade de dados (ZHOU *et al*., 2017). Para Lu *et al*. (2021) existem algumas áreas principais de aplicação para o *Machine Learning*/Inteligência Artificial, sendo que uma delas é a sua utilização para um conjunto de dados de simulação e experimentos de alta fidelidade para construir modelos para utilizar em simulação computacional.

A simulação é uma técnica de modelagem fundamentalmente experimental ou empírica, criando modelos capazes de relatar o comportamento do que for interessante em um ambiente computacional, permitindo que possa ser realizada observações desses comportamentos, e também capaz de realizar testes e comparações de cenários e *designs* alternativos, validando, explicando e apoiando os resultados da simulação e estudar recomendações de melhorias para o sistema (WHITE e INGALLS, 2018).

Existem três principais métodos de simulação, a Simulação Dinâmica, Simulação à Eventos Discretos e Simulação Baseada em Agentes (SBA), sendo essa última relativamente recente, mas que vem ganhando cada vez mais força nos últimos anos, uma técnica que é classificada como

bottom-up (de baixo para cima), onde a modelagem é voltada para o agente, que tem papel central e que possui comportamento individual (SUMMARI *et al*., 2013).

A SBA é uma técnica que facilita a interação de agentes autônomos entre eles, sendo uma modelagem de diferentes agentes autônomos com diferentes objetivos (BEAN e JOUBERT, 2021). Muñoz e Iglesias (2021) afirmam que o uso da técnica de SBA é bastante útil quando no sistema há uma complexa interação entre agentes, populações heterogêneas e comportamentos complexos.

Augustijn *et al*. (2020) afirmaram que o uso de algoritmos de *Machine Learning* para enriquecer modelos baseado em agentes tem aumentado com o passar dos anos. Os autores ainda comentam que essa integração agrega valor ao combinar as vantagens da abordagem baseada em dados e as possibilidades de explorar situações futuras e intervenções humanas, porém essa integração ainda está em estágio inicial e que essa integração de forma total muitas vezes se torna tecnicamente desafiadora.

Segundo Kavak *et al*. (2018) existe um grande aumento na quantidade de dados que podem ser desenvolvidos empiricamente com o auxílio da SBA, porém, na literatura não há uma oferta grande de trabalhos de modelagem estruturada direcionadas a essa área. Para evidenciar isso, foi realizado uma pesquisa acerca de trabalhos que mostraram como realizar a integração entre ML e SBA de forma mais clara, o que pode ser visualizado na Tabela 1.

Quadro 1- Trabalhos que abordam a sistemática entre a integração de ML e SBA

Trabalho	**Descrição**
Farhan, Göhre e Junprung (2020)	O trabalho demonstra como é o procedimento para inserção do algoritmo de ML em um modelo SBA com o auxílio do Pathmind, no qual, é treinado um barista e que precisa aumentar sua eficiência.
Pinciroli *et al*. (2020)	Os autores afirmam que que o aprendizado por reforço (*Reinforcement Learning* - RL) pode ser integrado a modelos de SBA, porém a utilização de algoritmos de RL requer habilidades específicas e não se dá de uma maneira direta e intuitiva, e no trabalho foi mostrado como fazer essa integração, no caso também utilizando o Pathmind, no qual os autores otimizaram uma política de otimização e manutenção em um parque eólico.
Zhang, Valencia e Chang (2021)	Os autores mostram um revisão abrangente da aplicação do ML em modelo de SBA, onde são investigados os algoritmos,

	estruturas, procedimentos de implementação e aplicações multidisciplinares para quatro cenários diferentes: microagente com aprendizagem de consciência situacional, microagente com intervenções de comportamento, emulador de emergência no nível macro e tomada de decisão no nível macro para SBA.
Zhang *et al*. (2016)	É demonstrado a implementação do ML em um sistema multiagentes, para isso foi aplicado na predição da adoção de sistema de energia solar em telhados residenciais.
Rand (2006)	O autor mostra como diferentes técnicas de ML podem ser incorporadas a alguns modelos de SBA, então mostra diretrizes de como integrar o ML a modelo SBA e discute as complicações que podem existir, para isso utiliza o Problema do El Farol Bar.
Prasanna, Holzhauer e Krebs (2019)	Este trabalho traz uma revisão de literatura sobre os métodos comuns de integração do ML e métodos baseados em dados a modelos de SBA de mercados de energia, e também discute a respeito dos requisitos para realizar essa integração e apresenta os métodos utilizados na literatura para atender a esses requisitos.

Fonte: Autor.

Esse trabalho propõe a implementação utilizando o *software* AnyLogic®, porque, segundo os autores Siebers *et al*. (2010) e Brailsford (2014), esse era o único *software* disponibilizado naquela época para fins comerciais. Mas ainda sim, até os dias atuais, o AnyLogic® segue sendo a ferramenta mais utilizada para a modelagem baseada em agentes, por isso será o *software* utilizado nessa pesquisa.

Então, nesse contexto esse trabalho buscará responder à seguinte questão de pesquisa: como é possível realizar a integração do aprendizado de máquina (*Machine Learning*) em um modelo baseado em agentes?

1.2 Objetivos

Considerando o contexto e a questão de pesquisa, esse trabalho teve como objetivo geral desenvolver um modelo de implementação de *Machine Learning* integrado a um modelo baseado

em agentes. Cabe ressaltar que a pesquisa se limita a demonstrar essa integração do ML com a SBA com o *software* AnyLogic®, que é o que será utilizado nesse modelo.

Dado o objetivo geral, essa pesquisa possuiu como objetivos específicos:

- Mostrar as etapas de implementação da integração utilizando uma ferramenta externa ao AnyLogic®, o Pathmind;
- Desenvolver um algoritmo de aprendizado por reforço Q-learning em linguagem de programação Java;
- Mostrar as etapas de implementação da integração de forma direta ao AnyLogic®, utilizando linguagem de programação Java.

1.3 Justificativa

Diante do contexto apresentado, fica evidente a crescente pesquisa a respeito de Simulação Baseada em Agentes, sobre *Machine Learning* e, também, combinando as duas técnicas. Para evidenciar isso em termos de pesquisa científica, foi elaborado uma pesquisa bibliométrica em fevereiro de 2022 sobre os termos de forma isolada e a combinação entre eles, como mostrado na Tabela 2. Vale destacar que foi utilizado o termo em inglês para SBA (*Agent Based Simulation*) por se tratar de bases científicas internacionais e as bases utilizadas para isso foram *Scopu*s e *Web of Science*, uma vez que segundo os autores Franceschini, Maisano e Mastragiacomo (2014), essas duas são consideradas os principais bancos de dados multidisciplinares do mundo.

Tabela 1 – Número de artigos encontrados na análise bibliométrica dos termos chave dessa pesquisa

Base de Dados	*Agent Based Simulation*	*Machine Learning*	*Agent Based Simulation + Machine Learning*	*Agent Based Simulation + Machine Learning + Framework*	*Agent Based Simulation + Machine Learning + Tutorial*
Scopus	7.270	330.781	465	251	43
Web of Science	4.682	252.106	81	19	0

Fonte: Autor.

Segundo a Tabela 1, existem trabalhos acadêmicos que integram a SBA com o ML, dentre eles, é possível destacar alguns recentes, como: Augustijn *et al*. (2020), Batata, Augusto e Xie (2018), Bose *et al*. (2021), Bosse (2021), Darville e Celik (2020), Farhan, Göhre e Junprung (2020), Fuller, de Arruda e Ferreira Filho (2020), Hassanpour *et al*. (2021), Irannezhada, Prato e Hickman

(2020), Källström (2020), Moriyama *et al*. (2019), Olave-Rojas e Nickel (2019), Pinciroli *et al*. (2020), Rößler *et al*. (2020) e Sena *et al*. (2017).

Dois dos trabalhos supracitados, Farhan, Göhre e Junprung (2020) e Pinciroli *et al*. (2020), utilizaram a ferramenta Pathmind, e eles destacam como essa ferramenta não apresenta qualquer limitação para a implementação do ML em um modelo SBA utilizando o AnyLogic®, mostrando que essa ferramenta é eficaz, e ao mesmo tempo, ela facilita esse processo de implementação, pois não exige alto nível de conhecimento de ML por parte do modelador.

É importante também destacar que essa ferramenta está disponível para a versão gratuita do *software* AnyLogic® (versão PLE – *Personal Learning Edition*), porém isso só ocorreu em maio de 2021, fazendo com que essa seja uma ferramenta recentemente disponível para o público em geral.

Segundo Farhan, Göhre e Junprung (2020) e Pinciroli *et al*. (2020), a ferramenta Pathmind se mostrou bastante eficiente. Porém, esse trabalho buscou também uma segunda forma de demonstração da implementação, no caso de forma direta ao AnyLogic®, sem o uso de qualquer ferramenta externa ao *software*.

Assim, esse trabalho se torna academicamente justificado, pois está cada vez mais comum a integração entre SBA e ML, apesar de essa não ser uma tarefa simples de ser executada. Então, será cientificamente relevante mostrar as etapas para realizar essa integração dentro do *software* AnyLogic®, mostrando das duas formas possíveis: utilizando uma ferramenta externa e realizar de forma direta no *software*.

1.4 Estrutura do Trabalho

Esse trabalho será dividido em outros quatro capítulos, além desse introdutório. No Capítulo 2 apresenta-se a fundamentação teórica. Esse capítulo foi dividido de forma que o primeiro conceito a ser apresentado foi a simulação computacional, em seguida discute-se a respeito de Simulação Baseada em Agentes e também algumas aplicações recentes dessa ferramenta em diferentes áreas, em seguida sobre *Machine Learning* e na última etapa desse capítulo são apresentados alguns trabalhos recentes que envolvem a combinação de Simulação Baseada em Agentes e *Machine Learning*. O Capítulo 3 traz o método de pesquisa a ser utilizado, com a classificação dessa pesquisa quanto ao método e, também, como essa pesquisa foi conduzida, mostrando as etapas seguidas. No Capítulo 4, encontra-se o desenvolvimento da pesquisa. Mostra-se, inicialmente, como realizar a implementação utilizando a ferramenta Pathmind e, em seguida, como realizar essa implementação sem o uso de uma ferramenta externa. Por fim, no Capítulo 5, são apresentadas as conclusões dessa

pesquisa e também as sugestões para trabalhos futuros, seguido das referências bibliográficas utilizadas para a elaboração desse texto.

2. REFERENCIAL TEÓRICO

Nessa seção se mostrará o embasamento teórico para o desenvolvimento dessa pesquisa. Inicialmente, se discutirá sobre a simulação computacional. Então, será abordada uma das técnicas de simulação, a Simulação Baseada em Agentes (SBA), que é o foco dessa pesquisa, bem como suas vantagens e desvantagens e, também, algumas aplicações recentes. A seguir, discorre-se a respeito de *Machine Learning*, detalhando-se oaprendizado por reforço. Se discutirá sobre a ferramenta externa utilizada, o Pathmind. E por fim serão apresentados alguns trabalhos recentes que utilizaram o ML integrado a modelos de SBA.

2.1 Simulação Computacional

Primeiramente, é importante destacar a diferença entre simulação não computacional e simulação computacional. Como os próprios nomes dizem, elas são respectivamente, aquela que não necessita de um computador para ser realizada, como por exemplo, pode ser realizada a partir do uso de um protótipo, e a segunda é aquela que necessita do uso de um computador para ser realizada (CHWIF e MEDINA, 2015). Este trabalho discute somente a respeito da simulação computacional.

A simulação e as aproximações analíticas se diferem na resolução de modelos. As resoluções analíticas implicam na criação e resolução de fórmulas e equações matemáticas que descrevem o sistema, sendo essa técnica preferível em relação à simulação quando possível de ser aplicada, porém, na grande maioria dos sistemas complexos, é impossível, sendo assim necessário a utilização da simulação (WHITE e INGALLS, 2018).

Melão e Pidd (2006) relatam que a simulação vem sendo amplamente utilizada em processos industriais desde os anos de 1990, uma vez que essa é capaz de criar modelos que reproduzem um processo industrial real em um ambiente computacional, permitindo analisar diferentes cenários e *designs*.

Nesse contexto, Montevechi *et al*. (2007) descrevem a simulação como uma representação da realidade em um ambiente controlado, a qual pode ser estudada sob diversas condições, sem que sejam necessários altos custos ou algum risco físico. Baines *et al*. (2004) retratam a simulação de forma similar, pois segundo eles, essa é uma técnica de construção de um modelo capaz de retratar um sistema do mundo real, fazendo com que esse modelo possa ser utilizado para testar o desempenho do sistema real em diferentes condições de operações.

White e Ingalls (2018) definem um modelo como sendo uma entidade que é utilizada para representar uma outra entidade, com uma finalidade definida, que de forma geral são abstrações simplificadas. Os autores afirmam que os modelos devem ser utilizados quando a investigação do sistema real não é possível ou quando não se é aconselhável, seja porque essa alteração no sistema real é cara, lenta, interruptiva, insegura ou até mesmo ilegal.

Robinson (2011) sugere que o modelo deva possuir o máximo de detalhes possível, pois assim estará, possivelmente, mais próximo do sistema real e, consequentemente, sua eficácia será maior, uma vez que esse é uma abstração simplificada de um sistema real. A simulação é a representação de um sistema real através de um modelo, permitindo análises sem a necessidade de alteração no sistema real (HARREL, GHOSH e BOWDEN, 2004).

Jahangirian *et al*. (2010) afirmam que a simulação é a principal técnica no apoio a tomada de decisão em um processo de *design* de uma cadeia de suprimentos, isso devido a sua grande flexibilidade. Assim como Shannon (1998), que alega que a simulação é capaz de responder questões sobre "o que aconteceria se?" (*what-if questions*).

Em meio a tantas definições sobre o que é a simulação, Chwif e Medina (2015) optaram pelo caminho contrário, listando o que a simulação não é:

- Uma bola de cristal, pois a simulação não é capaz de prever o futuro. Ela é capaz de prever com certa confiança o comportamento de um sistema, isso a partir de dados fornecidos na entrada;
- Um modelo matemático, pois a simulação até pode conter fórmulas matemáticas, mas não existe uma expressão analítica fechada ou um conjunto de equações capazes de gerar uma resposta analítica direta para o sistema;
- Uma ferramenta diretamente de decisão, pois essa ferramenta não é capaz por si de gerar uma resposta ótima para o sistema, sendo a sua função a de gerar cenários que serão analisados;
- Uma técnica de último recurso, pois antigamente acreditava-se que a simulação só deveria ser empregada quando todas as outras técnicas não funcionassem, porém, atualmente a simulação já é uma das principais técnicas da pesquisa operacional (PO) e da ciência da administração;
- Uma panaceia, pois ela não é capaz de resolver todos os problemas possíveis, ela é aplicada a problemas bem específicos e se adequa muito bem.

Kovacic e Pecek (2007) afirmam que a simulação tem aumentado suas aplicações em processos que são imprevisíveis, aqueles em que as pessoas possuem o papel principal, como nas

áreas de saúde, industrial, modelagem de tráfego, militar, etc. Assim como também para Zhu *et al*. (2015), que relatam que estão acontecendo rápidos avanços para resolução de problemas de sistemas complexos na simulação, uma vez que segundo eles, esta ferramenta é uma das mais importantes no estudo de sistemas complexos.

Segundo Kelton e Law (2000), os modelos de simulação podem ser classificados em três dimensões:

1. Estáticos ou dinâmicos, sendo os modelos estáticos aqueles que ocorrem em um tempo em particular, onde o tempo não possui relevância, enquanto os dinâmicos, envolvem o tempo, em que o sistema evolui no decorrer do tempo;
2. Determinísticos ou estocásticos, sendo os modelos determinísticos aqueles que não possuem probabilidade, possuindo sua saída determinada, enquanto os estocásticos são aqueles que possuem probabilidades;
3. Contínuos e discretos, sendo que na simulação contínua o sistema acompanha continuamente a dinâmica, sem saltos discretos, enquanto nos discretos os eventos e os estados das variáveis se alteram instantaneamente em pontos separados do tempo.

Banks *et al*. (2005) defenderam que a simulação muitas vezes é utilizada na resolução de problemas, uma vez que essa ferramenta é capaz de imitar um sistema real. Os autores ainda comentam que essa ferramenta possui algumas vantagens como o fato de permitir a exploração de novos cenários sem a necessidade de interrupção do sistema real, testar novos *designs* e *layouts* sem precisar da aquisição de recursos, permite a aceleração ou desaceleração no tempo, auxilia na análise dos gargalos da produção e facilita a análise de como o sistema realmente opera.

A simulação facilita a experimentação e o estudo de diversos cenários nas empresas, sendo capaz de prover benefícios tangíveis e intangíveis, com vantagens que podem ser na qualidade, tempo, redução de custos, inovações, atendimento ao cliente ou na performance do produto, que a longo prazo podem criar lucros para a empresa (PECEK e KOVACIC, 2011).

Apesar de todas as vantagens, a simulação também apresenta algumas desvantagens, Banks *et al*. (2005) citaram algumas, como o fato da construção do modelo necessitar de um treinamento especial, os resultados da simulação podem ser difíceis de serem interpretados, a análise e a modelagem na simulação podem ser caros e consumir muito tempo e também que a simulação as vezes é utilizada quando apenas um resultado analítico seria possível.

Existem três principais métodos de simulação: Simulação Dinâmica (SD), Simulação a Eventos Discretos (SED) e Simulação Baseada em Agentes (SUMARI *et al*., 2013). Os autores

discutem a respeito dos recursos, vantagens, desvantagens e *softwares* utilizados em cada uma delas, segundo eles:

- A Simulação Dinâmica é utilizada para entendimento do comportamento de um sistema a longo prazo e é focada no fluxo do sistema. Ela facilita o entendimento de sistemas complexos, melhorando a identificação de fatores relevantes nesses sistemas, porém se torna complexa de compreender quando se trata de um sistema muito grande. Os *softwares* mais utilizados são Vensim® e Stella®;
- A Simulação a Eventos Discretos é utilizada em sistemas onde é muito evidente o problema de filas, utilizando uma abordagem de cima para baixo, sendo bem utilizada no comportamento das entidades, sendo mais direto para ser modelada, porém não é muito boa na modelagem do comportamento humano. Alguns dos *softwares* mais utilizados são o Arena, ProModel®, Simul8® e FlexSim®;
- A Simulação Baseada em Agentes é utilizada para identificar melhor a relação e operação entre as entidades de forma mais real, focada nas interações que ocorrem no sistema. Utilizando uma abordagem de baixo para cima, ela é capaz de capturar fenômenos emergentes e descrever o sistema de formas naturais, porém necessita de um alto nível de habilidades em computação. O *software* mais utilizado é o AnyLogic®.

Assim, foram apresentados nesta seção os conceitos de modelagem computacional, mostrando como essa ferramenta é poderosa por conseguir simular um sistema real e um computador, assim se tornando vantajosa por permitir alterações nesse sistema simulado sem a necessidade de alteração no sistema real, porém uma desvantagem é necessitar de um conhecimento específico para realiza-la e, por fim, seus três principais métodos. Na seção seguinte se discutirá de forma mais aprofundada a Simulação Baseada em Agentes, uma vez que essa é o foco desse trabalho.

2.2 Simulação Baseada em Agentes (SBA)

A Simulação Baseada em Agentes é uma técnica de modelagem relativamente nova, que vem ganhando cada vez mais força nos últimos anos, e esse crescimento é evidenciado pelo aumento do número de publicações na área (MACAL e NORTH, 2009). A técnica de SBA é recente se comparada aos outros métodos clássicos, como a Simulação a Eventos Discretos, utilizada para simular sistemas com interação entre agentes autônomos, e sua definição pode variar de acordo com o campo de estudo (ou até mesmo no mesmo campo), mas a sua aplicação é sempre semelhante,

isto é, a simulação de objetos autônomos (agentes) que são capazes de identificar, explicar, gerar e projetar comportamentos emergentes (CHAN, SON e MACAL, 2010).

Bonabeu (2002) afirma que a modelagem baseada em agentes é uma ferramenta de modelagem de simulação muito poderosa, onde um sistema é modelado como uma coleção de entidades autônomas de tomada de decisão, chamados de agentes. Assim como também para Railsback, Lytinen e Jackson (2006), que defendem que o uso dessa ferramenta está aumentando em diversos campos, e que essa explicação se deve ao fato dessa ferramenta ser capaz de resolver problemas que métodos mais convencionais não são.

Segundo Dubiel e Tsimhoni (2005), a SBA busca simular entidades inteligentes e autônomas, os agentes, e como eles interagem buscando o mesmo objetivo no ambiente. Os autores afirmam que essa técnica de simulação vem sendo utilizada em diversas situações, como evolução social, segregação, propagação de doenças e eficácia de propagandas, tornando-se um tópico bastante pesquisado na indústria de simulação.

Para Siebers *et al*. (2010), a SBA auxilia no entendimento de sistemas do mundo real em que a representação ou modelagem de diversos indivíduos é importante e cada um desses indivíduos possuí comportamento autônomo. Um modelo típico de SBA possui três elementos: um conjunto de agentes (incluindo seus atributos e comportamentos), um conjunto de relações entre agentes e métodos de interação (definindo com quem e como os agentes interagem) e o ambiente do agente (BRAILSFORD, 2014).

Existem quatro razões pelo qual a SBA vem crescendo muito: primeiro, porque há um aumento da complexidade dos sistemas a serem analisados em termos de interdependência e as outras técnicas de modelagem podem já não ser tão aplicáveis como antes; segundo, porque alguns sistemas estão se tornando muito complexos para podermos modelar adequadamente, como a modelagem de sistemas econômicos, que são modelos em forma de mercados perfeitos, com agentes homogêneos e equilíbrio de longo prazo, então com a SBA é capaz de se relaxar algumas dessas suposições; terceiro, porque os dados estão sendo coletados de forma cada vez mais precisos e; quarto, e mais importante, é que há um grande avanço computacional, permitindo a simulação em grande escala de micro simulações, o que não era possível antes (MACAL e NORTH, 2009).

A SBA se caracteriza pela perspectiva de modelagem de sistema *bottom-up* (de baixo para cima), no qual a modelagem do sistema é feita modelando as entidades individuais que compõem o sistema e suas interações (MACAL, 2010). Negahban e Yilmaz também afirmam que a SBA utiliza a abordagem *bottom-up*, em que os elementos centrais do modelo são os agentes.

Siebers *et al*. (2010) citam sete atributos para se classificar um modelo como sendo de SBA:

1. Baseado no indivíduo, sendo o foco da modelagem nas entidades e nas interações entre elas;
2. Abordagem de modelagem *bottom-up*;
3. Cada agente possui sua linha de controle (descentralizado);
4. Entidades ativas, isto é, cada entidade pode por si própria tomar iniciativa para fazer algo, a inteligência é representada dentro de cada entidade individual;
5. Não possui o conceito de filas;
6. Não possui o conceito de fluxo, macro comportamento não é modelado, ele surge de micro decisões de agentes individuais;
7. Os dados de entrada normalmente são baseados em teorias ou dados subjetivos.

O fundamental na construção da modelagem na SBA são os agentes e seus comportamentos, sendo que estes afetam as próprias ações dos agentes e as ações dos outros agentes e dos ambientes, uma vez que o recurso mais importante e distintivo da SBA é a perspectiva do agente que é assumida ao visualizar qualquer sistema consistido de agentes (MACAL, 2016).

A SBA possui conexões com diversas áreas, incluindo as ciências complexas, a ciência de sistemas, sistemas dinâmicos, ciência da computação e outros, ela também está ligada a inteligência artificial (MACAL e NORTH, 2009). Borschev e Filippov (2004) também defendem que a SBA vem sido desenvolvida em diferentes campos, como a inteligência artificial, ciências complexas, teoria dos jogos e outros.

Não há uma definição universal sobre o termo agente no contexto da SBA, sendo esse um assunto de muito debate e não só no meio acadêmico (MACAL e NORTH, 2009). Para Dong, Liu e Lu (2012), os agentes podem representar pessoas, companhias, projetos, entre outros, uma vez que esses possuem comportamentos, memórias, contatos, etc. E Bonabeau (2002) afirma que cada agente é capaz de avaliar individualmente sua situação e tomar decisões, podendo executar vários comportamentos apropriados para o sistema que representam.

Macal e North (2009) afirmam que os agentes são diferentes, heterogêneos e dinâmicos em seus atributos e regras de comportamento, como mostrado na Figura 1. Os autores consideram que os agentes devem possuir certas propriedades e atributos:

- Um agente é autônomo e autodirigido: um agente pode funcionar independentemente em seu ambiente e em suas interações com outros agentes, geralmente a partir de uma gama limitada de situações de interesse, e o comportamento de um agente é como a representação de um processo que une a percepção do agente ao ambiente e às suas decisões e ações;

- Os agentes são modulares ou independentes: um agente é um indivíduo identificável e discreto com um conjunto de características ou atributos, comportamentos e capacidade de tomada de decisão, além disso, tem uma fronteira em um sentido e pode facilmente determinar se algo (isto é, um elemento do estado do modelo) é parte de um agente, se não é parte de um agente ou se é uma característica compartilhada entre agentes;
- Um agente é social, interagindo com outros agentes: agentes possuem protocolos ou mecanismos que descrevem como eles interagem uns com os outros, assim como um agente possui comportamentos, isto é, os protocolos de interação de agentes comuns incluem contenção por espaço e prevenção de colisões, reconhecimento de agente, comunicação e troca de informações, influência, e outros mecanismos específicos de domínio ou aplicativo.

Os autores ainda afirmam que os agentes podem possuir propriedades adicionais, no qual podem ou não ser consideradas como definição de propriedades ou necessidade para o agente:

- Um agente pode viver em um ambiente: agentes interagem com seus ambientes assim como com outros agentes. Um agente está situado, no sentido de que seu comportamento é dependente da situação, o que significa que seu comportamento é baseado no estado atual de suas interações com outros agentes e com o ambiente;
- Um agente pode possuir metas explícitas que orientam o seu comportamento: as metas não são necessariamente objetivos para maximizar, tanto quanto os critérios com os quais para avaliar a eficácia de suas decisões e ações. Isso faz com que o agente sempre modifique seu comportamento com a intenção de melhorar os resultados;
- Um agente pode ter a habilidade de aprender e adaptar seu comportamento baseado em experiências: o aprendizado individual e adaptação requer ao agente possuir memória, geralmente na forma de um atributo dinâmico ao agente;
- Os agentes normalmente possuem atributos de recurso que indicam seu estoque atual de um ou mais recursos.

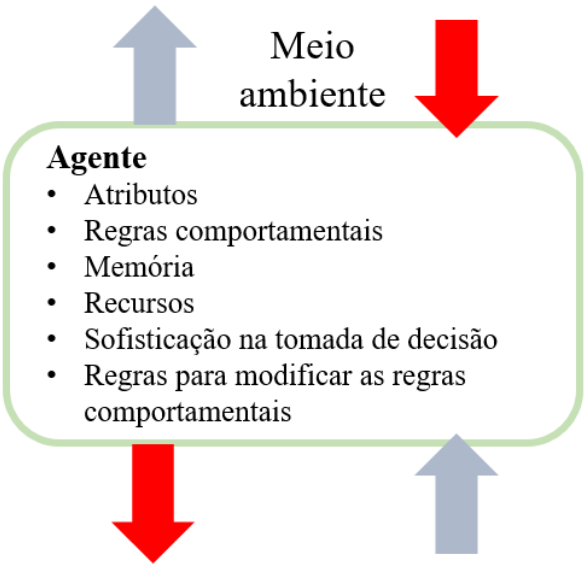


Figura 1 - Um típico agente
Fonte: adaptado de Macal e North (2009)

Uma vez que uma única definição SBA é impossível de ser universalmente aceita, Macal (2016) propôs uma simplificação de definição de SBA a partir do nível de complexidade dos agentes, de acordo com o aumento de sua complexidade, como mostrado na Tabela 3.

Quadro 2- Definições das propriedades do agente

Propriedades do agente	**Individualidade**	**Comportamentos**	**Interações**	**Adaptabilidade**
Individual	Heterogêneos individuais	Prescrito roteirizado	Limitadas	Nenhuma
Autônomo	Heterogêneos individuais	Autônomo, dinâmico	Limitadas	Nenhuma
Interativo	Heterogêneos individuais	Autônomo, dinâmico	Entre outros agentes e o ambiente	Nenhuma
Adaptativo	Heterogêneos individuais	Autônomo, dinâmico	Entre outros agentes e o ambiente	Alteram seu comporta-mento durante a simulação

Fonte: adaptado de Macal (2016)

2.2.1 Vantagens e desvantagens da SBA

A principal vantagem de um sistema de multi-agentes é que os agentes não precisam depender de uma entidade específica para executar ações, assim como a falha de um componente não interromperá a operação de todo o sistema de produção (XIANG e LEE, 2008). A técnica baseada em agentes é muito poderosa porque permite capturar estruturas mais complexas e dinâmicas, e outra vantagem é que essa técnica permite a construção de modelos na ausência de conhecimentos sobre as interdependências globais (BORSCHEV e FILLIPOV, 2004).

Wakeland *et al*. (2004) destacam que uma grande vantagem da SBA é que essa ferramenta é excelente para descrever o comportamento de entidades individuais. A SBA é uma técnica que utiliza a abordagem *bottom-up*, diferentemente das técnicas de modelagem convencionais que utilizam a abordagem *top-down* (de cima para baixo), e isso ocasiona uma vantagem, pois permite que o sistema seja capaz de capturar mais corretamente as relações entre os componentes do sistema (DEVILLERS *et al*., 2010).

Garcia (2005) defende que a SBA é mais adequada para sistemas onde a unidade natural de análise é a individual (como por exemplo o consumidor, a firma, o empregado) e também quando tanto quanto o comportamento micro dos indivíduos e os padrões macro de interação desses indivíduos são de interesse. A autora também comenta outras duas vantagens da SBA: uma é que essa técnica torna mais fácil a distinção entre espaço físico e espaço temporal e outra que é por conta da sua facilidade de implementação, isso se comparado a outros modelos analíticos, mesmo que seja necessário algum conhecimento de programação.

Um agente em uma SBA pode ser utilizado para capturar com eficiência, velocidade e variabilidade a performance humana, o que é extremamente importante para a segurança e análise do sistema como um todo (LEE, RAVINDER e JOHNSTON, 2005). Os autores ainda discorrem sobre a importância da cognição humana em projetos de simulação, e que os modelos de desempenho humano têm incorporado cada vez mais as capacidades e limitações perceptivas, cognitivas e motoras dos operadores humanos.

Apesar da SBA se apresentar uma ferramenta tão poderosa e com tantas vantagens, ela também apresenta desvantagens. Para Siebers *et al*. (2010) e Brailsford (2014), uma desvantagem é que por mais que existam excelentes ferramentas desenvolvidas academicamente (como Repast® e NetLogo®), até então o único *software* comercialmente disponível era o AnyLogic®, porém, recentemente já possuem outros *softwares* disponíveis, apesar do AnyLogic® ser o mais utilizado, e fora que para todas essas ferramentas é necessário o conhecimento de programação orientada a objetos e que o modelador conheça de linguagem Java, assim a SBA acaba se tornando mais limitada a relativamente poucos desenvolvedores que possuam essas habilidades e também a pesquisadores acadêmicos.

Bobashev *et al*. (2007) afirmam que uma desvantagem é que por mais que a SBA consiga capturar interações locais, isso pode exigir uma carga computacional e paramétrica pesada, uma vez que rastrear e programar um grande número de agentes interagindo requer altos requisitos computacionais. Já Ross, Ulieru e Gorod (2014) afirmam que a SBA é uma ferramenta que demanda

muito esforço computacional se utilizada para modelar processos rotineiros e determinísticos, isso se comparada a outras técnicas.

2.2.2 Aplicações da SBA

Por conta de seu potencial e vantagens, a SBA é aplicável em diversas áreas, Macal e North (2009) comentam que ela é capaz de abranger sistemas humanos sociais, físicos e biológicos, que sua aplicação vai desde modelagem de civilizações antigas que se foram há centenas de anos, até o *design* de mercado para novos produtos que ainda não existem. Os autores ainda citam como exemplo algumas dessas áreas de aplicação, como: controle de tráfico aéreo, antropologia, pesquisa biomédica, química, análise de crimes, ecologia, análise energética, análise de mercado, na tomada de decisão organizacional e modelagem de epidemias. Serão apresentados exemplos recentes de cada uma dessas áreas de aplicação a seguir.

Esmaeilzadeh, Grenn e Roberts (2018) utilizaram a SBA testando o atraso de voo dentro de um sistema complexo de transporte aéreo, onde os atrasos são devidos às estratégias de gerenciamento de tráfego aéreo que buscam equilibrar a demanda aérea em condições não normais (como mau tempo). O trabalho concluiu que as estratégias atuais podem resultar em atrasos desnecessários e em um número insuficiente de voos em período de demandas bem abaixo da capacidade, que estratégias agressivas podem ter um efeito reverso e que estratégias de longa duração possuem um impacto mais significativo em atrasos de condições climáticas seriamente adversas.

O trabalho de Souza, Mateos e Madella (2020) utilizou a SBA para explorar a expansão de quatro culturas arqueológicas nas terras baixas da América do Sul, principalmente originarias da bacia amazônica e de seus arredores no período Holoceno. Os autores concluíram que embora processos demográficos possam ser modelados de forma mais realista, outros já não são, possivelmente por conta dos diferentes processos que conduzem sua dispersão (como a difusão cultural) ou dados arqueológicos problemáticos ou incompletos.

Ozik *et al*. (2019) construíram um modelo baseado em agentes de imunovigilância contra tumores heterogêneos, incluindo a dinâmica espacial das interações de contato estocástico tumor-imunológico. A partir de aprendizagem ativa e algoritmos genéricos, os autores descobriram iterativamente regiões de regressão de câncer ideais dentro de restrições biológicas e clínicas.

Hutty, Dong e Brown (2020) criaram um modelo de SBA para avaliar o desempenho de adequação de armazenamento de energia utilizando células reversíveis de óxido sólido e/ou em bateria de uma microrrede (composta por residências equipadas com geração solar fotovoltaica), permitindo assim que a autossuficiência dessa microrrede seja quantificada por meio do modelo,

para analisar uma possível redução de custos. Os autores utilizaram a Inglaterra e o Texas como objeto de estudo, e concluíram que atualmente sistemas puros de células reversíveis ainda são inviáveis, mas caso reduza o seu preço, como a literatura sugere, um sistema com 50% de independência alcançaria o retorno em 20 anos. Concluíram também que o tempo de retorno no Texas é inferior ao da Inglaterra, por conta da alta incidência solar e também que o armazenamento híbrido (bateria e células reversíveis) é considerado preferível aos sistemas puros de bateria quando é necessária uma alta taxa de autossuficiência e quando uma grande sobrecarga de geração fotovoltaica não é possível.

O trabalho de Nardin, Székely e Andrighetto (2017) focou em descrever os princípios do *design*, comentar os recursos e limitações e também sobre as possíveis aplicações do simulador GLODERS-S. Esse simulador é configurável a partir de agentes, capaz de reproduzir dinâmicas de organizações criminosas, sendo uma plataforma altamente flexível permitindo a análise de diversos cenários, tornando-se uma ferramenta útil na análise das fraquezas do crime organizado.

Armstrong *et al.* (2021) desenvolveram uma estrutura conceitual alternativa ao objetivo comum de planejamento de adaptação biológica, que é identificar locais que permaneçam mais frescos durante o verão, o regime de crescimento, considerando a criação sazonal e da paisagem no desempenho fisiológico, focado em peixes ribeirinhos, utilizando modelo de temperatura para quatorze bacias hidrográficas. O trabalho demonstrou a partir de uma SBA que os habitats mais quentes as jusantes da meia estação podem alimentar a produção anual de peixes, e também revelou uma sinergia entre habitats frios e quentes que podem ser fundamentais para apoiar a pesca em água fria.

Cai *et al*. (2017) propuseram uma visão de sistema de distribuição de energia onde os prossumidores (produtores que também são consumidores) são incentivados a equilibrar sua eletricidade em uma comunidade local. A pesquisa utilizou a SBA com uma representação em quatro camadas para estudar as características dessa comunidade e as melhores estratégias de incentivo para o desempenho desejado.

O trabalho de Ruhang e Zhilin (2017) buscou identificar a transformação das expectativas sob diferentes condições exógenas em um mercado imobiliário utilizando a SBA. Os autores construíram o banco de ensaio do mercado imobiliário baseado em agentes e que esse se mostrou eficiente, permitindo simular situações semelhantes à realidade e auxiliando nas análises de mercado para um determinado caso de aplicação.

Rodrigues, Pinho e Sena (2020) utilizaram um sistema híbrido (SED + SBA) para estudar programas de produção *job shop* flexíveis, isto é, sistemas que possuem flexibilidade de *mix* e de

volume. O trabalho buscou comparar três métodos de programação, que é com a sequência de chegada, outra com o agente utilizando uma lógica de sequenciamento e um terceiro método que também é o agente usando a lógica, porém com ajustes na sequência durante a produção do lote, enfatizando que o agente gerente reduza o *makespan* e aumente a utilização das máquinas a medida que aumenta sua interferência no modelo.

Por conta do cenário atual da pandemia do novo coronavírus, o COVID-19, a SBA tem se mostrado uma ferramenta muito poderosa na área de modelagem de epidemias, e muitos trabalhos estão sendo realizados nessa área para auxiliar no combate ao novo coronavírus. A seguir serão apresentados alguns desses trabalhos.

Como o trabalho de Silva *et al*. (2020) em que os autores criaram um modelo de SBA de suscetibilidade, exposição, infeção e recuperação (SEIR - *Susceptible-Exposed-Infected-Recovered*), o COVID-ABS, simulando agentes de pessoas, negócios e governos, propondo sete cenários diferentes: (1) nada é feito; (2) *lockdown*; (3) *lockdown* parcial; (4) isolamento vertical; (5) isolamento parcial; (6) uso de máscara facial e (7) uso de máscaras junto a 50% de adesão do isolamento social. O resultado dessa simulação mostrou que os cenários 2 e 3 apresentaram os melhores resultados epidemiológicos, seguido do cenário 7, e que os cenários 1 e 4 apresentam os maiores números de mortes, assim os autores sugerem que em caso não seja possível a realização do *lockdown* ou do *lockdown* parcial, a realização do uso de máscaras junto a 50% do isolamento social como a melhor possibilidade. O COVID-ABS é um software aberto e pode ser expansível e alterado, se adaptando a novos cenários de acordo com as necessidades.

Nishi *et al*. (2020) também utilizaram um modelo SBA de SEIR com o objetivo de investigar alternativas para reduzir a transmissão do coronavírus sem precisar interromper as atividades econômicas. A simulação decorreu com as pessoas envolvendo atividades em grupo de diversos setores (como ir ao trabalho, supermercado), dividindo em duas estratégias, a primeira seria dividir cada grupo em dois subgrupos que dividiriam suas atividades (como um grupo iria ao supermercado pela manhã, enquanto o outro iria à tarde) e a segunda estratégia foi equilibrar o número de membro de cada grupo (cada supermercado com o mesmo número de clientes), mostrando que a estratégia de divisão de grupos é capaz de reduzir bastante a transmissão e que se unidas as duas estratégias, se torna mais eficiente ainda.

Kim e Koo (2021) objetivaram em seu trabalho analisar a eficácia da testagem e rastreamento de contato de pessoas através da SBA com um modelo de suscetibilidade, exposição, infeção, confirmação e recuperação (SEICR - *Susceptible-Exposed-Infectious-Confirmed-Recovered*). Os autores comprovaram que utilizando o teste em conjunto com o rastreamento de

contato é capaz reduzir o pico de infecções significativamente, reduzindo assim uma sobrecarga de leitos nos hospitais.

Abdollahi *et al*. (2020) desenvolveram um modelo SBA para avaliar a eficácia na redução do número de casos com o fechamento das escolas, com cenários com a implementação do auto isolamento em casos sintomáticos, para isso estratificaram em grupos etários e suas taxas de contato dentro de cada grupo e entre os grupos. O trabalho mostrou que o impacto do fechamento das escolas pode ser limitado sem medidas que interrompam a cadeia de transmissão do vírus.

Nesta seção foi apresentado como a SBA é uma ferramenta, que apesar de ser mais recente se comparada a outras técnicas de simulação, ela está sendo cada vez mais utilizada, ela utiliza agentes em sua modelagem, que são indivíduos com características individuais, fazendo com que essa ferramenta se torne poderosa, como por exemplo, na representação humana, mas que possui desvantagens como o fato de necessitar um conhecimento específico de programação orientada a objetos, e para mostrar sua aplicabilidade, foram mostrados alguns trabalhos recentes utilizando essa ferramenta em diversas áreas diferentes. Como o intuito do trabalho é discorrer sobre a SBA em conjunto com *Machine Learning* (ML), a próxima seção trará o embasamento teórico sobre ML.

2.3 *Machine Learning* (ML)

Inicialmente é importante destacar um conceito mais amplo, que é o de Indústria 4.0 (I4.0), que segundo os autores Perales, Valero e García (2018) ainda não há uma definição clara sobre o termo, que mesmo os fundadores da ideia apenas descrevem a visão e as tecnologias básicas que a ideia visa. Hermann, Pentek e Otto (2016) afirmam que o termo “Indústria 4.0” é utilizado para a nova revolução industrial que está começando e que este termo se tornou público a partir de 2011.

Alcácer e Cruz-Machado (2019) citam onze itens necessários na Indústria 4.0, nove que são considerados as principais tecnologias da I4.0: a internet das coisas industriais, a computação em nuvem, a grande quantidade de dados (*Big Data*), a simulação, a realidade aumentada, a manufatura aditiva, a integração vertical e horizontal do sistema, os robôs autônomos e a *cyber*-segurança; e dois itens são a fábrica inteligente da I4.0: os sistemas *cyber*-físicos e a internet de serviços. A Figura 2 sumariza esses itens conectados a I4.0.

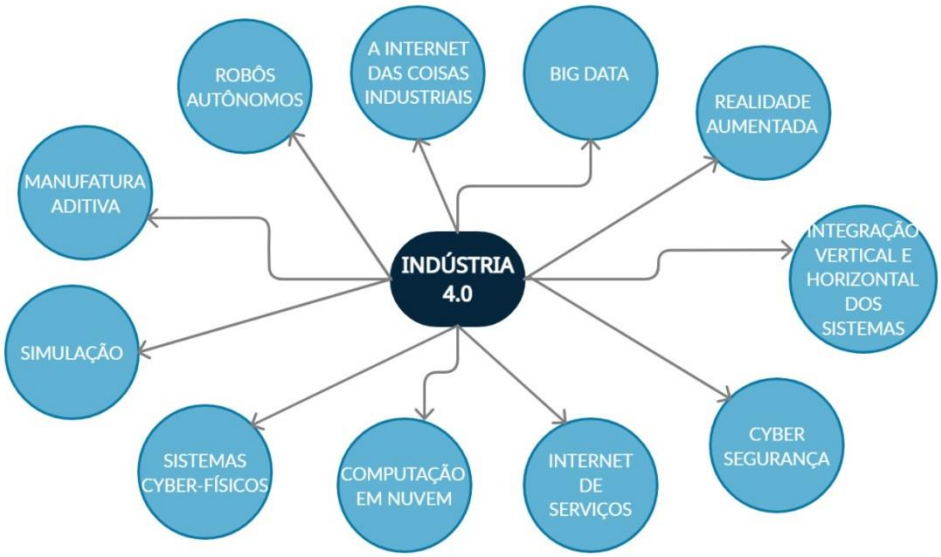


Figura 2 - Principais itens da I4.0
Fonte: adaptado de Alcácer e Cruz-Machado (2019)

Dois desses itens merecem destaque nesse trabalho, que é a simulação e o *Big Data*, sendo que o primeiro já foi discutido anteriormente e o *Big Data*, segundo Alcácer e Cruz-Machado (2019), é uma grande quantidade de dados que podem ser estruturados, semi-estruturados ou não-estruturados.

Brown, Chui e Manyika (2011) afirmam que o *Big Data* pode alterar a concorrência, transformando processos, alterando ecossistemas cooperativos e facilitando a inovação. Mas os autores Babiceanu e Seker (2016) completam que o fato de colecionar, armazenar ou utilizar essa grande quantidade de dados (*Big Data*) sem uma análise dos dados (*Data Anaylis*), é algo sem muito valor.

A ciência dos dados (*Data Science*) tende a ser sinônima de *Big Data*, mas vale ressaltar a diferença entre os termos, afinal, a *Data Science* é a aplicação de soluções encontradas através de pesquisa matemática e computacional, enquanto o *Big Data* descreve problemas, focando na análise de dados com relação à volume, variação e velocidade (3V) (CONCOLATO e CHEN, 2017). Song e Zhu (2018) afirmam que a nossa sociedade está passando por uma transformação digital devido à recente explosão do *Big Data*, entrando em novo mundo com inúmeros desenvolvimentos significativos, e que no centro dessa transformação digital está a ciência dos dados, que é a disciplina que dá sentido ao *Big Data*.

O *Big Data* pode ser caracterizado em cinco dimensões: volume (quantidade/número de dados), velocidade (velocidade da geração de dados), variedade (tipo, natureza e formato dos dados), veracidade (confiabilidade/qualidade dos dados capturados) e valor (discernimento e impacto) (ZHOU *et al*., 2017). Para melhor explicar essas cinco dimensões, os mesmos autores

elaboraram uma classificação por camadas, representada pela Figura 3, onde a camada "*Big*" é a camada fundamental, a "*Data*" é o centro do *Big Data* e a camada "Valor" caracteriza o impacto do *Big Data* no mundo real e suas aplicações, e também é possível identificar que quanto mais abaixo nas camadas (como volume e velocidade), mais depende dos avanços tecnológicos e quanto mais acima (como a camada "Valor") é mais orientada a aplicações voltadas ao poder estratégico do *Big Data*.

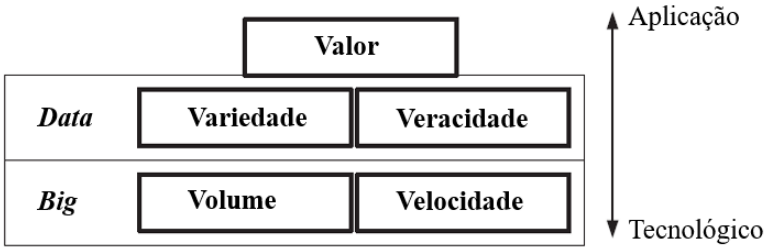


Figura 3 - Representação do *Big Data* por camadas.
Fonte: adaptado de Zhou *et al.* (2017)

Concolato e Chen (2017) afirmam que a *Data Science* possui uma relação muito próxima a mineração de dados (*Data Mining*), inteligência artificial (IA), e estatísticas no sentido de implementação dessas técnicas fundamentais. Para explorar os dados, são necessários uma análise de dados avançada, utilizando a computação em nuvem através de análises, métodos e ferramentas avançadas, *off-line* e em tempo real os dados são analisados e refinados, como por exemplo com aprendizado por máquina (*Machine Learning*), modelos de previsão e outros (ALCÁCER e CRUZ-MACHADO, 2019).

Segundo Zhou *et al*. (2017) a era do *Big Data* provê bastante interesse no *Machine Learning* (ML), pois o *Big Data* oferece muitas informações ricas sem precedentes para os algoritmos de ML, que podem extrair padrões subjacentes e construir modelos preditivos. Os autores ainda completam fornecendo uma estrutura do papel do ML no *Big Data*, mostrado pela Figura 4, onde a estrutura é centrada no ML interagindo com outros quatro componentes em ambas as direções, sendo eles:

- Big Data: serve as entradas para o ML e por fim, gera as saídas;
- Usuário: interage com o ML provendo o conhecimento do domínio, preferências pessoais e *feedback* da usabilidade;
- Domínio: serve tanto como fonte de conhecimento para orientar o ML, como também aplicando os modelos aprendidos;
- Sistema: como os algoritmos de aprendizados devem ser executados e quão eficientes é executá-los.

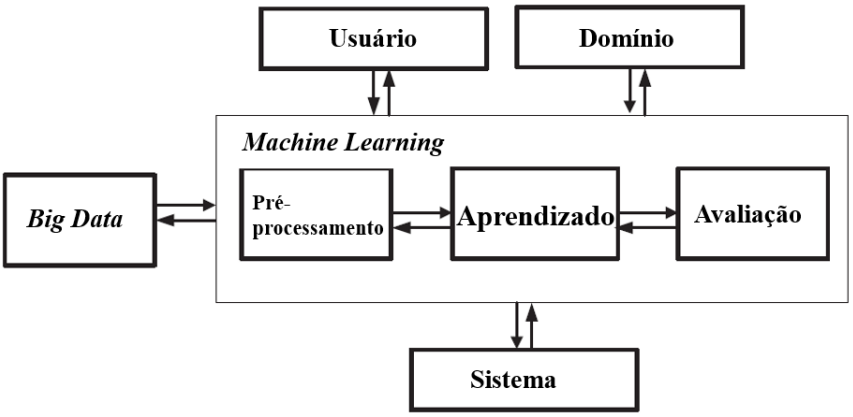


Figura 4 - Estrutura do ML no Big Data
Fonte: adaptado de Zhou *et al*. (2017)

Mohri, Rostamizadeh e Talwalkar (2018) definem amplamente *Machine Learning* como sendo métodos computacionais que utilizam experiências (informações passadas disponíveis para o sistema, geralmente em forma de dados eletrônicos) para melhorar o desempenho ou fazer previsões precisas. Os autores ainda discorrem que o ML consiste em projetar algoritmos de predição eficientes e precisos, que as medidas críticas da qualidade são de acordo com sua complexidade de tempo e espaço.

Para Vermeulen (2020) o ML é a capacidade de um sistema aprender sem fornecer explicitamente as regras, e o desenvolvimento do software pode ou não fazer parte do ML, dependendo de qual ferramenta/técnica que está sendo usada. Enquanto os autores Wuest *et al*. (2016) citam que o fato desse campo ser muito diverso, com diversos algoritmos, teorias e métodos disponíveis, isso pode ser barreira para os profissionais de manufatura em relação à adoção dessas ferramentas.

Zhou *et al*. (2017) caracterizaram o ML em três dimensões, como mostrado na Figura 5:

1. Objetivo do aprendizado:
   - Representação: objetiva aprender novas representações dos dados que tornam mais fácil extrair informações úteis ao construir classificadores ou outros preditores;
   - Tarefa: normalmente possui resultados definidos, podendo ser categorizado em classificação, regressão e agrupamento.
2. Tempo de disponibilidade dos dados:
   - Aprendizado em *Batch*: gera modelos baseado no treinamento de todos os dados;
   - Aprendizado *Online*: atualiza os modelos baseado em cada novo dado de entrada (*input*).

3. Natureza do *feedback* de aprendizagem:
    - Supervisionado: representado por exemplos de pares de dados entrada-saída (*input-output*), objetivando aprender a função que orienta os dados de entradas aos dados de saídas;
    - Não supervisionado: o sistema não é fornecido com o *feedback* explicito ou saída desejada, o objetivo é descobrir padrões nos dados de entrada;
    - Por reforço: não é representado por pares de dados entrada-saída, o *feedback* é dado a partir de experiências anteriores, sendo esse *feedback* caracterizado por recompensas ou punições associadas às ações ao invés da saída desejada ou correção explícita de ações subótimas.

Os autores Zhou *et al*. (2017) ainda completam que há um quarto tipo de *feedback* de aprendizagem o semi-estruturado, que seria entre o supervisionado e o não supervisionado, onde o sistema é representado tanto com um baixo número de parede dados entrada-saída e também um grande número de entradas não anotadas, o objetivo desse aprendizado é semelhante ao supervisionado, a diferença é que ele aprende tanto a partir de dados anotados como dados não anotados. Mas esse trabalho focará apenas no aprendizado por reforço (*Reinforcement Learning* – RL), uma vez que esse é o método empregado no decorrer da pesquisa.

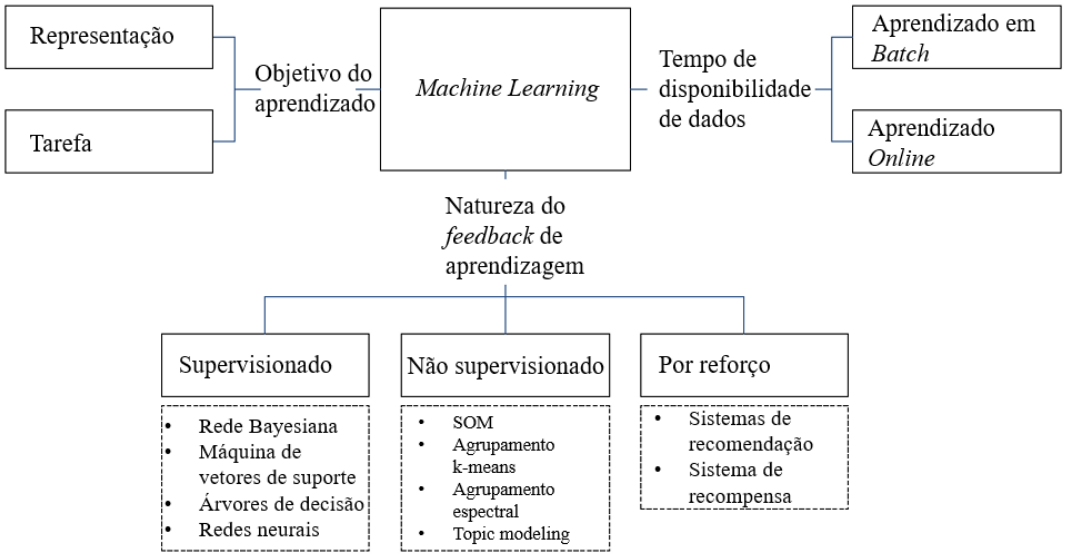


Figura 5 - Taxonomia das dimensões do ML
Fonte: adaptado de Zhou *et al*. (2017)

2.3.1 Aprendizado por Reforço

Sutton e Barto (2018) definem que os problemas de RL envolvem um aprendizado que busca maximizar um numérico de recompensa, são chamados de problemas *closed-loop* (sistema fechado), onde as ações dos sistemas de aprendizado influenciam em seus *inputs* (dados de entrada) posteriores. Os autores ainda completam que esse tipo de problema é basicamente capturar os

aspectos mais importantes do problema a partir de um agente interagindo com o ambiente para concluir um objetivo.

Tizhoosh (2016) comentou sobre uma analogia do RL com um cachorro, onde o animal aprende a partir de recompensas para um determinado objetivo, o mesmo ocorre no RL, onde os agentes aprendem a partir da acumulação de conhecimento (sinais de reforço) para dadas ações produzindo as maiores recompensas. Esse mesmo autor também comenta que os agentes no RL conseguem explorar de forma autônoma ambientes altamente dinâmicos e estocásticos, e de desenvolver ótimas políticas de controle, acumulando um *feedback* avaliativo do ambiente, mas um problema de algoritmos de RL é que necessita um tempo longo para explorar um ambiente desconhecido.

O *Reinforcement Learning* é um problema onde o agente deve aprender a partir de interações de tentativa e erro com um ambiente dinâmico (KAELBLING, LITTMAN E MOORE, 1996). Os autores destacam que há duas principais estratégias para resolver esse tipo de problema, que a primeira é pesquisar no espaço de comportamentos para encontrar um que se adeque bem ao ambiente e que a segunda é utilizar técnicas estatísticas e métodos de programação dinâmica para estimar a utilidade das tomadas de ação nos estados do mundo.

Sutton e Barto (2018) destacam quatro outros elementos presentes no RL, além do agente e do ambiente, que segundo eles são a política, o sinal de recompensa, a função de valor e opcionalmente o modelo do ambiente:

- Política: define o sentido do comportamento do aprendizado do agente em um determinado período de tempo, um mapeamento dos estados percebidos do ambiente para determinar as ações quando estiverem nesses estados;
- Sinal de recompensa: define o objetivo em um problema de RL, onde o ambiente envia um ao agente do RL um sinal numérico, uma recompensa, onde o objetivo é maximizar o total de recompensas adquiridas;
- Função de valor: enquanto o sinal de recompensa indica em um sentido imediato o que é bom, a função de valor especifica ao decorrer do processo, sendo o total de recompensas que o agente pode esperar acumular em um até um determinado ponto no futuro, partindo de um tal estado;
- Modelo: é algo capaz de imitar o comportamento do ambiente, permitindo que inferências sejam feitas sobre como o ambiente deve se comportar.

Os mesmos autores Sutton e Barto (2018) ainda discorreram a respeito da interface da interação agente-ambiente, representado na Figura 6, onde o agente aprende e é um tomador de

decisão, onde ele interage com o ambiente para alcançar um objetivo. Nessa interação, o agente e o ambiente interagem em uma sequência discreta de tempos ($t = 0, 1, 2, 3, ...$), em cada passo t o agente recebe uma nova representação do estado $S_t \in S$, onde S é um conjunto de possíveis estados, e com base nisso, seleciona uma ação $A_t \in A(S_t)$, onde $A(S_t)$ é um conjunto de possíveis ações no estado S_t, após isso, no próximo passo o agente recebe uma recompensa numérica, R_{t+1}, e assim se encontra em novo estado, S_{t+1}.

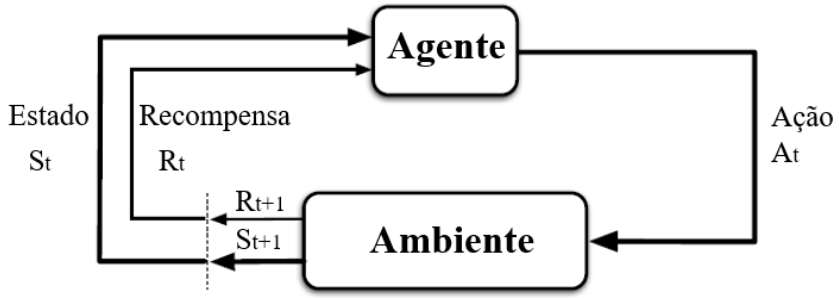


Figura 6 - Interação agente-ambiente em um RL
Fonte: adaptado de Sutton e Barto (2018)

Há dois modos de proceder no RL, o *model-free* (modelo livre) que é onde um controlador aprende sem aprender o modelo e o *model-based* (baseado em modelo) que é quando aprende o modelo e utiliza disso para direcionar o controlador (KAELBLING, LITTMAN e MOORE, 1996). Vermeulen (2020) destaca alguns dos algoritmos *model-free* e suas características, utilizados no RL atualmente no mundo, sumarizados na Tabela 4. Segundo o autor, são: Monte Carlo, *Q-learning*, SARSA, *Q-learning Lambda*, DQN (*Deep Q-Network*), DDPG (*Deep Deterministic Policy Gradient*), A3C (*Asynchronus Actor-Critic Algorithm*), NAF (*Q-learning with Normalized Advantages Functions*), TRPO (*Trust Region Policy Optimization*) e PPO (*Proximal Policy Optimization*). Importante ressaltar que posteriormente serão detalhados apenas dois desses algoritmos, o *Q-learning* e o PPO, uma vez que esses serão utilizados nessa pesquisa.

Quadro 3- Algoritmos utilizados em *model-free* no RL

Algoritmo	**Política**	**Espaço das ações**	**Espaço dos estados**	**Operador**
Monte Carlo	Não possui	Discreto	Discreto	*Sample-means*
Q-learning	Não possui	Discreto	Discreto	Q-valor
SARSA	Possui	Discreto	Discreto	Q-valor
Q-learning-Lambda	Não possui	Discreto	Discreto	Q-valor
SARSA-Lambda	Possui	Discreto	Discreto	Q-valor
DQN	Não possui	Discreto	Contínuo	Q-valor
DDPG	Não possui	Contínuo	Contínuo	Q-valor
A3C	Não possui	Contínuo	Contínuo	Q-valor

NAF	Não possui	Contínuo	Contínuo	*Advantage*
TRPO	Possui	Contínuo	Contínuo	*Advantage*
PPO	Possui	Contínuo	Contínuo	*Advantage*

Fonte: adaptado de Vermeulen (2020).

O algoritmo *Q-learning* foi apresentado por Watkins (1989), onde são atualizados pares estado-ação, buscando maximizar o valor da recompensa após o seu desconto, sendo representado em sua forma mais simples de passo único pela Equação 1, os autores Sutton e Barto (2018) apresentam o algoritmo base sem política implementada, como de acordo com o Quadro 1:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R_{t+1} + \gamma max_a\left(s_{t+1,a}\right) - Q(s,a)\right) \quad (1)$$

Sendo:

- a: ação a ser tomada;
- s: estado do sistema;
- α: taxa de aprendizado $(0 < \alpha < 1)$;
- R_{t+1}: valor da recompensa ou punição;
- γ: coeficiente de desconto da recompensa futura $(0 < \gamma < 1)$.

Quadro 4 - Algoritmo base do *Q-learning* sem política

Inicialize $Q(s,a)$ arbitrariamente, e $Q(estado\ final\ \cdot) = 0$
Repita (para cada episódio)
Inicialize s
Repita (para cada passo do episódio):
Escolha a de s usando uma política derivada de Q (por exemplo, ε-greedy)
Escolha uma ação a, observe R, s'
$Q(s,a) \leftarrow Q(s,a) + \alpha\left(R + \gamma max_a Q(s',a) - Q(s,a)\right)$
$s \leftarrow s'$;
Até s ser um estado final.

Fonte: adaptado de Sutton e Barto (2018).

O algoritmo PPO - *Proximal Policy Optimization* (Otimização de Política Proximal) é um método de política de gradientes para RL, onde é realizado uma atualização do gradiente por amostra de dados, a função objetivo permite vários períodos em atualizações de *minibatch*, um método que possui as mesmas vantagens do TRPO, porém muito mais simples de ser implementado, é mais geral e possui uma melhor complexidade de amostras (empiricamente) (SCHULMAN *et al.*, 2017). Os autores também descrevem um algoritmo base, mostrado no Quadro 2, onde a cada iteração cada um dos N *actors* (atores) paralelos coleta T etapas de tempos de dados, e o *surrogate* (substituto) de perda em cada etapa de dados NT e é otimizado com o *minibatch* SGD (*Stochastic Gradient Descent*) para K períodos.

Quadro 5 - Algoritmo base do PPO

Para iteração = 1, 2, ... faça
  Para *actor* = 1, 2, ... N faça
    Rode política $\pi_{\theta_{antigo}}$ no ambiente para T passos
    Compute estimativas de vantagem $\hat{A}_1, \ldots, \hat{A}_T$
  Finalize para
  Otimize *surrogate* L escrevendo θ, com K períodos e *minibatch* tamanho $M \leq NT$
  $\theta_{antigo} \leftarrow \theta$
Finalize para

Fonte: adaptado de Schulman *et al*. (2017).

Sendo π_θ a política estocástica e $\hat{A}_t$ o estimador da função *advantage* no tempo de dado t.

Assim foram apresentados nessa seção os itens presentes na Indústria 4.0, entre eles estão a Simulação e o *Big Data*, onde o *Big Data* é uma grande quantidade de dados que necessitam de interpretação para extrair as informações necessárias, e para realizar isso pode-se usar o *Machine Learning*. Uma das formas de utilização do ML é a partir do aprendizado por reforço, que funciona em um sistema fechado utilizando um sistema de recompensas pelos erros ou acertos, e que para implementá-lo é necessária a utilização de algoritmos específicos, e, foram detalhados dois deles, o *Q-learning* e o PPO. Na seção seguinte será discutido a respeito da ferramenta externa que será utilizada para implementação do ML em um modelo de SBA, o *Pathmind*.

2.4 Pathmind

Inicialmente será discutido a respeito de como é o funcionamento geral da ferramenta Pathmind, bem como também suas aplicações e vantagens.

As informações a respeito da ferramenta serão obtidas a partir do próprio desenvolvedor Pathmind (2021), uma vez que ela é muito recente. Essa ferramenta é uma plataforma SaaS (*Software as a Service – Software* como serviço), no qual é um modelo baseado na nuvem, possuindo um sistema alocado em uma nuvem sendo acessada via internet.

Essa ferramenta permite a implementação de um algoritmo de aprendizado de máquina utilizando o aprendizado por reforço para cenários do mundo real sem a necessidade de ser um especialista em ciência dos dados, permitindo que o foco do modelador seja na simulação em si. Ela também é útil na aplicação de gêmeos digitais.

É possível visualizar seu funcionamento na Figura 7, que é através de uma política. O início é como em qualquer simulação, no qual é importada as informações relevantes do mundo real para o sistema simulado, permitindo também o uso para gêmeos digitais, então é aplicado o Pathmind com uma política treinada, e esse resultado retorna ao sistema real ou para a simulação.

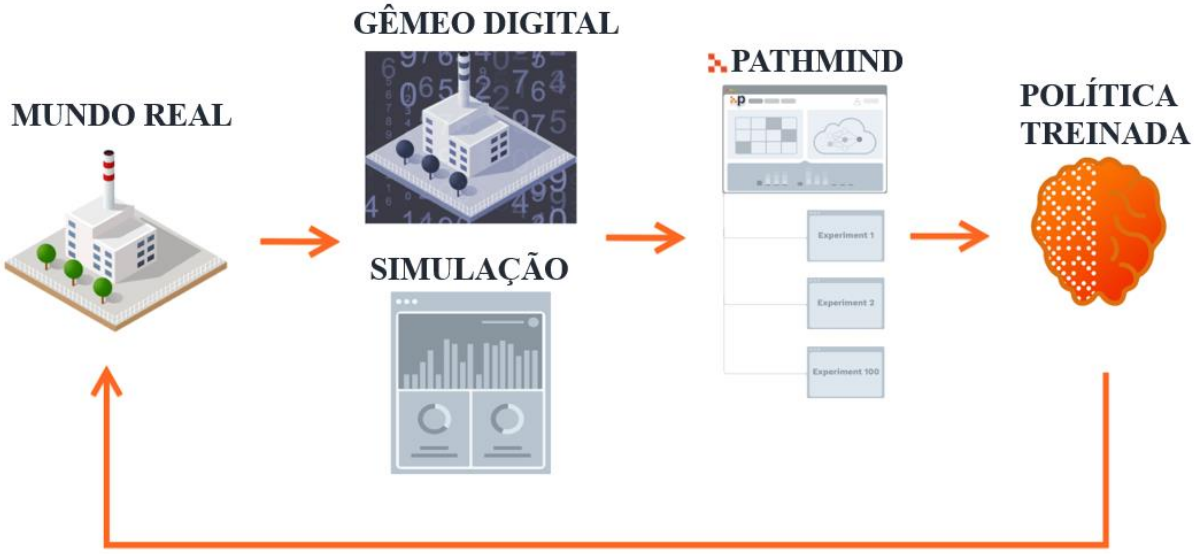


Figura 7 - Funcionamento do Pathmind
Fonte: adaptado de Pathmind (2021)

O Pathmind possui algumas vantagens, como:

- Não é necessário que seja um cientista de dados ou especialista em ML;
- Realiza o treinamento automaticamente, fazendo o ajuste de hiperparâmetros e configuração da infraestrutura, utilizando o algoritmo PPO;
- Realiza o treinamento em uma nuvem, evitando problemas com quedas de energia ou similares;
- Implementação do ML a partir de políticas treinadas em sistemas do mundo real.

A fim de mostrar sua aplicabilidade, o Pathmind (2021) disponibiliza alguns modelos construídos no AnyLogic®. As aplicações vão desde exemplos mais simples, como um modelo introdutório e um mostrando como um rato alcança o queijo, até exemplos mais complexos. Alguns exemplos mostrados são: sobre *call centers* interconectadas, uma cafeteria, uma otimização de uma cadeia de suprimentos, um veículo guiado automaticamente e até sobre pouso lunar.

Para configurar o Pathmind, é necessário configurar três parâmetros:

- Observações - as informações utilizadas pelo agente para auxiliar na tomada de decisões;
- Ações - determinam quais as ações que o agente pode tomar;
- Métricas - onde são definidas as recompensas, os indicadores de performance.

Essa pesquisa envolve o uso do ML integrado ao SBA. Então, após ser mostrado a respeito de SBA, de ML e também sobre o Pathmind, que se mostra ser uma ferramenta eficiente e permite a integração do ML a um modelo de SBA sem a necessidade que o modelador seja um especialista

em ciência dos dados ou ML. Então na seção seguinte serão apresentados alguns trabalhos recentes publicados que envolvem a união entre SBA e ML, inclusive, alguns utilizando o Pathmind.

2.5 Integração do ML à SBA

Bose *et al*. (2021) realizaram uma avaliação sobre Mercados Locais de Energia (MLE), integrando fontes de energia renováveis nos sistemas de energia de comunidades locais, isso utilizando algoritmos de aprendizado em um sistema de SBA, simulando um MLE com RL. A simulação da pesquisa envolve um sistema multi-agentes com 100 residências incluindo energia fotovoltaica, micro-cogeração e aparelhos de deslocamento de demanda. O trabalho mostrou que é possível alcançar uma autossuficiência de até 30% com negociação e de até 41,1% com negociação e resposta de demanda através de uma instalação de painéis fotovoltaicos de apenas 5kWp em 45% das residências com preços de energia acessíveis.

O trabalho de Farhan, Göhre e Junprung (2020) destacaram que há uma importância do RL na indústria da simulação, onde os autores apresentam sobre uma aplicação do RL no *software* AnyLogic® utilizando o Pathmind, aplicando para treinar um barista em uma cafeteria para que ele consiga melhorar sua eficiência e reduzir o tempo de atendimento ao cliente. A simulação com o RL se mostrou eficiente, pois gerou uma redução do tempo de atendimento a cada cliente e consequentemente sendo capaz de atender a mais clientes, mas o foco principal do trabalho era outro e também foi alcançado, que era mostrar a eficiência da ferramenta Pathmind, se mostrando útil para aplicar o RL em qualquer tipo de modelo no AnyLogic®, desde que esse modelo siga os princípios do processo de decisão de Markov.

Irannezhad, Prato e Hickman (2020) apresentaram um protótipo de um sistema inteligente de apoio a tomada de decisão sobre logística portuária utilizando um modelo multi-agentes com algoritmo de aprendizagem, buscando responder como esse sistema inteligente poderia auxiliar na tomada de decisão e como poderiam então gerar uma melhor estratégia cooperativa para conseguir obter maior agilidade para conseguir atender a demanda e fornecimento nesse sistema de carga e transporte de containers. Os autores obtiveram resultados que mostraram uma grande redução nos custos totais de transporte e na distância percorrida e também maior utilização dos caminhões com o compartilhamento de recursos.

O trabalho de Darville e Celik (2020) destacou a importância da necessidade da produção e distribuição de energia de forma eficiente, nesse contexto, os autores propõem a criação de um modelo de SBA com algoritmo de aprendizagem buscando pontos de demanda a serem considerados como prioridades em cada região. Os resultados mostraram uma redução de 80% na energia fornecida e não utilizada (supergeração).

Källström (2020) discorreram sobre como a SBA é eficaz na retratação de operadores humanos e tomadores de decisão, porém apresenta muitas dificuldades para implementação em conjunto com a IA, assim investigou sobre como o ML pode ser utilizado nesse contexto. E o trabalho mostrou que o RL multi-objetivo multi-agente é uma abordagem promissora na criação de agentes com características diversas e adaptativas, permitindo estimular humanos em treinamento.

Hassanpour *et al*. (2021) buscaram apresentar um modelo baseado em agentes de evacuação de pedestres utilizando o ML, onde os agentes do sistema podem decidir autonomamente em um modelo hierárquico de três níveis, com nós e camadas de seleção de células, sendo esse modelo testado e aprovado em um processo de evacuação de pedestres da plataforma do centro de transporte de Britomart em Auckland – Nova Zelândia durante um evento destrutivo abstrato. O trabalho teve como resultado um modelo que pode ser utilizado por *designers* e gerentes na busca de um melhor sistema de evacuação, além do modelo também poder ser utilizado em conjunto com outras ferramentas de construção.

A pesquisa de Rößler *et al*. (2020) utilizou a SBA com ML no auxílio do planejamento das circulações de trações das unidades de tração em uma rede ferroviária, buscando uma melhor robustez contra atrasos, utilizando dados baseados em previsões de um modelo de ML construído a partir de dados operacionais históricos. Os autores mostram os resultados promissores de um caso de teste em um dos maiores aglomerados de trens com tráfico nacional e internacional, onde 79 unidades de tração seriam necessárias.

Fuller, de Arruda e Ferreira Filho (2020) propuseram a aplicação do ML um modelo de SBA onde os agentes possuem autonomia para definir as regras de interação necessárias, aplicando em um sistema de redes de filas, comparando os agentes de aprendizagem com os de regras já conhecidas. Os resultados mostraram que o modelo de aprendizagem oferece políticas mais eficientes, fornecendo uma estrutura interessante para simulação.

O trabalho de Moriyama *et al*. (2019) propôs a implementação de agentes inteligentes mais complexos em uma unidade de processamento gráfico (GPU – *Graphics Processing Unit*) em conjunto com o ML, onde os agentes aprendem seu comportamento em um jogo iterativo simultâneo de duas pessoas enquanto trocam pares. Os autores compararam dois métodos, a atribuição baseada em agente e a atribuição baseada em pares, e concluíram que a baseada em pares obteve um melhor desempenho em cinco programas, porém o desempenho não foi diferente da baseada em agente quando o número de agentes era alto.

Pinciroli *et al*. (2020) utilizaram o Pathmind para contornar o fato de que explorar algoritmos de RL no AnyLogic® requer habilidades específicas e a compreensão das soluções possivelmente

contra-intuitivas propostas pelo RL não serem diretas, pois segundo os autores a ferramenta de *software* Pathmind permite explorar com eficácia os recursos de RL sem um conhecimento profundo de ML. Para provar essa eficácia, o trabalho realizou a simulação de um estudo de caso de parque eólico para identificar uma melhor política de operação e manutenção, isso utilizando o RL com o Pathmind no AnyLogic®, onde resultaram em uma política que necessita de menos interrupções para manutenção e consequentemente reduz os custos de manutenção e as perdas de produção, se comparado a políticas de agendamento e preditivo.

Augustijn *et al*. (2020) destacaram como o ML tem sido um grande aliado a SBA, e que essa integração ainda está em seu estado inicial e que enfrenta muitos desafios, assim os autores apresentaram uma comparação entre duas versões do mesmo modelo de cólera, sendo que em um desses modelos o comportamento do agente era conduzido diretamente por um algoritmo de aprendizagem e em seguida os autores aplicaram um algoritmo de aprendizagem diretamente nos dados e implementaram os resultados como regras de comportamento no modelo. O trabalho mostrou que, quando se busca criar agentes orientados a dados, derivar as regras oriundas dos dados se torna uma boa alternativa, mas destacaram que nessa estratégia, o conjunto de dados se torna o elemento chave.

O trabalho de Sena *et al*. (2017) utilizou o ML em conjunto com a simulação híbrida (SED com SBA) propondo uma política de compras de medicamentos em uma farmácia hospitalar, buscando reduzir a quantidade de medicamentos não atendidos e expirados. Os autores conseguiram reduzir em 100% o não atendimento de um medicamento e de 100% e de 67,89% em outros dois medicamentos.

Batata, Augusto e Xie (2018) alertaram sobre o risco de *burnout* (exaustão extrema psicológica profissional) por parte dos cuidadores e que acaba resultando em muitas internações, mas quando há uma pré-identificação auxilia muito no tratamento. Assim, os autores propuseram um modelo de SBA com ML para buscar melhores soluções, onde encontraram nessa abordagem uma melhor forma de identificação previa do *burnout*, consequentemente diminuindo o número de internações por conta disso.

Bosse (2020) abordou a otimização de fluxo do tráfego por meio do controle micro-nível auto-organizado combinando SBA com ML, sendo os veículos representados por agentes autônomos e adaptativos. Os resultados do trabalho mostraram que é possível aumentar a eficiência da mobilidade urbana em termos de distância do caminho e de tempo de viagem utilizando a SBA com a implantação de um controle de navegação de veículos de nível micro e com redirecionamento com base em sensores ambientais locais.

Olave-Rojas e Nickel (2021) afirmaram que tanto a SBA como a SED são ferramentas muito úteis no auxílio a tomada de decisão na área da saúde, principalmente em cenários imprevisíveis como acidentes, desastres naturais ou terrorismo, assim propuseram um modelo geral de serviços de emergência pré-hospitalares utilizando a simulação híbrida (SBA com SED) em conjunto com o ML. Os autores validaram o modelo utilizando dados do mundo real de um centro de coordenação de emergência do norte da Alemanha e apresentaram também uma abordagem para previsão de velocidade de viagem e uma aplicação do modelo para analisar a capacidade crítica no centro de coordenação.

Assim foi apresentada nesse Capítulo a fundamentação teórica utilizada para o desenvolvimento dessa pesquisa, onde inicialmente foi mostrado à respeito de simulação computacional, depois um aprofundamento em SBA, após isso os conceitos de ML e por fim, nessa última seção foram apresentados trabalhos recentes utilizando dessa temática utilizando a SBA em conjunto com o ML. Na sequência do trabalho descreve-se o método de pesquisa empregado, mostrando a classificação dessa pesquisa e como ela foi conduzida.

3. MÉTODO DE PESQUISA

Nesse Capítulo, inicialmente, estabelece-se como a pesquisa é cientificamente classificada, e as suas etapas.

3.1 Classificação da Pesquisa

Miguel *et al*. (2010) sugeriram que a pesquisa seja classificada quanto à sua natureza, seus objetivos, a abordagem e o método utilizado. Desta forma, a presente pesquisa pode ser classificada quanto a sua natureza como aplicada, pois segundo Appolinário (2006) esse tipo de pesquisa tem como objetivo resultado prático, podendo ser aplicada na resolução de problemas do mundo real, e essa pesquisa busca a criação de um modelo que pode ser utilizado na realização de problemas reais.

Este trabalho é classificado como uma abordagem quantitativa, pois Bertrand e Fransoo (2002) definem que essa abordagem é aquela em que o sistema busca resultados de forma mensurável em dimensões pré-definidas. Os mesmos autores Bertrand e Fransoo (2002) também distinguem a pesquisa quantitativas entre axiomáticas e empíricas, sendo que esse trabalho se enquadra em empírica, por se tratar de uma possível aplicação no mundo real. Dentro da pesquisa empírica também há distinção entre duas formas, empírica descritiva e empírica normativa, sendo essa pesquisa considerada como empírica normativa, pois essa classificação é dada a pesquisas que objetivam a criação de políticas, estratégias ou ações que sejam capazes de melhorar um sistema real. A pesquisa também se enquadra com objetivo explicativo, uma vez que busca descrever as etapas da realização do processo de implementação do ML a SBA.

A última classificação é quanto ao método utilizado, onde será utilizado o método de modelagem e simulação, que segundo Bertrand e Fransoo (2002) é quando se utiliza um ambiente simulado computacionalmente para a representação de um sistema real, sem a alteração deste. Assim, é possível visualizar na Figura 8 de forma sucinta como essa pesquisa é classificada.

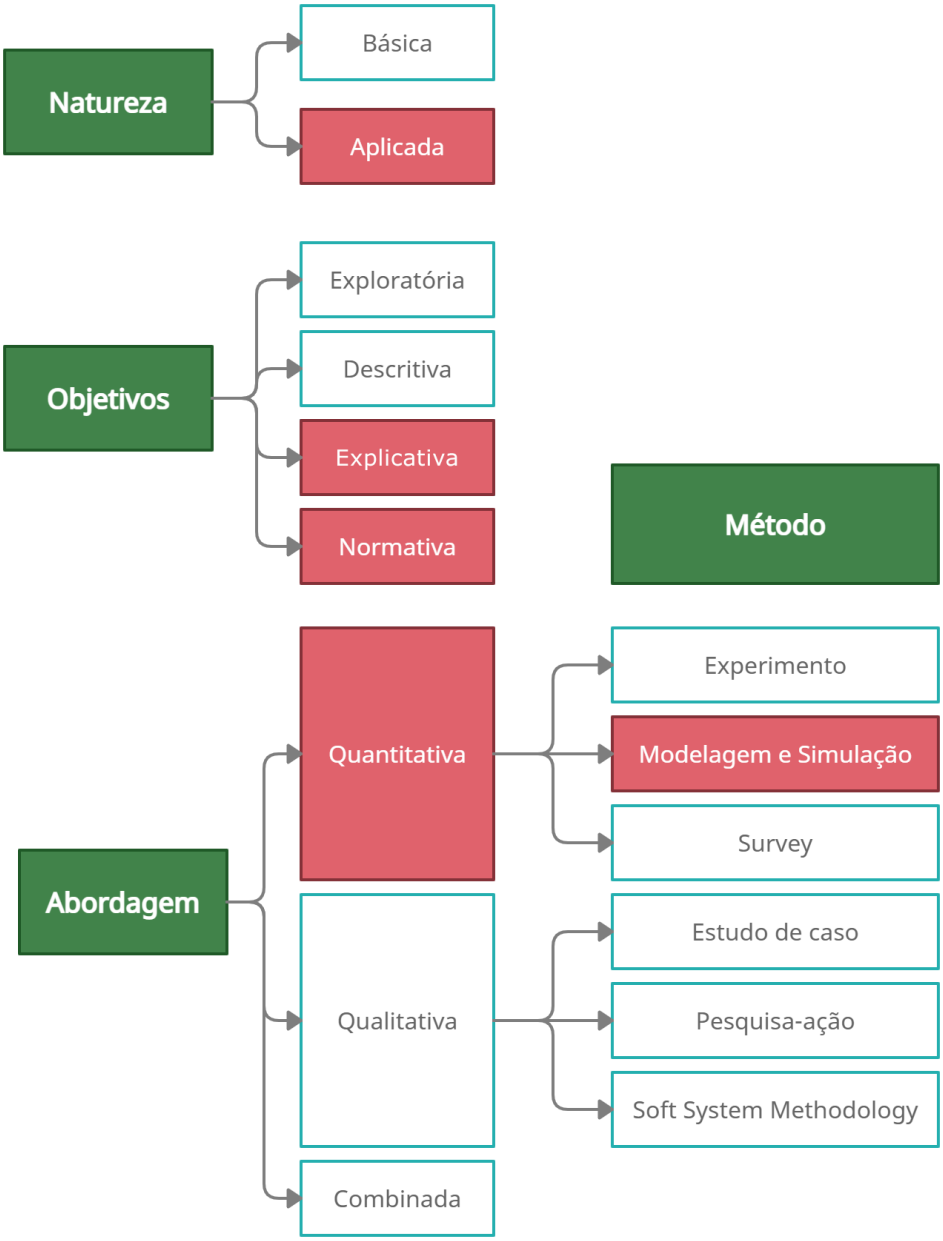


Figura 8 - Classificação da pesquisa
Fonte: adaptado de Miguel *et al*. (2010)

3.2 Condução da Pesquisa

Definido o método de pesquisa a ser utilizado, o trabalho seguiu os passos propostos por Arenales *et al*. (2007), no qual os autores definiram as etapas de um processo de modelagem utilizado na PO, sendo essa, possível de ser utilizada em processos como simulação computacional. O método é ilustrado na Figura 9 que, segundo os autores citados, segue quatro etapas:

- Formulação/modelagem: são definidas as variáveis e as relações matemáticas e abstraídas as informações relevantes para a representação do mundo real;
- Dedução e análise: etapa em que são aplicadas as técnicas matemáticas e tecnologia para resolver o modelo matemático e mostrar as soluções sugeridas;

- Interpretação/inferência: é analisado se as conclusões tiradas do modelo possuem informações suficientes para inferir conclusões ou decisões sobre o modelo real;
- Avaliação e julgamento: se as avaliações retiradas na etapa anterior não são adequadas e a modelagem necessita de revisão, o ciclo é reiniciado.

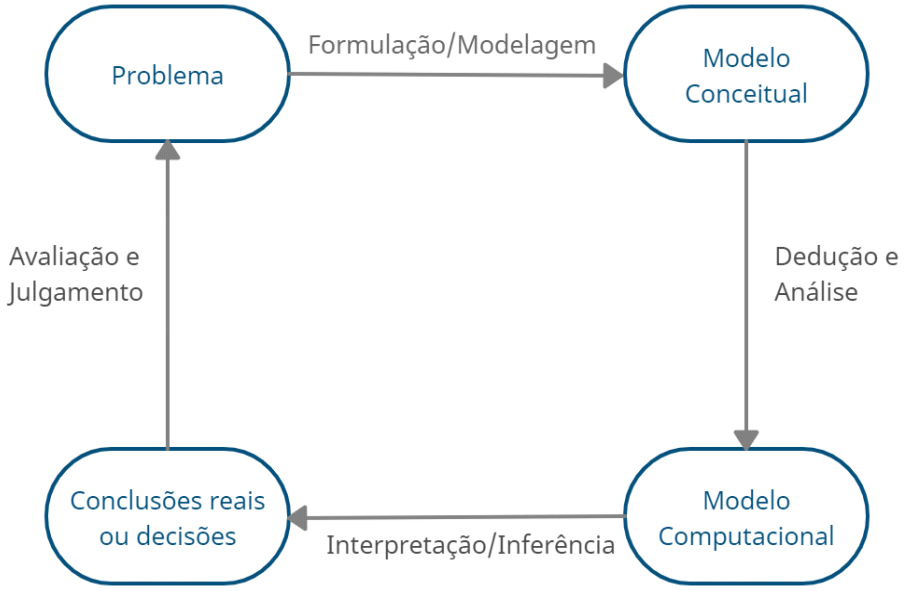


Figura 9 - Condução de pesquisa de modelagem e simulação
Fonte: adaptado de Arenales *et al.* (2007)

Sendo assim, a pesquisa será dividida em duas etapas. Inicialmente será mostrado as etapas de implementação utilizando a ferramenta externa Pathmind, onde será criado um modelo de separação de caixas de acordo com sua cor. A segunda etapa responderá a dois objetivos específicos, onde inicialmente será desenvolvido um algoritmo de aprendizagem por reforço *Q-learning* utilizando linguagem de programação Java e depois esse algoritmo será utilizado na implementação de forma direta.

Para essa pesquisa atender aos três objetivos específicos, ela seguiu os passos propostos por Arenales *et al.* (2007). Onde será mostrado inicialmente o modelo conceitual do sistema, onde será avaliado se a modelagem corresponde à realidade proposta. Ao verificar o modelo conceitual, será criado o modelo computacional utilizando o *software* AnyLogic®. Então será feito a análise se o modelo computacional está de acordo com o proposto.

Na etapa seguinte de interpretação do modelo computacional, é onde cada etapa se diferenciará, pois é quando será implementado o aprendizado de máquina, mostrando as etapas seguidas para a realização dessa implementação, no primeiro caso com a ferramenta Pathmind e no segundo com o algoritmo *Q-learning* desenvolvido. E por fim foi realizada a avaliação da eficiência do sistema.

4. DESENVOLVIMENTO DA PESQUISA

Nesse Capítulo apresenta-se os detalhes da implementação do ML em um modelo de SBA. A pesquisa mostrou que há duas formas possíveis de realizar essa implementação, que é utilizando uma ferramenta externa ou realizando de forma direta no *software* AnyLogic®. Destacando novamente que o trabalho mostrará apenas utilizando esse *software*, por se tratar de ser o único disponível para fins comerciais.

Inicialmente será mostrado utilizando a ferramenta externa Pathmind. O *software* AnyLogic® permite além do Pathmind, também o Microsoft Bonsai como ferramenta externa no auxílio da implementação do ML. A pesquisa optou pelo uso do Pathmind, pois apesar de ser uma ferramenta recente, esse já possui publicações cientificas corroborando com sua eficiência.

Finalmente descreve-se a implementação sem o uso de uma ferramenta externa, realizando o processo apenas com o uso do AnyLogic®. Para essa implementação utilizou-se a programação em Java, que é a linguagem utilizada pelo *software* AnyLogic®.

4.1 Integração utilizando ferramenta externa

O modelo utilizado para a explicação da forma de implementação do ML na SBA utilizando a ferramenta externa, será criado no AnyLogic®. O modelo é simples, uma vez que o foco será a descrição das etapas da implementação do ML, no qual o sistema gerará caixas de cores diferentes representadas por vetores, essas caixas então devem ser separadas de acordo com sua cor e por fim sairão do sistema.

Para a representação de seu modelo conceitual, será utilizado a técnica IDEF-SIM, proposta por Montevechi *et al*. (2010), no qual é explicado suas representações na Figura 10. O modelo inicia com a criação da entidade Peça, esta então é encaminhada para uma *Rack Store*, então é separada, etapa que é inserido o algoritmo de ML e assim, a entidade sai do sistema, como representado pela Figura 11.

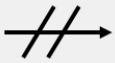
Elementos	Simbologia
Entidade	
Funções	
Fluxo de entidade	
Recursos	
Fluxo de entrada do sistema	
Fim do sistema	

Figura 10 – Representação dos elementos presentes no IDEF-SIM
Fonte: adaptado de Montevechi *et al.* (2010)

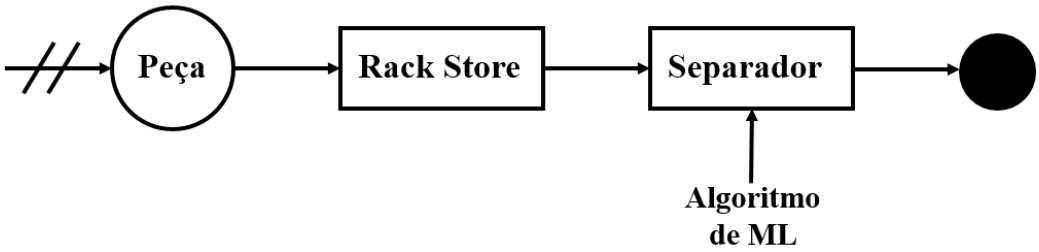


Figura 11 - Modelo conceitual do sistema
Fonte: Adaptado de Martins *et al.* (2021)

Definido o modelo conceitual, foi iniciada a construção do modelo computacional. Para isso foi necessário configurar o modelo para que sejam criadas as caixas com as três cores diferentes (vermelho, verde e azul), assim como sua representação por vetores. Então na entrada das peças foi configurado para antes da chegada como mostra o Quadro 3, onde é possível perceber o código de aleatoriedade para geração das caixas com as três cores diferentes.

Quadro 6 - Código de aleatoriedade da criação das caixas

```
Random rand = new Random();
random = rand.nextInt(3 - 1 + 1) + 1;

if(random == 1){
        pecaVermelha.caixa.setColor("MA_Box", blue);
        pecaVermelha.setColor(blue);
}else{
        if(random == 2){
                pecaVermelha.caixa.setColor("MA_Box", green);
                pecaVermelha.setColor(green);
        }else{
                pecaVermelha.caixa.setColor("MA_Box", red);
                pecaVermelha.setColor(red);
        }
}
```

Fonte: Autor

Enquanto no Quadro 4 é possível visualizar o código inserido na função criada “getCor”, no qual o agente, aqui chamado de “pecaVermelha”, é representado por vetores. A representação é feita da seguinte forma: um vetor de três dígitos, se a peça for vermelha, é colocado o valor 1 na posição zero do vetor, se for verde coloca 1 na posição um e se for azul coloca 1 na posição dois, no restante coloca 0. Para isso foi necessário a criação da variável “cores”.

Quadro 7 - Código da função getCor, a representação por vetores das cores

```
int[] states = {0, 1, 2};

int[] cores = new int[states.length];

cores[0] = (pecaVermelha.getColor() == red) ? 1 : 0;
cores[1] = (pecaVermelha.getColor() == green) ? 1 : 0;
cores[2] = (pecaVermelha.getColor() == blue) ? 1 : 0;
traceln("Cores = "+ cores[0]+" "+cores[1]+" "+cores[2]);
return cores;
```

Fonte: Autor

Assim, com o modelo computacional criado e sendo capaz de atender com a especificação de aleatoriedade na criação, foi então iniciado a implementação do algoritmo de *Machine Learning* com o auxílio da ferramenta Pathmind. Por se tratar de uma ferramenta externa ao *software* AnyLogic®, a primeira etapa é inserção da ferramenta ao AnyLogic®. Para realizar isso, é necessário fazer o *download* da extensão no próprio *site* do Pathmind (2021), chamado de PathmindHelper. Então é necessário adicioná-lo a paleta do AnyLogic®, como mostrado na Figura 12.

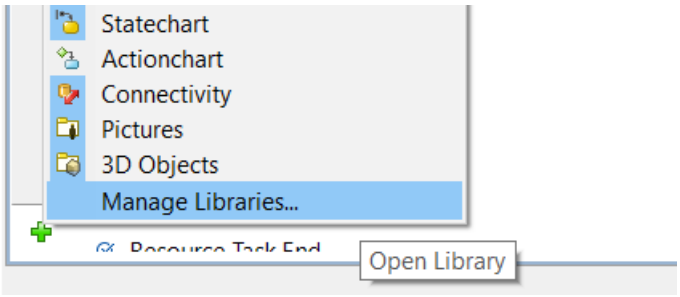


Figura 12 - Como adicionar o Pathmind ao AnyLogic®
Fonte: Autor

Depois de inserir o Pathmind no AnyLogic® através do PathmindHelper, é preciso definir os três parâmetros do aprendizado por reforço (*Reinforcement Learning values*): as observações (*Observations*), as métricas (*Metrics*) e as ações (*Actions*) e também onde é inserido a quantidade de agentes, no caso da aplicação em questão será apenas um, a peça criada chamada de "pecaVermelha", porém, é possível com mais agentes.

As observações que é onde ficam as informações necessárias, sendo nesse caso se referindo as cores de cada caixa. Para isso é necessário retomar a função "getCor" mostrada no Quadro 4 em conjunto com a variável "cores", como mostrado no Quadro 5. No caso, o sistema retoma então qual é a cor de cada caixa de acordo com seu vetor gerado.

Quadro 8 - Obervações do PathmindHelper

```
class Observations {
        int[] cores = getCor();
}
```

Fonte: Autor

O parâmetro seguinte a ser definido foram as métricas, que são responsáveis pelas recompensas, é onde é avaliado o quão eficiente tem sido. Nessa etapa deve-se definir quais itens de recompensa devem ser calculados, como no caso desse trabalho, o importante é o número de acertos das cores das caixas, assim foi criado três novas variáveis "rew_0", "rew_1" e "rew_2", sendo responsáveis pelas recompensas relativas ao acerto das cores das caixas vermelhas, verdes e azuis, respectivamente.

Essas variáveis criadas farão a contagem dos acertos ou erros do sistema. Elas serão aplicadas nas métricas e na variável "doAction". No Quadro 6 ela é representada mostrando que elas são responsáveis pelas recompensas.

Quadro 9 - Métricas do PathmindHelper

```
class Metrics {
        double reward_0 = rew_0;
        double reward_1 = rew_1;
        double reward_2 = rew_2;
}
```

Fonte: Autor

O último parâmetro a ser definido é o das ações, onde são determinadas quais ações o agente deve tomar. Primeiro deve definir o espaço da ação, no qual o Pathmind permite a utilização de diversos, porém, no trabalho é utilizado discreto. Então é necessário definir "n" que é a quantidade de ações possíveis que o agente pode tomar, no caso do trabalho ele só possui duas ações possíveis, que é o fato da caixa ser ou não ser da cor correta. Em seguida deve definir o "size", que é o número de possibilidades de ações, no exemplo são três cores diferentes, ou seja, vetor de tamanho três. Por fim é utilizado a variável "doAction", que será explicada detalhadamente, o seu código pode ser visualizado no Quadro 7.

Assim, o Pathmind consegue informações suficientes para tentar "adivinhar" qual é o vetor (representação da cor) gerado por cada caixa. Por mais que sejam criado apenas três vetores possíveis ([1,0,0]; [0,1,0]; [0,0,1]), o sistema testa todas as combinações possíveis para um vetor desse tamanho, resultando um total de oito possibilidades, sendo elas: [1,0,0]; [1,0,1]; [1,1,0]; [1,1,1]; [0,0,1]; [0,1,0]; [0,1,1]; [0,0,0].

Quadro 10 - Ações do PathmindHelper

```
class Actions {
   @Discrete(n = 2, size = 3) int[] action;
   void doIt() {  doAction(action);
   }
}
```

Fonte: Autor

A última definição, é a criação de uma variável com o nome exatamente "doAction", pois sua utilização no Pathmind necessita que a variável possua esse nome. Essa variável é responsável pela implementação das regras, no qual o Pathmind analisa se o vetor criado é igual ao vetor cor da caixa, ou seja, verifica se o sistema acertou ou errou a cor da caixa. Nessa etapa que são somadas ou subtraídas as recompensas, assim sendo, se o sistema aplica um vetor idêntico ao vetor da cor, é somado 1 ao valor da recompensa, caso contrário, é subtraído 1 desse valor.

Quadro 11 - Código da variável doAction

```
traceln("Action = "+ action[0]+" "+action[1]+" "+action[2]);

if(action[0] == 1 && pecaVermelha.getColor() == red){
        rew_0++;
        Acerto = Acerto + 1;
        if(action[0] + action[1] + action[2] > 1){
    Acerto = Acerto - 1;
   }
}else{
        if(action[0] == 1 && pecaVermelha.getColor() != red){
                rew_0--;
        }
}

if(action[1] == 1 && pecaVermelha.getColor() == green){
        rew_1++;
        Acerto = Acerto + 1;
        if(action[0] + action[1] + action[2] > 1){
    Acerto = Acerto - 1;
   }
}else{
        if(action[1] == 1 && pecaVermelha.getColor() != green){
                rew_1--;
        }
}

if(action[2] == 1 && pecaVermelha.getColor() == blue){
        rew_2++;
        Acerto = Acerto + 1;
        if(action[0] + action[1] + action[2] > 1){
    Acerto = Acerto - 1;
   }
}else{
        if(action[2] == 1 && pecaVermelha.getColor() != blue){
                rew_2--;
        }
}
traceln("Recompensas = " +rew_0 +" "+ rew_1 +" "+ rew_2);
```

Fonte: Autor

Definidos todos os parâmetros e variáveis necessários para a utilização do Pathmind, é inserido então a função que integra o Pathmindhelper ao modelo, que como na Figura 11, ocorre no "Separador", no qual é utilizada a função "pathmindHelper.triggerNextAction();" na entrada do Separador. Então, a etapa seguinte é a de exportação do modelo, no qual é necessário verificar se o experimento de simulação está utilizando uma semente aleatória de criação, ao invés de uma semente fixada.

Importante destacar que para utilizar essa ferramenta utilizando a versão PLE do AnyLogic® é necessário possuir no mínimo a versão 8.7.4 do *software*, que foi lançada no dia 4 de

maio de 2021. Para isso é preciso a criação de um experimento de RL dentro do AnyLogic® como mostrado na Figura 13, então exportá-lo para o Pathmind, como mostrado na Figura 14.

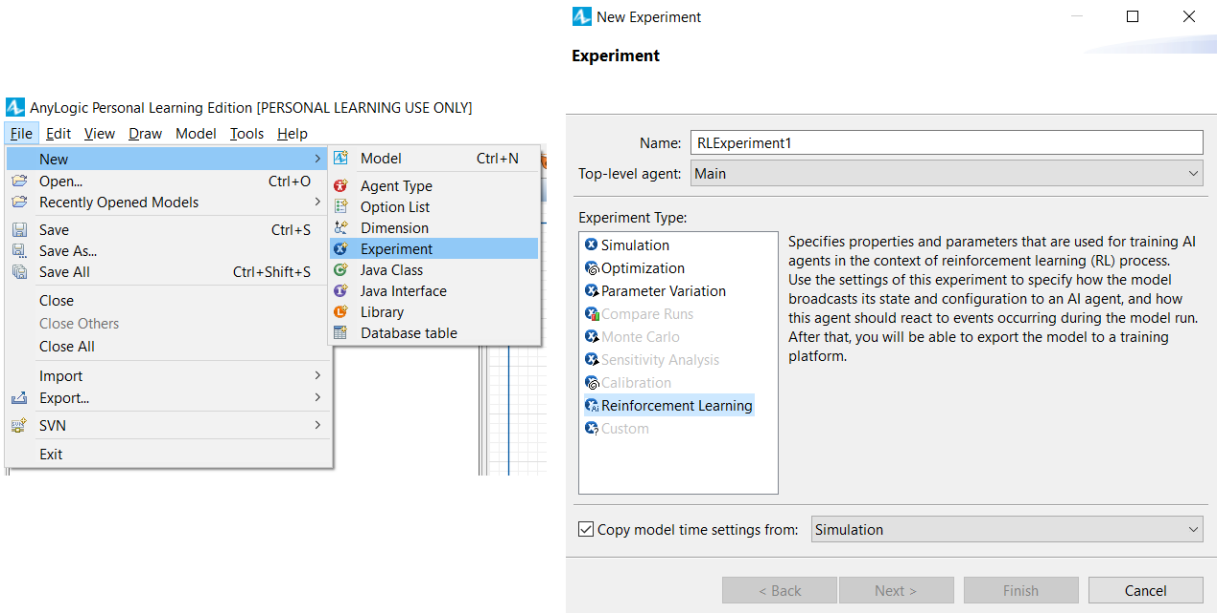


Figura 13 - Criação um experimento de RL no AnyLogic®
Fonte: Autor

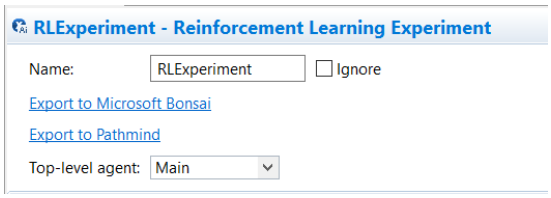


Figura 14 – Exportação do modelo para o Pathmind
Fonte: Autor

A continuidade da implementação utilizando essa ferramenta é realizada na nuvem do próprio *site* da Pathmind (2021). É preciso se cadastrar em Pathmind (2021), onde o Pathmind disponibiliza uma "Acess Token", uma chave de acesso, que deve ser inserida ao modelo no AnyLogic® para permitir a exportação.

Com isso o sistema gerará um arquivo no formato ".zip" e será redirecionado novamente a nuvem do Pathmind (2021). Nessa etapa é então direcionado quais objetivos devem ser alcançados nas recompensas, se é de maximização ou de minimização. Na aplicação atual, é de maximização para todas as recompensas, pois a cada acerto o sistema acrescenta 1 ponto na recompensa e julga isso como positivo. Para isso, é necessário inserir a função de recompensa "reward += after.reward_0 - before.reward_0" para cada uma das recompensas, como mostrado na Figura 15, e após isso treinar a política, "Train Policy".

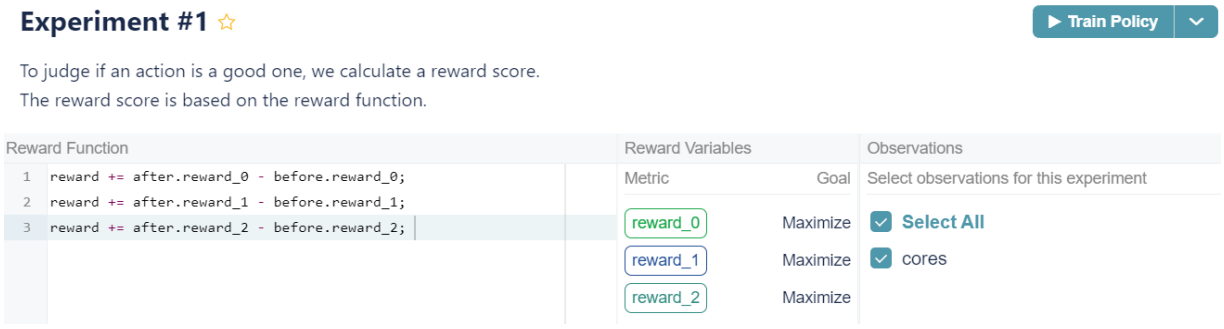


Figura 15 - Função de recompensa e treinamento de política no Pathmind
Fonte: Autor

Realizado isso, o próprio sistema do Pathmind começa a realização do treinamento da política do sistema utilizando o algoritmo de PPO como política de gradientes, assim como a hiperparametrização, utilizando a *Population-Based Training*, proposto por Jaderberg *et al*. (2017). Esse treinamento é relativamente demorado, no caso da pesquisa demorou cerca de 50 minutos, ele ocorre totalmente na nuvem do Pathmind (2021), o que evita qualquer adversidade como queda de energia, uma vez que o sistema segue o treinamento mesmo que perca a conexão.

Finalizado o treinamento, o Pathmind (2021) fornece as informações a respeito dos resultados, que se mantém disponíveis para acesso no próprio *site*. O sistema gera várias (cerca de 250) iterações, o que pode ser visto pela Figura 16. Os resultados são mostrados divididos em cada recompensa, como pode ser visto na Figura 16, onde é possível visualizar as métricas, o histograma e a média do valor da recompensa para cada uma das três recompensas.

Figura 16 - Resultados obtidos do Pathmind
Fonte: Autor

O resultado gerado é uma política que deve ser exportada para o AnyLogic®, no qual é gerado um arquivo ".zip" que deve ser implementado no Pathmind. Isso é possível dentro do Pathmindhelper, como mostrado na Figura 17, o sistema mostra que há duas opções de rodar o modelo gerado, agora utilizando o "*Use Policy"* (política) ou o "*Use Random Actions*" (forma aleatória), destacando que na Figura 17 foi removido a pasta de origem do arquivo do *Policy* gerado.

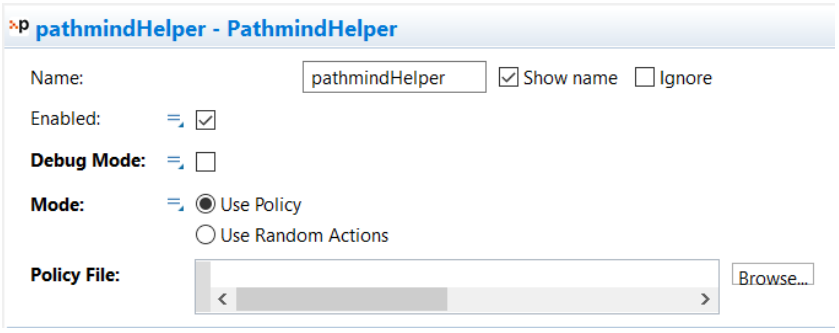


Figura 17 - Utilização da política no Pathmindhelper
Fonte: Autor

Após a exportação do *Policy* e sua implementação junto ao Pathmindhelper, o algoritmo de ML está implementado ao sistema. É possível perceber como o sistema funciona a partir da Figura 18, onde é gerado uma caixa de uma cor diferente de acordo com vetor, então o algoritmo de ML já treinado identifica qual é, e assim é capaz de colocar o vetor correspondente, acertando a resposta, e consequentemente ele aumenta em 1 o valor da recompensa para aquela cor de caixa. Importante destacar a diferença pelo uso da semente aleatória de criação, como pode ser percebido a diferente ordem de criação de caixas pelas Figuras 18 (a), 18 (b) e 18 (c), no qual são três rodadas diferentes de geração de caixas, e sempre com o algoritmo sendo capaz de identificar as cores das caixas.

```
Cores = 0 1 0
Action = 0 1 0
Recompensas = 0.0 1.0 0.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 1.0 1.0 0.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 2.0 1.0 0.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 3.0 1.0 0.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 4.0 1.0 0.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 4.0 2.0 0.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 4.0 2.0 1.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 4.0 3.0 1.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 4.0 4.0 1.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 5.0 4.0 1.0
```

(a)

```
Cores = 0 0 1
Action = 0 0 1
Recompensas = 0.0 0.0 1.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 0.0 1.0 1.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 0.0 1.0 2.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 0.0 2.0 2.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 0.0 3.0 2.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 1.0 3.0 2.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 2.0 3.0 2.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 2.0 3.0 3.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 2.0 3.0 4.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 3.0 3.0 4.0
```

(b)

```
Cores = 0 0 1
Action = 0 0 1
Recompensas = 0.0 0.0 1.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 1.0 0.0 1.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 2.0 0.0 1.0
Cores = 1 0 0
Action = 1 0 0
Recompensas = 3.0 0.0 1.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 3.0 0.0 2.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 3.0 1.0 2.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 3.0 2.0 2.0
Cores = 0 1 0
Action = 0 1 0
Recompensas = 3.0 3.0 2.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 3.0 3.0 3.0
Cores = 0 0 1
Action = 0 0 1
Recompensas = 3.0 3.0 4.0
```

(c)

Figura 18 - Três rodadas de aplicações do algoritmo de ML
Fonte: Autor

Assim, para identificar sobre a eficiência do sistema com o algoritmo implementado bem como a ferramenta Pathmind, foram inseridas duas variáveis no sistema “Total” e “Acerto”. A primeira é responsável pela contagem de quantas caixas são criadas, sendo inserido a função “Total = Total + 1;” na saída de “entradaPecas”. Enquanto a segunda variável em questão é responsável pela contagem apenas dos acertos, ou seja, quando o vetor do algoritmo de ML é idêntico ao vetor da cor da caixa criada, o que pode ser visualizado no Quadro 8. Então, é possível analisar a relação de quantas caixas foram criadas e quantos acertos o sistema teve.

4.1.1 Avaliação de resultado da implementação com ferramenta externa

Para testar então a diferença entre o modelo utilizando ou não o algoritmo de ML, foram realizadas 300 rodadas no sistema utilizando o modo aleatório e 300 utilizando o *Policy*. Essas rodadas foram divididas em 100 rodadas com o sistema simulando por 12h, 100 rodadas por 24h e 100 rodadas por 36h.

Os resultados das rodadas utilizando o modo aleatório podem ser vistos de forma sintetizadas pela Tabela 5 e de forma completa no Apêndice A. Como dito, o sistema é capaz de

gerar oito possibilidades de vetores do tamanho em questão. Assim, o número de acertos esperados para esse modo é de 12,50% de acerto.

Tabela 2 - Taxa de acertos no modo aleatório

Modo aleatório					
Período (em horas)	**Total Produzido**	**Média de Acertos**	**Média Esperada**	**Acertos (%)**	**Desvio Padrão**
12	14.400	1.800,5	1.800	12,50%	40,9
24	28.800	3.598,6	3.600	12,49%	61,1
36	43.200	5.404,9	5.400	12,51%	79,5

Fonte: Autor

Então para identificar a diferença após a implementação, foram aplicadas as mesmas 300 rodadas com o *Policy*. O resultado mostrou a eficiência da ferramenta Pathmind com o algoritmo de ML, uma vez que a taxa de acerto subiu para 100% de acerto.

Para confirmar sobre o número total de acertos, utilizando o *Policy* foi gerado uma rodada de simulação de um ano, nessa etapa foi utilizado o período de teste da versão profissional do AnyLogic®, na qual permite um número ilimitado de criação de agentes. O sistema gerou 10.512.000 caixas, possuindo um número de acertos de 10.511.882, ou seja, possuindo uma taxa de erro de apenas 0,001122%.

4.1.2 Fluxograma de implementação utilizando o Pathmind

Para mostrar de forma mais sucinta as etapas para realizar a implementação do ML em um modelo de SBA utilizando o Pathmind, foi realizado um fluxograma, com as etapas já descritas.

A primeira etapa para utilizar o Pathmind após a criação do modelo conceitual e computacional, é fazer o seu *download* no *site* Pathmind (2021) e acoplá-lo ao AnyLogic®. Então é necessário definir os três parâmetros do Pathmindhelper: observações, métricas e ações. Para a utilização desse último parâmetro, é necessário a criação de uma variável chamada “doAction”.

Definidas todas as funções necessárias do Pathmind, é preciso criar um experimento de RL no AnyLogic® e exportá-lo ao Pathmind. Feito isso, é necessário definir se deseja maximizar ou minimizar as funções na plataforma do Pathmind (2021), onde o sistema realiza o treinamento em sua nuvem.

Por fim, é criado um Policy pelo sistema que deve ser implementado ao modelo no AnyLogic®. Assim, o algoritmo de ML estará implementado no modelo. Essas informações foram sintetizadas no fluxograma da Figura 19.

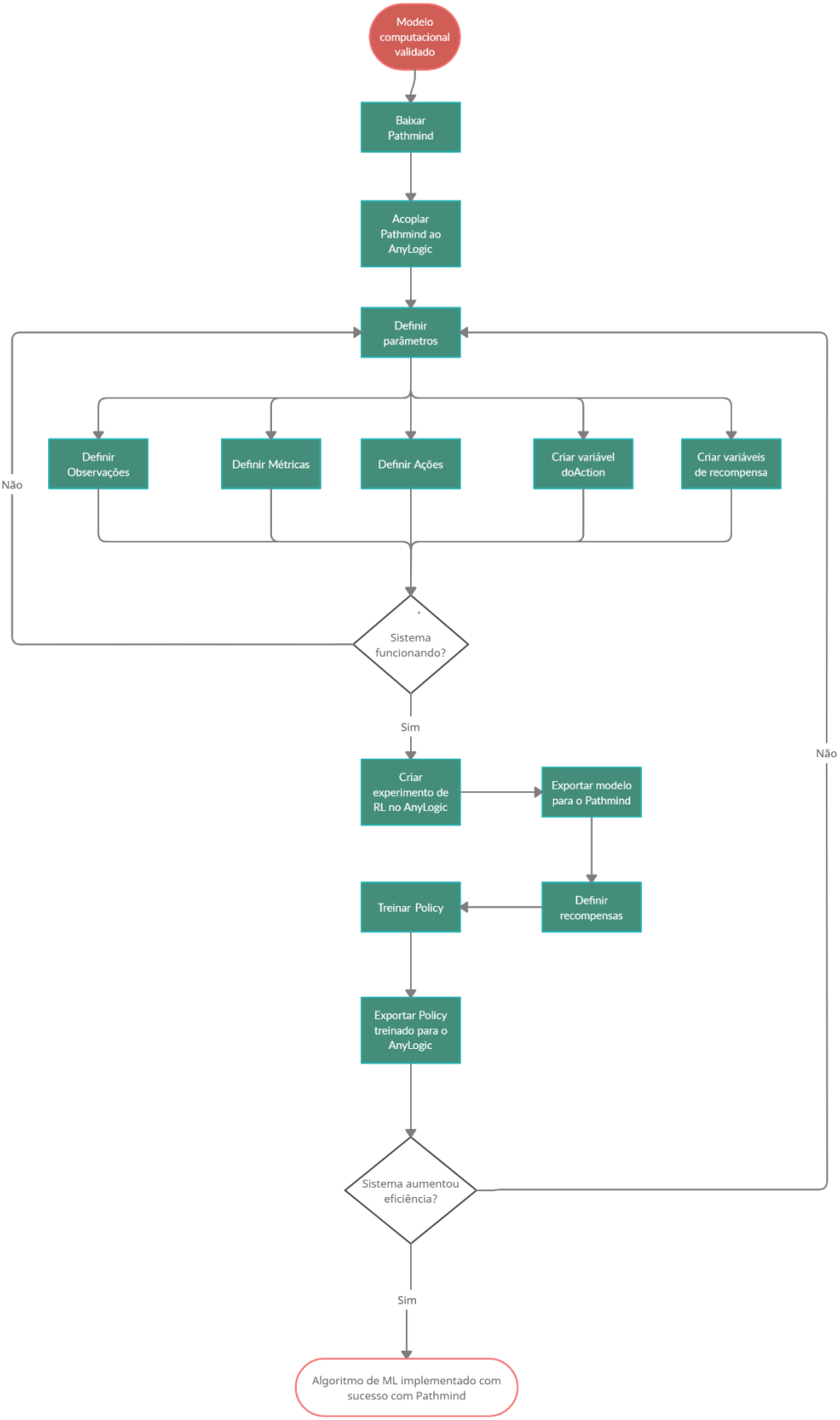


Figura 19 - Fluxograma de implementação do ML utilizando Pathmind
Fonte: Autor

4.2 Integração de forma direta

A segunda forma de implementação do aprendizado por reforço, será de forma direta, ou seja, sem a utilização de uma ferramenta externa ao *software* AnyLogic®. Essa implementação ocorrerá utilizando o algoritmo de aprendizagem por reforço *Q-learning*.

Para a essa etapa, também foi utilizado o modelo proposto adaptado de Martins *et al*. (2021). Mantendo o sistema de criação de caixas de cores diferentes de forma aleatória, e o sistema será capaz de distinguir a cor de cada caixa. Porém, para esta etapa, as cores das caixas não serão representadas por vetores, como no caso da aplicação da ferramenta externa, mas sim a partir de *strings*, e nesse caso serão utilizadas cinco cores diferentes, sendo elas: “AZUL”, “VERMELHO”, “VERDE”, “BRANCO” e “PRETO”.

A criação das caixas ocorre de forma aleatória, criando a cada cinco caixas, uma de cada cor e em sequências diferentes. Inicialmente foi criada uma variável chamada “CaixasAleatorias”, esta é definida como uma *array* de *string* de tamanho cinco, que inicia vazia. Então é criada uma função, mostrada no Quadro 9, onde criou-se uma *array* que recebe as cinco cores em questão, e após isso, é gerada uma lista de cinco números aleatórios e depois alocando esses números de acordo a ordem das cores da caixa, fazendo assim, com que a ordem das cores criadas seja de forma aleatória.

Quadro 12 - Função de aleatoriedade das caixas

```
String[] CoresCaixas = new String[] { "AZUL", "VERMELHO", "VERDE",
"BRANCO", "PRETO"};

List<Integer> lista = new ArrayList<Integer>();
Integer valor;

while(lista.size() < 5) {
	valor = (int) (Math.random() * 5);
	if(!lista.contains(valor)) {
		lista.add(valor);
	}
}

CaixasAleatorias[0] = CoresCaixas[lista.get(0)];
CaixasAleatorias[1] = CoresCaixas[lista.get(1)];
CaixasAleatorias[2] = CoresCaixas[lista.get(2)];
CaixasAleatorias[3] = CoresCaixas[lista.get(3)];
CaixasAleatorias[4] = CoresCaixas[lista.get(4)];
```

Fonte: Autor

Feito isso, foram criadas duas novas variáveis “Total” e “Contador”, sendo a primeira responsável pela contagem total de caixas criadas, enquanto a segunda a referência para que o sistema seja capaz de identificar que a cada cinco caixas deverá ser iniciada uma nova contagem

para a criação de novas cinco caixas com uma nova sequência aleatória e também para que o sistema ative o algoritmo de aprendizado *Q-learning*.

Isso pode ser visualizado a partir do Quadro 10, onde é mostrado o código na criação das caixas. Nele é possível visualizar que o sistema ativa a função mostrada no Quadro 9 a cada cinco novas caixas criadas. Feito isso, o código emite qual é a cor da caixa criada, o que pode ser verificada a aleatoriedade do sistema pela Figura 20, onde são mostradas três sequências de cinco caixas criadas. Então o código ativa o algoritmo *Q-learning*, mas isso apenas após a criação de cinco caixas, para conseguir então identificar as cores das últimas cinco caixas criadas. E por fim, é zerado a variável "Contador", para que o algoritmo ocorra a cada cinco caixas.

Quadro 13 - Código criação das caixas e inicialização do Q-learning

```
if (Contador % 5 == 0) {function();};
Contador = Contador + 1;
Total = Total + 1;

System.out.println();
System.out.println("A cor da caixa criada é: " + CaixasAleatorias[Contador - 1]);

if (Contador == 5){long BEGIN = System.currentTimeMillis();
//QLearning obj = new QLearning();
QLearning obj = new QLearning(CaixasAleatorias);
obj.run();
obj.printResult();
long END = System.currentTimeMillis();
System.out.println("Time: " + (END - BEGIN) / 1000.0 + " sec.");};

if (Contador == 5) {Contador = 0;
};
```

Fonte: Autor

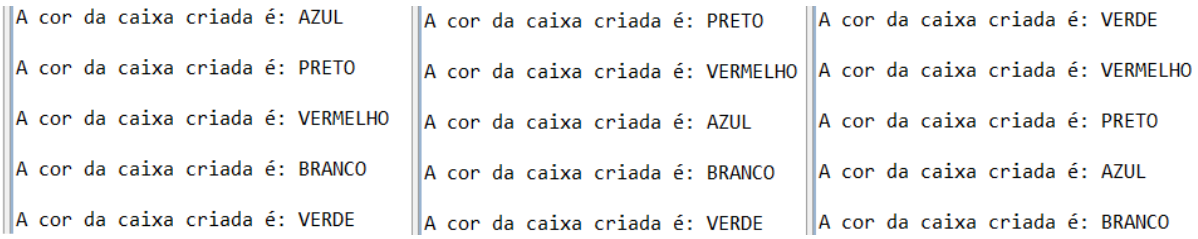


Figura 20 - Três rodadas de criação de caixas de forma aleatória
Fonte: Autor

Assim, foi finalizado a etapa de criação do modelo, então na sequência é implementado o algoritmo *Q-learning*, para aplicar o aprendizado por reforço que seja capaz de identificar a cor das caixas. Para isso, primeiramente teve que se desenvolver o algoritmo *Q-learning* na linguagem de programação Java. Porque foi percebido que não há esse algoritmo de forma mais completa e sem ser independente de um problema de questão nessa linguagem de programação. Por isso, se tornou

necessário a criação de um algoritmo que possa funcionar de forma completamente independente, se tornando altamente replicável.

Então foi desenvolvido um algoritmo *Q-learning* em linguagem de programação Java que seja completo e que principalmente seja replicável, uma vez que ele foi desenvolvido de forma aberta, no qual a etapa de treinamento age independente dos estados do sistema, fazendo com que outro modelador possa apenas adequar para que esse código funcione de acordo com seu próprio modelo.

É importante destacar novamente sobre como é o funcionamento desse algoritmo. Como já dito, ele funciona a partir do aprendizado por reforço através de várias iterações. Para isso, são definidos os hiperparâmetros de taxa de aprendizado e coeficiente de desconto. E o algoritmo produz uma nova ação para cada estado criado, sendo que essa ação gera uma recompensa ou punição, e esse resultado é armazenado em uma matriz de recompensas (R) que gera uma matriz de resultado (Q). Assim, o sistema vai iterando, criando novos estados a partir de ações cada vez mais inteligentes de acordo com o valor de Q, sendo o número de iterações pré-determinado.

A primeira etapa para implementação propriamente dita desse algoritmo *Q-learning* dentro do *software* AnyLogic® é a criação de uma nova classe Java, vale ressaltar novamente que essa que é a linguagem de programação utilizada dentro desse *software*. Para a criação dessa classe, é necessário que seja dentro do projeto, como mostrado na Figura 21. O algoritmo utilizado dentro dessa classe pode ser visualizado de forma completa no Apêndice B.

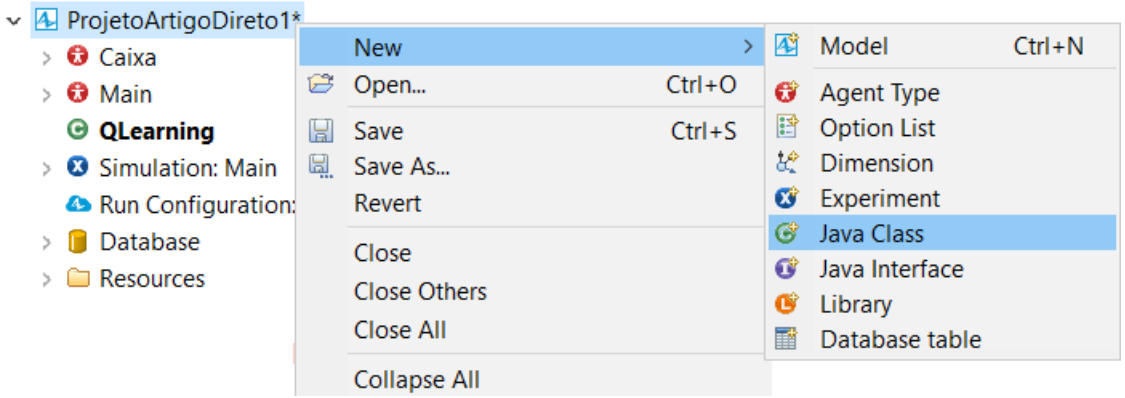


Figura 21 - Criação de uma nova classe Java
Fonte: Autor

É necessário modelar o algoritmo para que ele funcione de acordo com o modelo proposto. Primeira definição que deve ser feita se refere aos estados possíveis que o sistema terá. No caso em questão, cada é representado pelas caixas criadas, no qual, cada cor diferente é um estado diferente possível, assim sendo, serão cinco estados possíveis, como pode ser visualizado no Quadro 11.

Outra definição inicial necessária diz respeito a quais ações são possíveis para determinado estado. Como em um exemplo de labirinto (aplicação muito utilizada para o algoritmo *Q-learning*), são os caminhos possíveis que o agente pode seguir, seja para baixo, para cima, para o lado, etc. No caso da pesquisa, as ações possíveis são as possibilidades de cores para cada caixa criada, sendo aqui definidas que cada caixa pode ser de qualquer uma das cinco possíveis cores.

Quadro 14 - Código definição dos estados e suas ações

```
final int stateA = 0;
final int stateB = 1;
final int stateC = 2;
final int stateD = 3;
final int stateE = 4;

final int[] states = new int[]{stateA,stateB,stateC,stateD,stateE};

final int statesCount = states.length;

int[] actionsFromStateA = new int[] { stateA, stateB, stateC, stateD, stateE};
int[] actionsFromStateB = new int[] { stateA, stateB, stateC, stateD, stateE};
int[] actionsFromStateC = new int[] { stateA, stateB, stateC, stateD, stateE};
int[] actionsFromStateD = new int[] { stateA, stateB, stateC, stateD, stateE};
int[] actionsFromStateE = new int[] { stateA, stateB, stateC, stateD, stateE};


int[][] actions = new int[][] { actionsFromStateA, actionsFromStateB,
actionsFromStateC, actionsFromStateD, actionsFromStateE};


String[] stateNames = new String[] { "VERMELHO", "VERDE", "AZUL",
"BRANCO", "PRETO"};
```

Fonte: Autor

Então, é preciso configurar as recompensas do sistema. Como já dito, o algoritmo de aprendizagem por reforço funciona a partir de recompensas e punições, assim é necessário definir o que será recompensa positiva ou o que será uma punição para o sistema. Na pesquisa é feita uma comparação entre o estado gerado e a ação realizada sobre aquele determinado estado, e caso eles sejam iguais, o sistema gera uma recompensa positiva no valor de 100, caso contrário, um valor negativo de -25. Esses valores são utilizados para construir as matrizes R de recompensa e Q de resultado.

Quadro 15 - Código definição das recompensas

```
    public void init() {

      for (int i = 0; i < statesCount; i++) {
         int state = states[i];
         int[] actionsFromState = actions[state];

            for (int j = 0; j < actionsFromState.length; j++) {

               int reward = -25;

               if (state == actionsFromState[j]) reward = 100;

               setQ(state, actionsFromState[j], reward);
               setR(state, actionsFromState[j], reward);

            }
      }

  }
```

Fonte: Autor

A definição que foi realizada na sequência é a respeito da apresentação dos resultados. Nessa pesquisa optou-se por apresentar o resultado mostrando a matriz de resultado Q com todos seus valores. Assim, a resposta ótima será mostrada de acordo com aquela ação gerada que possuir maior valor na matriz Q, o que pode ser visualizado no Quadro 13. Vale ressaltar que o valor de resultado ótimo no algoritmo *Q-learning* tende ao valor de 1.

Quadro 16 - Código apresentação dos resultados

```
   void printResult() {

      for (int i = 0; i < statesCount; i++) {
         System.out.print("A CAIXA " + stateNames[i] + " TEM AS SEGUINTES
POSSIBILIDADES DE PALPITES:  \n");
         double maxValueQState = Double.MIN_VALUE;

         for (int j = 0; j < Q[i].length; j++) {
            double value = Q[i][j];

            if (value > maxValueQState) maxValueQState = value;

            if (value > 0) System.out.print(stateNames[j] + " " + value+ "\n");
         }

         System.out.print("O PALPITE CORRETO PARA ACERTAR A COR DA
CAIXA " + stateNames[i] +" TEM VALOR DE " + maxValueQState + "\n");
         System.out.println();
      }
   }
```

Fonte: Autor

A seguir é inserida a parte do algoritmo que é o algoritmo *Q-learning* de forma pura, que pode ser visualizado pelo Apêndice B, onde, de fato, é realizado o treinamento do sistema. É importante ressaltar que essa parte do código não deve ser alterada, uma vez que ao se alterar a parte do treinamento, é alterada a essência do próprio algoritmo. Ou seja, é importante separar a parte da modelagem para se adequar à questão de problema para a parte do aprendizado do *Q-learning* propriamente dita.

Por fim, para configurar o algoritmo, diferentemente de quando se utiliza a ferramenta externa Pathmind, é necessário definir os hiperparâmetros e o número de treinamentos a serem realizados. Ambos hiperparâmetros (alpha e gamma) variam entre 0 e 1, alpha é responsável pela taxa de aprendizagem, quanto mais baixo, menor a alteração no sistema. Enquanto gamma é responsável pelo valor que as recompensas futuras terão em relação as imediatas. Assim sendo, os valores que melhor se adequaram para o modelo e que apresentaram os melhores resultados foram de 0,9 para ambos hiperparâmetros, e com 10.000 rodadas de treinamento.

Quadro 17 - Código definição dos hiperparâmetros e número de treinamentos

```
Random rand = new Random();

final double alpha = 0.9;
final double gamma = 0.9;
final int countLearningTimes = 10000;
```

Fonte: Autor

Definidos todos parâmetros foi então para a etapa de avaliação dos resultados e eficiência do sistema. Para isso, assim como no caso da ferramenta externa, foram utilizadas 300 rodadas do sistema.

Em todos os casos o sistema apresentou uma eficiência ótima, uma vez que foi capaz de identificar as cores das caixas criadas a partir da análise da matriz resultado Q. Um exemplo de resultado pode ser visto na Figura 22. Vale ressaltar que de acordo com os hiperparâmetros e número de iterações, o sistema só apresentou alterações no resultado a partir da 11ª casa decimal, o que se torna um valor irrisório.

```
A cor da caixa criada é: VERMELHO

A cor da caixa criada é: VERDE

A cor da caixa criada é: BRANCO

A cor da caixa criada é: AZUL

A cor da caixa criada é: PRETO
A CAIXA VERMELHO TEM AS SEGUINTES POSSIBILIDADES DE PALPITES:
VERMELHO 999.9999999999997
VERDE 874.9999999999997
BRANCO 874.9999999999997
AZUL 874.9999999999997
PRETO 874.9999999999997
O PALPITE CORRETO PARA ACERTAR A COR DA CAIXA VERMELHO TEM VALOR DE 999.9999999999997

A CAIXA VERDE TEM AS SEGUINTES POSSIBILIDADES DE PALPITES:
VERMELHO 874.9999999999997
VERDE 999.9999999999997
BRANCO 874.9999999999997
AZUL 874.9999999999997
PRETO 874.9999999999997
O PALPITE CORRETO PARA ACERTAR A COR DA CAIXA VERDE TEM VALOR DE 999.9999999999997

A CAIXA BRANCO TEM AS SEGUINTES POSSIBILIDADES DE PALPITES:
VERMELHO 874.9999999999997
VERDE 874.9999999999997
BRANCO 999.9999999999997
AZUL 874.9999999999997
PRETO 874.9999999999997
O PALPITE CORRETO PARA ACERTAR A COR DA CAIXA BRANCO TEM VALOR DE 999.9999999999997

A CAIXA AZUL TEM AS SEGUINTES POSSIBILIDADES DE PALPITES:
VERMELHO 874.9999999999997
VERDE 874.9999999999997
BRANCO 874.9999999999997
AZUL 999.9999999999997
PRETO 874.9999999999997
O PALPITE CORRETO PARA ACERTAR A COR DA CAIXA AZUL TEM VALOR DE 999.9999999999997

A CAIXA PRETO TEM AS SEGUINTES POSSIBILIDADES DE PALPITES:
VERMELHO 874.9999999999997
VERDE 874.9999999999997
BRANCO 874.9999999999997
AZUL 874.9999999999997
PRETO 999.9999999999997
O PALPITE CORRETO PARA ACERTAR A COR DA CAIXA PRETO TEM VALOR DE 999.9999999999997

Time: 0.008 sec.
```

Figura 22 - Resultados da aplicação direta
Fonte: Autor

4.2.1 Fluxograma de implementação de forma direta

Assim como foi feito para a aplicação do aprendizado por reforço utilizando a ferramenta externa, também foi elaborado um fluxograma para mostrar, de forma sucinta e clara, as etapas necessárias para implementar o RL de forma direta no AnyLogic® por meio do algoritmo *Q-learning*.

O primeiro passo após ter o modelo computacional funcionando foi a criação de uma nova classe de Java dentro do projeto, pois é nessa área que é inserido o algoritmo *Q-learning*. Dentro dessa classe foi necessário inserir o algoritmo e modelá-lo de acordo com o modelo. E também deve-se colocar dentro do sistema um código para chamar a classe ocorrer quando necessário.

Primeira alteração no código que deveria ser feito foi definir quais os estados possíveis do sistema e depois cada ação possível para cada estado criado. Feito isso, foram definidas as recompensas e punições do sistema.

Então é necessário configurar como serão mostrados os resultados obtidos. Feito isso, é inserida a parte do código do treinamento *Q-learning*, parte essa que não deve ser alterada. E para finalizar, precisa configurar os hiperparâmetros e o número de iterações do sistema, a fim de encontrar um valor ótimo para cada sistema. Essas informações são mostradas de forma mais clara no fluxograma da na Figura 23.

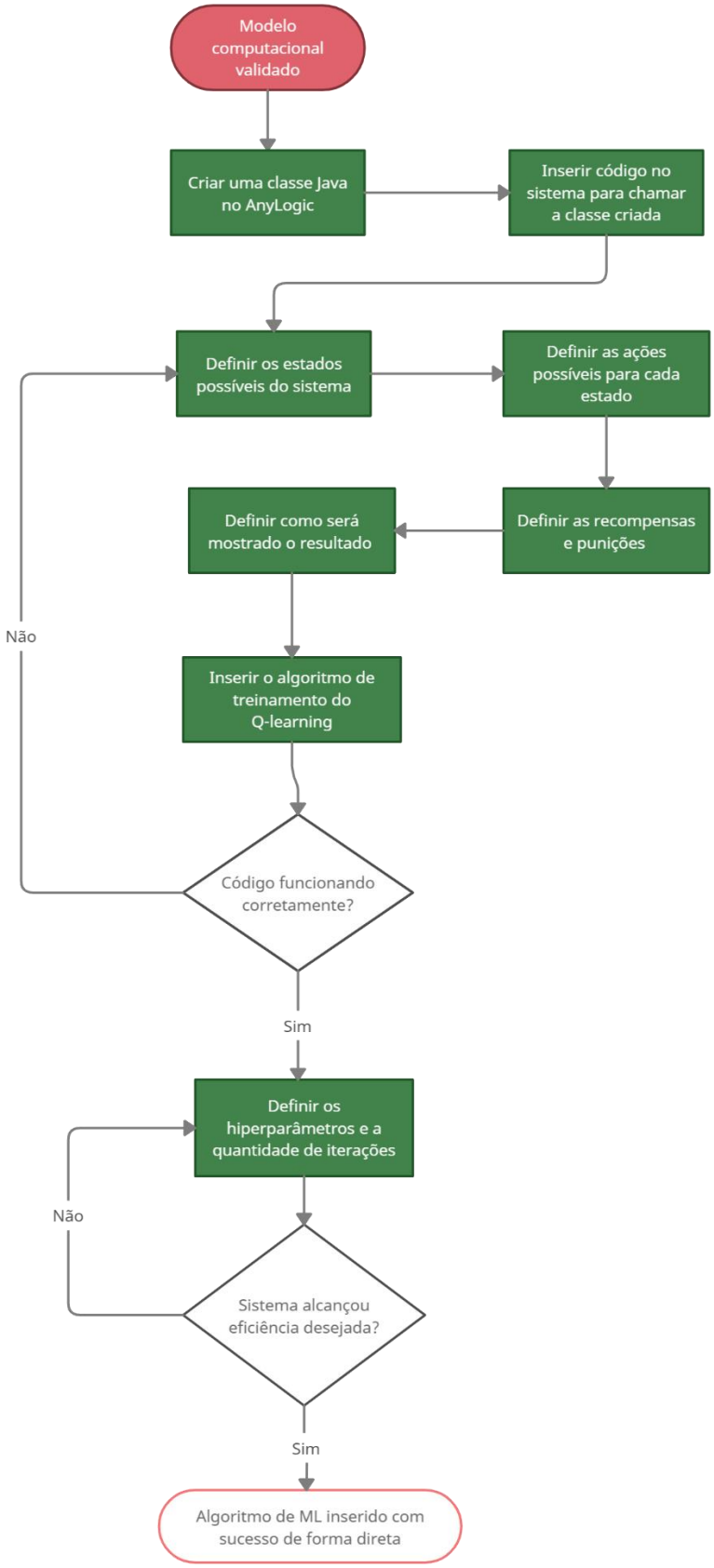


Figura 23 - Fluxograma de implementação do ML de forma direta
Fonte: Autor

4.3 Comparativo entre as duas formas de implementação

Após finalizar a implementação das duas formas, é importante destacar alguns pontos acerca de ambos os métodos.A primeira forma, utilizando a ferramenta externa Pathmind, se mostrou uma ferramenta muito prática, uma vez que não necessita de tantos conhecimentos especializados sobre ML, já que o treinamento do sistema é realizado pelo próprio Pathmind. Isso torna-se uma forma mais rápida e direta de se implementar o ML em um modelo de SBA, permitindo um foco maior na própria modelagem em si.

Porém, vale destacar que nessa forma também apresenta alguns empecilhos, por se tratar de uma ferramenta externa ao AnyLogic®, cria-se uma dependência do uso dessa ferramenta, que pode dificultar seu uso, como por exemplo deixando de ser uma ferramenta gratuita, ou até mesmo não possibilitando mais o seu uso, caso essa ferramenta seja removida do mercado. Outro ponto negativo é que por se tratar de uma ferramenta *blackbox*, na qual não se tem acesso ao código de origem, acaba-se perdendo flexibilidade, uma vez que é impossível alterar esse código fonte de treinamento.

Se tratando da forma de implementação direta, é uma forma que apresenta basicamente o contrário das vantagens e desvantagens da utilização da ferramenta externa. Pois é um método que permite uma flexibilidade no código, onde pode ser visualizado a alterado da forma que for necessário e também, por não necessitar de uma ferramenta externa, é impossível que se impossibilite ou dificulte seu uso, uma vez que só é necessário possuir o *software* AnyLogic®.

Mas, em contrapartida, para realizar esse tipo de implementação de forma direta, foi necessário um altíssimo nível de conhecimento de linguagem de programação Java, que é a utilizada no *software* AnyLogic®, uma vez que teve que ser desenvolvido um código de algoritmo de aprendizagem por reforço *Q-learning*, já que não existiam código assim disponível, que seja tão completo e adaptável. E consequentemente, faz com que se exija um nível de conhecimento de ML muito superior por parte do modelador, por se tratar de uma programação de alto nível de complexidade.

Essas informações sobre vantagens e desvantagens das duas formas de implementação do ML em um modelo de SBA, podem ser visualizadas de forma sintetizadas na Tabela 6.

Tabela 3 - Comparativo entre as duas formas de integração

	Vantagens	**Desvantagens**
Ferramenta externa	• Mais rápido, prático e direto; • Não necessita alto conhecimento de ML.	• Dependência do uso da ferramenta; • Sem flexibilidade por ser *blackbox*.
Forma direta	• Não há dependência de nenhuma ferramenta; • Flexibilidade total no código.	• Alto nível de complexidade; • Necessita conhecimentos especializados de ML e programação.

Fonte: Autor.

Por fim, é importante destacar que essas não são as únicas formas possíveis de implementação do ML em um modelo de SBA. Outras formas por exemplo, é a partir do Project Bonsai (2022) ou também, por meio da linguagem de programação Python, porém, para utilizar esse tipo de linguagem no *software* AnyLogic®, é necessária a utilização do aplicativo Py4J (2022) e da biblioteca Alpyne disponível em Github (2022).

5. CONCLUSÕES

5.1 Verificação dos objetivos e resposta à questão de Pesquisa

O trabalho destaca como na era atual a geração de dados está aumentando cada vez mais, o *Big Data*, e com essa enorme quantidade de dados, vêm aumentando também a dificuldade de interpretá-los e gerar bons resultados a partir deles. Uma saída para esse problema vem sendo o uso do *Machine Learning*, mais especificamente com o aprendizado por reforço. Assim como o seu uso em conjunto com a simulação computacional, no caso da pesquisa em questão, a Simulação Baseada em Agentes.

Apesar de já possuírem diversos trabalhos já publicados que utilizam essa combinação de SBA com ML, essa implementação não se dá de forma tão simples e intuitiva. Por isso esse trabalho buscou mostrar como pode ser realizada essa combinação, a fim de torná-la mais simples para os demais modeladores. Para tornar completa a demonstração de como é possível essa implementação, o intuito é mostrar de duas formas possíveis: utilizando uma ferramenta externa, o Pathmind, e de forma direta através de programação Java no *software* AnyLogic®.

O trabalho alcançou seus três objetivos propostos. O primeiro objetivo especifico da pesquisa, foi concluído ao se apresentar a forma de introduzir o aprendizado por reforço utilizando a ferramenta externa Pathmind. Para esse, foi desenvolvido um modelo de criação de caixas de cores diferentes, no qual o sistema deveria ser capaz de acertar as cores através do *Machine Learning*. Com o detalhamento apresentado, será possível para um modelador utilizar as etapas propostas e aplicar em seu modelo.

O Pathmind se mostrou uma ferramenta eficiente, uma vez que antes da implementação da ferramenta, o sistema se comportava de forma aleatória, acertando apenas 12,5% do número de caixas, valor que subiu para 100% após a implementação. Além disso, essa ferramenta mostrou benefícios como não ter a necessidade que o modelador seja um especialista na área de *Machine Learning*, fazendo com que o modelador possa focar mais na área de modelagem. Outra vantagem dessa ferramenta é o fato dela ter se tornado acessível para a versão gratuita do *software* AnyLogic®.

Os outros dois objetivos específicos foram concluídos em sequência. Primeiramente foi desenvolvido o algoritmo de aprendizagem por reforço *Q-learning* na linguagem de programação Java. E então, esse algoritmo foi implementado com sucesso no modelo criado, sendo capaz de identificar as cores das caixas criadas a partir do algoritmo *Q-learning*. Ressaltando que foi dado

uma ênfase na demonstração das etapas a serem seguidas. O algoritmo também se mostrou bastante eficiente, uma vez que o sistema só apresentou alterações nos resultados a partir da 11ª casa decimal.

Feito isso, pode-se afirmar que o trabalho concluiu seu objetivo geral, uma vez que foi capaz de demonstrar como é realizada a implementação do ML à SBA no *software* AnyLogic®. Sendo capaz de demonstrar duas formas de realizar essa implementação, permitindo ao modelador escolher a que melhor se adeque à situação.

Por fim, se comparou as duas formas de implementação. Onde é possível perceber como as duas formas são praticamente antagônicas. Pois enquanto ao utilizar a ferramenta externa, exige um menor conhecimento de ML e é um método mais prático e rápido, mas essa forma possui uma dependência da ferramenta e é menos flexível. Já de forma direta, não há dependência de nenhuma outra ferramenta e é mais flexível, porém exige um alto conhecimento de ML e de programação, se tornando mais complexa e demorada.

5.2 Sugestões para continuidade do trabalho

Assim, como sugestão de trabalhos futuros, sugere-se a replicação desses métodos aqui apresentados em sistemas reais mais complexos. Bem como também a implementação utilizando outras formas aqui citadas, que é o caso utilizando o Microsoft Bonsai (2022) ou utilizando linguagem de programação Python a partir do Py4J (2022) utilizando a biblioteca do Github (2022).

5.3 Contribuições do trabalho

Vale ressaltar que este trabalho gerou, como resultado, um artigo em congresso que foi apresentado no Simpósio Brasileiro de Pesquisa Operacional (SBPO) de 2021.

REFERÊNCIAS

ABDOLLAHI, E.; HAWORTH-BROCKMAN, M.; KEYNAN, Y.; LANGLEY, J. M.; MOGHADAS, S. M. Simulating the effect of school closure during COVID-19 outbreaks in Ontario, Canada. **BMC medicine**, v. 18, n. 1, p. 1-8, 2020.

ALCÁCER, V.; CRUZ-MACHADO, V. Scanning the industry 4.0: A literature review on technologies for manufacturing systems. **Engineering science and technology, an international journal**, v. 22, n. 3, p. 899-919, 2019.

APPOLINÁRIO, F. **Metodologia da ciência - filosofia e prática da pesquisa**. São Paulo: Editora Pioneira Thomson Learning, 2006.

ARENALES, M.; ARMENTANO, V.; MORABITO, R.; YANASSE, H. **Pesquisa Operacional para cursos de engenharia: Modelagem e algoritmos.** Rio de Janeiro: Editora Campus, 2007.

ARMSTRONG, J. B.; FULLERTON, A. H.; JORDAN, C. E.; EBERSOLE, J. L.; BELLMORE, J. R.; ARISMENDI, I.; PENALUNA, B. E.; REEVES, G. H. The importance of warm habitat to the growth regime of cold-water fishes. **Nature Climate Change**, v. 11, n. 4, p. 354-361, 2021.

AUGUSTIJN, E. W.; ABDULKAREEM, S. A.; SADIQ, M. H.; ALBABAWAT, A. A. Machine learning to derive complex behaviour in agent-based modellzing. In: **2020 International Conference on Computer Science and Software Engineering (CSASE)**. IEEE, 2020. p. 284-289.

BABICEANU, R. F.; SEKER, R. Big Data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook. **Computers in Industry**, v. 81, p. 128-137, 2016.

BAINES, T. S.; MASON, S.; SIEBERS, P. O.; LADBROOK, J. Humans: the missing link in manufacturing simulation? **Simulation Modelling Practice and Theory**, 12: 515–526, 2004.

BANKS, J.; CARSON, J.S.; NELSON, B.L.; NICOL, D.M. **Discrete-event system simulation**. 2 ed., New Jersey: Prentice Hall, 2005.

BATATA, O.; AUGUSTO, V.; XIE, X. Mixed machine learning and agent-based simulation for respite care evaluation. In: **2018 Winter Simulation Conference (WSC)**. IEEE, 2018. p. 2668-2679.

BEAN, W. L.; JOUBERT, J. W. An agent-based implementation of freight receiver and carrier collaboration with cost sharing. **Transportation Research Interdisciplinary Perspectives**, v. 11, p. 100416, 2021.

BERTRAND, J. W. M.; FRANSOO, J. C. Modelling and Simulation: operations management research methodologies using quantitative modeling. **International Journal of Operations & Production Management**, 22:241-264, 2002.

BOBASHEV, G. V.; GOEDECKE, D. M.; YU, F.; EPSTEIN, J. M. A hybrid epidemic model: combining the advantages of agent-based and equation-based approaches. In: **2007 Winter Simulation Conference**. IEEE, 2007. p. 1532-1537.

BONABEAU, E. Agent-based modeling: Methods and techniques for simulating human systems. **Proceedings of the national academy of sciences**, v. 99, n. suppl 3, p. 7280-7287, 2002.

BORSHCHEV, A.; FILIPPOV, A. From system dynamics and discrete event to practical agent based modeling: reasons, techniques, tools. In: **Proceedings of the 22nd international conference of the system dynamics society**. 2004.

BOSE, S.; KREMERS, E.; MENGELKAMP, E. M.; EBERBACH, J.; WEINHARDT, C. Reinforcement learning in local energy markets. **Energy Informatics**, v. 4, n. 1, p. 1-21, 2021.

BOSSE, S. Self-organising Urban Traffic control on micro-level using Reinforcement Learning and Agent-based Modelling. In: **Proceedings of SAI Intelligent Systems Conference**. Springer, Cham, 2020. p. 745-764.

BRAILSFORD, S. Discrete-event simulation is alive and kicking!. In: **Agent-Based Modeling and Simulation**. Palgrave Macmillan, London, 2014. p. 291-306.

BROWN, B.; CHUI, M.; MANYIKA, J. Are you ready for the era of 'big data'. **McKinsey Quarterly**, v. 4, n. 1, p. 24-35, 2011.

CAI, Y.; HUANG, T.; BOMPARD, E.; CAO, Y.; LI, Y. Self-sustainable community of electricity prosumers in the emerging distribution system. **IEEE Transactions on Smart Grid**, v. 8, n. 5, p. 2207-2216, 2016.

CHAN, W. K. V.; SON, Y.; MACAL, C. M. Agent-based simulation tutorial-simulation of emergent behavior and differences between agent-based simulation and discrete-event simulation. In: **Proceedings of the Winter Simulation Conference**, p. 135-150, 2010.

CHWIF, L.; MEDINA, A. **Modelagem e simulação de eventos discretos: Teoria e aplicações**. 4ª edição ed. São Paulo: Elsevier Brasil, 2015.

CONCOLATO, C. E.; CHEN, L. M. Data science: A new paradigm in the age of big-data science and analytics. **New Mathematics and Natural Computation**, v. 13, n. 02, p. 119-143, 2017.

DARVILLE, J.; CELIK, N. Simulation and Optimization for Unit Commitment using a Regionbased Sampling (RBS) Algorithm. In: **IIE Annual Conference. Proceedings**. Institute of Industrial and Systems Engineers (IISE), 2020. p. 1424-1430.

DEVILLERS, J.; DEVILLERS, H.; DECOURTYE, A.; AUPINEL, P. Internet resources for agent-based modelling. **SAR and QSAR in Environmental Research**, 21(3-4), 337-350, 2010.

DONG, F.; LIU, H.; LU, B. Agent-based simulation model of single point inventory system. **Systems Engineering Procedia**, v. 4, p. 298-304, 2012.

DUBIEL, B.; TSIMHONI, O. Integrating agent based modeling into a discrete event simulation. In: **Proceedings of the Winter Simulation Conference, 2005.** IEEE, p. 9, 2005.

ESMAEILZADEH, E.; GRENN, M. W.; ROBERTS, B. An agent-based model for improved system of systems decision making in air transportation. **Systems Engineering**, v. 22, n. 1, p. 20-42, 2019.

FARHAN, M.; GÖHRE, B.; JUNPRUNG, E. Reinforcement learning in anylogic simulation models: a guiding example using pathmind. In: **2020 Winter Simulation Conference (WSC)**. IEEE, 2020. p. 3212-3223.

FRANCESCHINI, F.; MAISANO, D.; MASTROGIACOMO, L. Scientific journal publishers and omitted citations in bibliometric databases: Any relationship?. **Journal of Informetrics**, v. 8, n. 3, p. 751-765, 2014.

FULLER, D. B.; DE ARRUDA, E. F.; FERREIRA FILHO, V. J. M. Learning-agent-based simulation for queue network systems. **Journal of the Operational Research Society**, v. 71, n. 11, p. 1723-1739, 2020.

GARCIA, R. Uses of agent-based modeling in innovation/new product development research. **Journal of Product Innovation Management**, v. 22, n. 5, p. 380-398, 2005.

GITHUB. **The AnyLogic-Python conector**, 2022. Disponível em <https://github.com/t-wolfeadam/Alpyne> Acesso em: 8 de março de 2022.

HARRELL, C., GHOSH, B. K., BOWDEN, R. **Simulation Using Promodel**. 3. ed., Boston: McGraw-Hill, 2004.

HASSANPOUR, S.; RASSAFI, A. A.; GONZÁLEZ, V. A.; LIU, J. A hierarchical agent-based approach to simulate a dynamic decision-making process of evacuees using reinforcement learning. **Journal of choice modelling**, v. 39, p. 100288, 2021.

HERMANN, M.; PENTEK, T.; OTTO, B. Design principles for industrie 4.0 scenarios. In: **2016 49th Hawaii international conference on system sciences (HICSS)**. IEEE, 2016. p. 3928-3937.

HUTTY, T. D.; DONG, S.; BROWN, S. Suitability of energy storage with reversible solid oxide cells for microgrid applications. **Energy Conversion and Management**, v. 226, p. 113499, 2020.

IRANNEZHAD, E.; PRATO, C. G.; HICKMAN, M. An intelligent decision support system prototype for hinterland port logistics. **Decision Support Systems**, v. 130, p. 113227, 2020.

JADERBERG, M.; DALIBARD, V.; OSINDERO, S.; CZARNECKI, W. M.; DONAHUE, J.; RAZAVI, A.; VINYALS, O.; GREEN, T.; DUNNING I.; SIMONYAN, K.; FERNANDO C.; KAVUKCUOGLU, K. Population based training of neural networks. **arXiv preprint arXiv:1711.09846**, 2017.

JAHANGIRIAN, M.; ELDABI, T.; NASEER, A.; STERGIOULAS, L. K.; YOUNG, T. Simulation in manufacturing and business: A review. **European Journal of Operational Research**, v. 203, n. 1, p. 1–13, 2010.

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

KÄLLSTRÖM, J. Adaptive Agent-Based Simulation for Individualized Training. In: **19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), May9–13, 2020, Auckland, New Zealand.** International Foundation for Autonomous Agents and Multiagent Systems (www. ifaamas. org), 2020. p. 2193-2195.

KAVAK, H.; PADILLA, J. J.; LYNCH, C. J.; DIALLO, S. Y. Big data, agents, and machine learning: towards a data-driven agent-based modeling approach. In: **Proceedings of the Annual Simulation Symposium**. 2018. p. 1-12.

KELTON, W. D.; LAW, A. M. **Simulation modeling and analysis**. McGraw Hill Boston, 2000.

KIM, Y.; KOO, P. Effectiveness of Testing and Contact-Tracing to Counter COVID-19 Pandemic: Designed Experiments of Agent-Based Simulation. In: **Healthcare**. Multidisciplinary Digital Publishing Institute, 2021. p. 625.

KOVACIC, A.; PECEK, B. Use of simulation in a public administration process. **Simulation**, v. 83, n. 12, p. 851-861, 2007.

LEE, S. M.; RAVINDER, U.; JOHNSTON, J. C. Developing an agent model of human performance in air traffic control operations using apex cognitive architecture. In: **Proceedings of the Winter Simulation Conference, 2005**, p. 9, 2005.

LIAO, Y.; DESCHAMPS, F.; LOURES, E. D. F. R.; RAMOS, L. F. P. Past, present and future of Industry 4.0-a systematic literature review and research agenda proposal. **International journal of production research**, v. 55, n. 12, p. 3609-3629, 2017.

LU, L.; GAO, X.; DIETIKER, J. F.; SHAHNAM, M.; ROGERS, W. A. Machine Learning Accelerated Discrete Element Modeling of Granular Flows. **Chemical Engineering Science**, p. 116832, 2021.

MACAL, C. M. To agent-based simulation from system dynamics. In: **Proceedings of the 2010 Winter Simulation Conference**. IEEE, 2010. p. 371-382.

MACAL, C. M.; NORTH, M. J. Agent-based modeling and simulation. In: **Proceedings of the 2009 Winter Simulation Conference (WSC)**. IEEE, 2009. p. 86-98.

MARTINS, A. L. P.; PEREIRA, A. B. M.; LIMA, Y. T.; PINHO, A. F.; MONTEVECHI, J. A. B. Integrando Inteligência Artificial no Software AnyLogic® utilizando Pathmind®. **LII Simpósio Brasileiro de Pesquisa Operacional – SBPO. João Pessoa-PB, 3 a 5 de novembro de 2021**. v. 53, 2021 – 139266.

MELÃO, N., PIDD, M. Using component technology to develop a simulation library for business process modeling. **European Journal of Operational Research**, 172: 163–178, 2006.

MIGUEL, P. A. C.; FLEURY, A.; MELLO, C. H. P.; NAKANO, D. N.; TURRIONI, J. B.; HO, L. L.; MORABITO, R.; MARTINS, R. A.; PUREZA, V. Metodologia de pesquisa em engenharia de produção e gestão de operações. **Rio de Janeiro: Elzevir**, 2010.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of machine learning**. MIT press, 2018.

MONTEVECHI, J. A. B.; LEAL, F.; PINHO, A. F.; COSTA, R. F. S.; OLIVEIRA, M. L. M.; SILVA, A. L. F. Conceptual modeling in simulation projects by mean adapted IDEF: an application in a Brazilian tech company. In: **Proceedings of the Winter Simulation Conference**, Baltimore,USA, 2010

MONTEVECHI, J. A. B.; PINHO, A. F. DE, LEAL, F, MARINS, F. A. S. Application of design of experiments on the simulation of a process in an automotive industry. In: **Proceedings of the Winter Simulation Conference**, Washington, DC, USA, p. 1601-1609, 2007.

MORIYAMA, K.; KUROGI, Y.; MUTOH, A.; MATSUI, T.; INUZUKA, N. Running Reinforcement Learning Agents on GPU for Many Simulations of Two-Person Simultaneous Games. In: **2019 IEEE International Conference on Agents (ICA)**. IEEE, 2019. p. 50-55.

MUÑOZ, S.; IGLESIAS, C. A. An agent based simulation system for analyzing stress regulation policies at the workplace. **Journal of Computational Science**, v. 51, p. 101326, 2021.

NARDIN, L. G.; SZÉKELY, Á.; ANDRIGHETTO, G. GLODERS-S: a simulator for agent-based models of criminal organisations. **Trends in organized crime**, v. 20, n. 1, p. 85-99, 2017.

NEGAHBAN, A.; YILMAZ, L. Agent-based simulation applications in marketing research: an integrated review. **Journal of Simulation**, v. 8, n. 2, p. 129-142, 2014.

NISHI, A.; DEWEY, G.; ENDO, A.; NEMAN, S.; IWAMOTO, S. K.; NI; M. Y.; TSUGAWA, Y.; IOSIFIDIS, G.; SMITH, J. D.; Young, S. D. Network interventions for managing the COVID-19 pandemic and sustaining economy. **Proceedings of the National Academy of Sciences**, v. 117, n. 48, p. 30285-30294, 2020.

OLAVE-ROJAS, D.; NICKEL, S. Modeling a pre-hospital emergency medical service using hybrid simulation and a machine learning approach. **Simulation Modelling Practice and Theory**, v. 109, p. 102302, 2021.

OZIK, J.; COLLIER, N.; HEILAND, R.; AN, G.; MACKLIN, P. Learning-accelerated discovery of immune-tumour interactions. **Molecular systems design & engineering**, v. 4, n. 4, p. 747-760, 2019.

PATHMIND. **Simulation Optimization | Add AI Simulation Models | Pathmind**, 2021. Disponível em <https://pathmind.com/> Acesso em: 2 de agosto de 2021.

PECEK, B., KOVACIC, A. Business process management: use of simulation in the public sector**. Economic Research-Ekonomska Istraživanja**, 24(1): 95-106, 2011.

PERALES, D. P.; VALERO, F. A.; GARCÍA, A. B. Industry 4.0: a classification scheme. **Closing the gap between practice and research in industrial engineering**, p. 343-350, 2018.

PINCIROLI, L.; BARALDI, P.; COMPARE, M.; ESMAEILZADEH, S.; FARHAN, M.; GÖHRE, B.; GRUGNI, R.; MANCA, L.; ZIO, E. Agent-based modeling and reinforcement learning for optimizing energy systems operation and maintenance: the Pathmind solution. In: **Proceedings of the 30th European Safety and Reliability Conference and the 15th Probabilistic Safety Assessment and Management Conference**. Research Publishing Services, 2020. p. 1476-1480.

PRASANNA, A.; HOLZHAUER, S.; KREBS, F. Overview of machine learning and data-driven methods in agent-based modeling of energy markets. **INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik–Informatik für Gesellschaft**, 2019.

PROJECT BONSAI. **Training AI-agents with Microsoft Project Bonsai**, 2022. Disponível em < https://www.anylogic.com/features/artificial-intelligence/microsoft-bonsai/> Acesso em: 8 de março de 2022.

Py4J. **A Bridge between Python and Java**, 2022. Disponível em <https://www.py4j.org/> Acesso em: 8 de março de 2022.

RAILSBACK, S. F.; LYTINEN, S. L.; JACKSON, K. Agent-based simulation platforms: Review and development recommendations. **Simulation**, v. 82, n. 9, p. 609-623, 2006.

RAND, W. Machine learning meets agent-based modeling: When not to go to a bar. In: **Conference on Social Agents: Results and Prospects**. 2006.

ROBINSON, S. Choosing the right model: Conceptual modeling for simulation. **Proceedings of the 2011 Winter Simulation Conference (WSC)**, p. 1423–1435, 2011.

RODRIGUES, R. P.; DE PINHO, A. F.; SENA, D. Application of Hybrid Simulation in production scheduling in job shop systems. **SIMULATION**, v. 96, n. 3, p. 253-268, 2020.

ROSS, W.; ULIERU, M.; GOROD, A. A multi-paradigm modelling & simulation approach for system of systems engineering: A case study. In: **2014 9th International Conference on System of Systems Engineering (SOSE)**. IEEE, 2014. p. 183-188.

RÖßLER, M.; WASTIAN, M.; JELLEN, A.; FRISCH, S.; WEINBERGER, D.; HUNGERLÄNDER, P.; BICHER, M.; POPPER, N. Simulation and optimization of traction unit circulations. In: **2020 Winter Simulation Conference (WSC)**. IEEE, 2020. p. 90-101.

RUHANG, X.; ZHILIN, L. Identifying housing market expectation transformation: an agent-based housing market testbed. **International Journal of Business Continuity and Risk Management**, v. 9, n. 2, p. 117-152, 2019.

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

SENA, D. C.; MIRANDA, R. C.; PINHO, A. F.; MONTEVECHI, J. A. B.; SILVA, E. M. M. Política de compra de medicamentos utilizando simulação híbrida e aprendizado por reforço. **XLIX Simpósio Brasileiro de Pesquisa Operacional–SBPO. Blumenau-SC, 27 a 30 de agosto de 2017**. p. 3245-3256.

SHANNON, R. E. "Introduction to the art and science of simulation." **Winter Simulation Conference. Proceedings**. Vol. 1, p. 7-14. IEEE, 1998.

SIEBERS, P.O.; MACAL, C.M.; GARNETT, J.; BUSTON, D.; PIDD, M. et al. Discrete-event simulation is dead, long live agent-based simulation!. **Journal of Simulation**, v. 4, n. 3, p. 204-210, 2010.

SILVA, P. C.; BATISTA, P. V.; LIMA, H. S.; ALVES, M. A.; GUIMARÃES, F. G.; SILVA, R. C. COVID-ABS: An agent-based model of COVID-19 epidemic to simulate health and economic effects of social distancing interventions. **Chaos, Solitons & Fractals**, v. 139, p. 110088, 2020.

SONG, Il-Y.; ZHU, Y. Big data and data science: opportunities and challenges of iSchools. **Journal of Data and Information Science**, v. 2, n. 3, p. 1-18, 2017.

SOUZA, J. G.; ALCAINA, J. M.; MADELLA, M. Archaeological expansions in tropical South America during the late Holocene: Assessing the role of demic diffusion. **PloS one**, v. 15, n. 4, p. e0232367, 2020.

SUMARI, S.; IBRAHIM, R.; ZAKARIA, N. H.; ABHAMID, A. H. Comparing three simulation model using taxonomy: System dynamic simulation, discrete event simulation and agent based simulation. **International Journal of Management Excellence**, v. 1, n. 3, p. 54-59, 2013.

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

TIZHOOSH, H. R. Opposition-based reinforcement learning. **Journal of Advanced Computational Intelligence and Intelligent Informatics**, v. 10, n. 3, 2006.

VERMEULEN, A. F. **Industrial machine learning: using artificial intelligence as a transformational disruptor**. Apress, 2020.

WAKELAND, W. W.; GALLAHER, E. J.; MACOVSKY, L. M.; AKTIPIS, A. A. A comparison of system dynamics and agent-based simulation applied to the study of cellular receptor dynamics. In: **37th Annual Hawaii International Conference on System Sciences, 2004. Proceedings of the**. IEEE, 2004. p. 10 pp.

WATKINS, C. J. C. H. **Learning from delayed rewards**. 1989.

WHITE, K. P.; INGALLS, R. G. The basics of simulation. In: **2018 Winter Simulation Conference (WSC)**. IEEE, p. 147-161, 2018.

WUEST, T.; WEIMER, D.; IRGENS, C.; THOBEN, K. D. Machine learning in manufacturing: advantages, challenges, and applications. **Production & Manufacturing Research**, v. 4, n. 1, p. 23-45, 2016.

XIANG, W.; LEE, H. P. Ant colony intelligence in multi-agent dynamic manufacturing scheduling. **Engineering Applications of Artificial Intelligence**, v. 21, n. 1, p. 73-85, 2008.

XU, L. D.; XU, E. L.; LI, L. Industry 4.0: state of the art and future trends. **International Journal of Production Research**, v. 56, n. 8, p. 2941-2962, 2018.

YANG, C.; WENG, Y.; HUANG, B.; IKBAL, M. A. Development and Optimization of CAD System based on Big Data Technology. **Computer-Aided Design and Aplications**. p. 112-133, 2022.

ZHANG, H.; VOROBEYCHIK, Y.; LETCHFORD, J.; LAKKARAJU, K. Data-driven agent-based modeling, with application to rooftop solar adoption. **Autonomous Agents and Multi-Agent Systems**, v. 30, n. 6, p. 1023-1049, 2016.

ZHANG, W.; VALENCIA, A.; CHANG, N. Synergistic Integration Between Machine Learning and Agent-Based Modeling: A Multidisciplinary Review. **IEEE Transactions on Neural Networks and Learning Systems**, 2021.

ZHOU, L.; PAN, S.; WANG, J.; VASILAKOS, A. V. Machine learning on big data: Opportunities and challenges. **Neurocomputing**, v. 237, p. 350-361, 2017.

ZHU, F.; YAO,Y.; TANG,W.; CHEN,D. A high performance framework for modeling and simulation of large-scale complex systems. **Future Generation Computer Systems**, v. 51, p. 132-141, 2015.

APÊNDICE A

Modo Aleatório			
Número do Experimento	Período		
	12h (43200s)	24h (86400s)	36h (129600s)
1	1859	3483	5491
2	1789	3585	5384
3	1773	3638	5444
4	1776	3540	5289
5	1746	3562	5388
6	1785	3575	5343
7	1822	3570	5406
8	1705	3685	5280
9	1843	3651	5494
10	1778	3680	5471
11	1791	3664	5334
12	1847	3636	5305
13	1862	3564	5487
14	1789	3566	5431
15	1833	3593	5315
16	1761	3499	5391
17	1750	3708	5304
18	1803	3638	5507
19	1821	3598	5410
20	1825	3654	5357
21	1808	3566	5415
22	1826	3432	5336
23	1860	3481	5510
24	1807	3646	5439
25	1793	3513	5369
26	1834	3628	5352

Modo Aleatório			
Número do Experimento	Período		
	12h (43200s)	24h (86400s)	36h (129600s)
27	1822	3565	5414
28	1753	3624	5473
29	1766	3641	5498
30	1791	3562	5404
31	1838	3549	5338
32	1889	3627	5399
33	1827	3611	5503
34	1756	3599	5416
35	1833	3648	5362
36	1752	3612	5305
37	1816	3579	5338
38	1707	3696	5398
39	1787	3491	5364
40	1788	3656	5362
41	1821	3573	5386
42	1815	3591	5419
43	1790	3662	5465
44	1798	3620	5615
45	1846	3487	5405
46	1802	3647	5388
47	1818	3649	5463
48	1745	3590	5376
49	1797	3589	5392
50	1778	3545	5349
51	1857	3644	5349

Modo Aleatório			
Número do Experimento	Período		
	12h (43200s)	24h (86400s)	36h (129600s)
52	1818	3631	5296
53	1741	3672	5395
54	1798	3572	5279
55	1787	3571	5374
56	1845	3734	5480
57	1808	3589	5316
58	1873	3643	5239
59	1787	3569	5456
60	1742	3666	5253
61	1756	3620	5438
62	1828	3632	5530
63	1862	3685	5479
64	1738	3683	5439
65	1780	3582	5400
66	1768	3523	5321
67	1822	3656	5292
68	1763	3534	5580
69	1803	3670	5501
70	1790	3537	5387
71	1753	3637	5335
72	1728	3576	5458
73	1743	3613	5436
74	1905	3475	5190
75	1784	3505	5462
76	1866	3690	5405

Modo Aleatório			
Número do Experimento	Período		
	12h (43200s)	24h (86400s)	36h (129600s)
77	1849	3572	5379
78	1849	3725	5440
79	1837	3596	5501
80	1821	3608	5486
81	1847	3495	5335
82	1811	3604	5423
83	1850	3557	5400
84	1829	3614	5448
85	1824	3517	5420
86	1791	3695	5340
87	1823	3550	5421
88	1757	3625	5428
89	1772	3617	5561
90	1804	3517	5537
91	1822	3593	5513
92	1779	3634	5484
93	1794	3600	5370
94	1838	3556	5398
95	1800	3562	5363
96	1806	3537	5225
97	1712	3569	5471
98	1773	3643	5410
99	1774	3654	5404
100	1718	3538	5555
Total	**14400**	**28800**	**43200**
Média	**1800,46**	**3598,55**	**5404,86**
Porcentagem	**12,50%**	**12,49%**	**12,51%**

APÊNDICE B

```
/**
 * QLearning
 */
public class QLearning implements Serializable {

        Random rand = new Random();

    final double alpha = 0.9;
    final double gamma = 0.9;
    final int countLearningTimes = 10000;

    final int stateA = 0;
    final int stateB = 1;
    final int stateC = 2;
    final int stateD = 3;
    final int stateE = 4;

    final int[] states = new int[]{stateA,stateB,stateC,stateD,stateE};

    final int statesCount = states.length;

    int[] actionsFromStateA = new int[] { stateA, stateB, stateC, stateD, stateE};
    int[] actionsFromStateB = new int[] { stateA, stateB, stateC, stateD, stateE};
    int[] actionsFromStateC = new int[] { stateA, stateB, stateC, stateD, stateE};
    int[] actionsFromStateD = new int[] { stateA, stateB, stateC, stateD, stateE};
    int[] actionsFromStateE = new int[] { stateA, stateB, stateC, stateD, stateE};


    int[][] actions = new int[][] { actionsFromStateA, actionsFromStateB, actionsFromStateC,
actionsFromStateD, actionsFromStateE};



    String[] stateNames = new String[] { "VERMELHO", "VERDE", "AZUL", "BRANCO",
"PRETO"};

    double[][] Q = new double[statesCount][statesCount];
    int[][] R = new int[statesCount][statesCount];

    int current = -1;
    int currentState = -1;

    /**
     * Default constructor
     */
    //public QLearning() {
    //     init();
    //}

    public QLearning(String[] cores) {
```

```
        stateNames = cores;
        init();
    }


    public void init() {

        for (int i = 0; i < statesCount; i++) {
            int state = states[i];
            int[] actionsFromState = actions[state];

                for (int j = 0; j < actionsFromState.length; j++) {

                    int reward = -25;

                    if (state == actionsFromState[j]) reward = 100;

                    setQ(state, actionsFromState[j], reward);
                    setR(state, actionsFromState[j], reward);

                }
        }

    }

/*      public static void main(String[] args) {

        long BEGIN = System.currentTimeMillis();

        Qlearning obj = new Qlearning();

        obj.run();
        obj.printResult();

        long END = System.currentTimeMillis();
        System.out.println("Time: " + (END - BEGIN) / 1000.0 + " sec.");
    }
*/

    public int getState() {
        return currentState;
    }

    void run() {
        learn(countLearningTimes);

        while(current != getState()) {
            current = getState();
            runOnce();
        }
    }
```

```
    public void runOnce(){
        int bestAction = bestAction(currentState);
        currentState = bestAction;
    }

    void printResult() {

        for (int i = 0; i < statesCount; i++) {
            System.out.print("A CAIXA " + stateNames[i] + " TEM AS SEGUINTES
POSSIBILIDADES DE PALPITES:  \n");
            double maxValueQState = Double.MIN_VALUE;

            for (int j = 0; j < Q[i].length; j++) {
                double value = Q[i][j];

                if (value > maxValueQState) maxValueQState = value;

                if (value > 0) System.out.print(stateNames[j] + " " + value+ "\n");
            }

            System.out.print("O PALPITE CORRETO PARA ACERTAR A COR DA CAIXA " +
stateNames[i] +" TEM VALOR DE " + maxValueQState + "\n");
            System.out.println();
        }
    }

    //QLearning
    //region

    double maxQ(int s) {
        int[] actionsFromState = actions[s];
        double maxValue = Double.MIN_VALUE;
        for (int i = 0; i < actionsFromState.length; i++) {
            int nextState = actionsFromState[i];
            double value = Q[s][nextState];

            if (value > maxValue)
                maxValue = value;
        }
        return maxValue;
    }

    double Q(int s, int a) {
        return Q[s][a];
    }

    void setQ(int s, int a, double value) {
        Q[s][a] = value;
    }

    int R(int s, int a) {
        return R[s][a];
```

```
}

void setR(int s, int a, int value) {
   R[s][a] = value;
}

public void learn (int countLearningTimes) {
   for (int i = 0; i < countLearningTimes; i++) { // train episodes
      currentState = states[rand.nextInt(states.length)];
      step();
   }
}

public void step() {
   int state = currentState;

   // Select one among all possible actions for the current state
   int[] actionsFromState = actions[state];

   // Selection strategy is random in this example
   int index = rand.nextInt(actionsFromState.length);
   int action = actionsFromState[index];

   // Using this possible action, consider to go to the next state
   double q = Q(state, action);
   double maxQ = maxQ(action);
   int r = R(state, action);

   double value = q + alpha * (r + gamma * maxQ - q);
   setQ(state, action, value);
}

public int bestAction(int state){
   double [] qValues = Q[state];

   int bestAction = -1;

   for (int action = 0; action < qValues.length; action++) {

         if (bestAction < 0){
            bestAction = action;
         } else if ((qValues[action] == qValues[bestAction]) && (Math.random()>0.5)){
            bestAction = action;
         } else if (qValues[action] > qValues[bestAction]){
            bestAction = action;
         }
      }
   return bestAction;
}

//endregion
```

```
    @Override
        public String toString() {
                return super.toString();
        }

        /**
         * This number is here for model snapshot storing purpose<br>
         * It needs to be changed when this class gets changed
         */
        private static final long serialVersionUID = 1L;

}
```