UNIVERSIDADE FEDERAL DE PERNAMBUCO

CENTRO DE TECNOLOGIA E GEOCIÊNCIAS

CURSO DE ENGENHARIA ELETRÔNICA - BACHARELADO

ELIZA THAIS NEVES AMANCIO

USO DO ALGORITMO Q-*LEARNING* PARA CONSTRUÇÃO DE COREOGRAFIA DE POLE *SPORT*

RECIFE

2023

ELIZA THAIS NEVES AMANCIO

USO DO ALGORITMO Q-*LEARNING* PARA CONSTRUÇÃO DE COREOGRAFIA DE POLE *SPORT*

Trabalho de conclusão de curso apresentado ao Curso de Engenharia Eletrônica - Bacharelado da Universidade Federal de Pernambuco, como requisito parcial para obtenção do título de bacharel em engenharia eletrônica.

Orientador (a): Daniel de Filgueiras Gomes

RECIFE

2023

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Amancio, Eliza Thais Neves .
USO DO ALGORITMO Q-LEARNING PARA CONSTRUÇÃO DE COREOGRAFIA DE POLE SPORT / Eliza Thais Neves Amancio. - Recife, 2023.
57 : il., tab.

Orientador(a): Daniel de Filgueiras Gomes
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Eletrônica - Bacharelado, 2023.

1. Q-Learning. 2. Aprendizagem por Reforço. 3. Otimização. 4. Pole Sport. I. Gomes, Daniel de Filgueiras. (Orientação). II. Título.

620 CDD (22.ed.)

Universidade Federal de Pernambuco
Departamento de Engenharia Eletrônica
Centro de Tecnologia e Geociências

AVALIAÇÃO DE TRABALHO DE CONCLUSÃO DE CURSO

Título do Trabalho
Uso do algoritmo q-learning para construção de coreografia de pole sport

Aluno (a): **Eliza Thais Neves Amancio**
Orientador (a): **Daniel de Filgueiras Gomes**
Examinador 1: **José Sampaio de Lemos Neto**

	Orientador(a)	Examinador 1	Média Final
Nota	9,0	9,0	9,0

A banca considerou o (a) aluno (a):
(x) Aprovado (a)
( ) Aprovado (a) com restrições (modificações no verso desta folha)
( ) Reprovado (a)

Com a média final: 9,0

Recife, 1 de Fevereiro de 2023.

gov.br Documento assinado digitalmente
DANIEL DE FILGUEIRAS GOMES
Data: 13/04/2023 11:20:57-0300
Verifique em https://validar.iti.gov.br

Assinatura do (a) Orientador (a)

gov.br Documento assinado digitalmente
JOSE SAMPAIO DE LEMOS NETO
Data: 13/04/2023 21:21:07-0300
Verifique em https://validar.iti.gov.br

Assinatura do Examinador 1

gov.br Documento assinado digitalmente
ELIZA THAIS NEVES AMANCIO
Data: 17/04/2023 17:01:36-0300
Verifique em https://validar.iti.gov.br

Assinatura do (a) Aluno (a)

“THE PURPOSE OF THINKING IS TO LET THE IDEAS DIE INSTEAD OF US DYING”.

— ALFRED NORTH WHITEHEAD

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE GRÁFICOS

SÚMARIO

RESUMO

A aprendizagem por reforço é um ramo da inteligência artificial que visa solucionar problemas do tipo ‘aprender o que fazer em cada situação’ a partir de uma base de dados composta por estados, ações e recompensas. Para compreendermos suas características, utilizamos principalmente os trabalhos de Sutton e Barto (2014-2015) e Lapan (2020). Como na maioria das vezes a aprendizagem por reforço é utilizada para sair do ponto A e chegar no ponto B obtendo as melhores recompensas, percebemos que ela poderia ser útil para facilitar a criação de sequências de movimentos para campeonatos esportivos uma vez que esses campeonatos também se baseiam em recompensas. Para validar a nossa hipótese, escolhemos as regras do campeonato mundial de pole *sport* (IPSF, 2018) para testar o algoritmo *Q-Learning* e verificar se ele poderia gerar coreografias de diferentes níveis de dificuldade. Em nossos testes, encontramos um modelo de algoritmo que nos permite explorar diversas novas sequências ao selecionar ora os movimentos mais simples, ora os mais difíceis, de forma parametrizada, a partir da proposição de Amunategui (2017). Assim percebemos que tal estrutura pode servir não só para o pole *sport*, mas para outros esportes que planejam sequências de movimentos coreografados e recebem pontuação por cada um dos movimentos.

Palavras-chave: *Q-Learning*, Aprendizagem por Reforço, Otimização, *Pole Sport*

ABSTRACT

Reinforcement learning is a branch of artificial intelligence that aims to solve problems such as 'learning what to do in each situation' from a database composed of states, actions and rewards. To understand its characteristics, we mainly used the works of Sutton and Barto (2014-2015) and Lapan (2020). As most of the time reinforcement learning is used to leave point A and arrive at point B obtaining the best rewards, we realized that it could be useful to facilitate the creation of sequences of movements for sports championships since these championships are also based on rewards. To validate our hypothesis, we chose the rules of the pole sport world championship (IPSF, 2018) to test the Q-Learning algorithm and verify if it could generate choreographies of different levels of difficulty. In our tests, we found an algorithm model that allows us to explore several new sequences by selecting sometimes the simplest movements, sometimes the most difficult ones, in a parameterized way, from the proposition of Amunategui (2017). Thus, it seems that such a structure can serve not only for pole sport, but for other sports that plan sequences of choreographed movements and receive points for each of the movements.

Keywords: Q-Learning, Reinforcement Learning, Optimization, Pole Sport

1 INTRODUÇÃO

O cérebro, uma das máquinas mais intrigantes, cujas funções ainda não conhecemos plenamente, é capaz de processar uma grande quantidade de informações internas e externas ao corpo humano, desenvolvendo regras de identificação e ação-resposta a partir de experiências vividas. É por meio dos neurônios, as estruturas básicas na formação do sistema nervoso, que se dá a condução de impulsos nervosos, transmitindo e processando informações. Tal conjunto de neurônios interconectados forma uma rede neural.

> diversos pesquisadores tentaram simular o funcionamento do cérebro, principalmente o processo de aprendizagem por experiência, a fim de criar sistemas inteligentes capazes de realizar tarefas como classificação, reconhecimento de padrões, processamento de imagens, entre outras atividades. Como resultado destas pesquisas surgiu o modelo do neurônio artificial e posteriormente um sistema com vários neurônios interconectados, a chamada Rede Neural [artificial] (DATA SCIENCE ACADEMY, 2022, s.p.).

Em 1943, no artigo intitulado *A Logical Calculus of the Ideas Immanent in Nervous Activity*, o neurofisiologista Warren McCulloch e o matemático Walter Pitts reconheceram que os eventos neurais e suas relações poderiam ser estudados por meio de uma lógica proposicional[1]. Com isso, eles perceberam que qualquer proposição lógica pode ser descrita em termos de uma rede de neurônios conectados. Essa foi a primeira identificação da possibilidade de criação de uma rede neural artificial.

> Em 1949, Donald Hebb escreveu *The Organization of Behavior*, uma obra que apontou o fato de que os caminhos neurais são fortalecidos cada vez que são usados, um conceito fundamentalmente essencial para a maneira como os humanos aprendem. Se dois nervos dispararem ao mesmo tempo, argumentou, a conexão entre eles é melhorada (DATA SCIENCE ACADEMY, 2022, s.p.).

Paralelamente, ainda na década de 1940 - no mundo marcado pela II Guerra Mundial -, crescia a necessidade de produção de artefatos para análise balística, quebra de códigos e máquinas de cálculos capazes de suportar o projeto de uma bomba atômica (LOSANO; JOFFILY, 1992). Tais necessidades impulsionaram a evolução tecnológica que deu origem aos primeiros projetos de construção de computadores, quando, em janeiro de 1944, surgiu na

[1] A forma mais simples de lógica refere-se à possibilidade de determinar uma proposição como verdadeira ou falsa a partir de outras proposições, que podem ser verdadeiras ou falsas. Também chamada de lógica sentencial, é uma forma matemática e lógica de representação do conhecimento que não leva em consideração qualidade ou quantidade, e sim transforma sentenças em valores V ou F que, por exemplo, comporiam uma tabela verdade. Para saber mais: https://www.javatpoint.com/propositional-logic-in-artificial-intelligence#:~:text=Propositional%20logic%20(PL)%20is%20the,in%20logical%20and%20mathematical%20form.

Inglaterra o Colossus, o primeiro computador, projetado por um grupo liderado por Tommy Flowers para fazer a criptoanálise de códigos utilizados pelos nazistas. Nessa época, os computadores não compreendiam máquinas de função geral, mas sim de propósito dedicado, até a criação do ENIAC (do inglês Electronic Numerical Integrator and Computer), o primeiro computador de propósito geral. O ENIAC possuía elementos de controle, memória e aritmética; bem como a capacidade de executar operações matemáticas em velocidade bastante superior à de seus antecessores, sendo considerado um computador de propósito geral por sua capacidade de comportar uma variedade de domínios de aplicação (linguagem de propósito geral) (KLEINA, 2020).

O desenvolvimento de tais computadores, naturalmente, incitou ainda mais o imaginário popular para a ideia de máquinas que pudessem replicar comportamentos humanos. Foi assim que, em 1950, o matemático Claude E. Shannon (hoje conhecido como o pai fundador da comunicação eletrônica e da Teoria de informação) publicou junto com Weaver um artigo no qual descreveu os passos para programar um computador para jogar xadrez (SHANNON, WEAVER 1969). Apesar de declarar, em seu artigo, que o trabalho poderia não possuir tanta relevância prática, suas teorizações foram o pontapé inicial para o desenvolvimento de máquinas de xadrez eficientes e cujos conhecimentos puderam ser aplicados na execução de outras tarefas de grande relevância.

Entretanto, o marco inicial da inteligência artificial como conhecemos hoje aconteceu somente em 1956, com a conferência de Dartmouth, ocasião em que se encontraram diversos pesquisadores interessados no campo (batizado na conferência) da ‘Inteligência Artificial’, cuja definição dada foi: “cada aspecto de aprendizado ou outra forma de inteligência pode ser descrita de forma tão precisa que uma máquina pode ser criada para simular isso” (MINSKY; PAPERT, 1988 *apud* PUCRS Online, 2020, s.p.).

Uma vez que o campo foi estabelecido, multiplicaram-se as tentativas de simular artificialmente as respostas lógicas da forma percebida no cérebro. John von Neumann, por exemplo, sugeriu o uso de tubos de vácuo e relés telegráficos para reproduzir funções simples de neurônios. Frank Rosenblatt, um neurobiologista, curioso acerca do funcionamento do olho das moscas, percebeu que grande parte do processo de decisão para fuga acontece nos olhos. Sua pesquisa resultou no *hardware* Perceptron, sendo a mais antiga rede neural. Entretanto, Marvin Minsky e Seymour Papert comprovaram, durante o período que ficou conhecido como “anos desiludidos”, que o Perceptron é um classificador linear (MINSKY; PAPERT, 1988 *apud* GOODFELLOW, BENGIO, COURVILLE, 2016).

> A teoria foi desenvolvida para um sistema nervoso hipotético, ou máquina: um Perceptron. O perceptron é desenhado para ilustrar algumas das propriedades fundamentais de sistemas inteligentes, no geral, sem se aprofundar em condições especiais e frequentemente desconhecidas que se aplicam a determinados organismos biológicos. A analogia entre o perceptron e os sistemas biológicos deve ser aparente ao leitor (ROSENBLATT, 1958, p. 387). [2]

Em 1959, foram desenvolvidos os modelos chamados Adaline e Madaline, por Bernard Widrow e Marcian Hoff, de Stanford. Adaline é uma unidade desenvolvida com o fim de reconhecer e prever padrões binários nas redes telefônicas, enquanto a Madaline seria uma rede multicamadas de várias unidades Adaline, formando uma rede neural capaz de eliminar ecos nas chamadas telefônicas. Essa tecnologia é usada até hoje (WIDROW, HOFF, 1960).

Houve uma época, porém, em que os estudos relacionados à inteligência artificial reduziram drasticamente, sendo chamado de Inverno da Inteligência Artificial. Ficou difícil conseguir investimento para as pesquisas, já que os resultados não correspondiam às expectativas criadas, assim como algumas limitações foram identificadas por Minsky e Papert, que demonstraram que só era possível resolver problemas lineares com Perceptrons, não sendo possível fazer a operação XOR. Essa limitação só poderia ser resolvida com redes mais complexas e o algoritmo de retropropagação 20 anos depois.

Esse período durou até 1982, quando as pesquisas de Hopfield renovaram o interesse na área, seguido, em 1986, de avanços na implementação de redes neurais de várias camadas com o algoritmo de *backpropagation*, capaz de distribuir informação de erro para toda a rede. Esse algoritmo abriu um novo campo de possibilidades para a resolução de problemas não lineares que viriam a ser fundamentais nas décadas seguintes.

Uma vez que se criou a possibilidade de redes neurais com mais camadas, passou-se a discutir como fazer com que essas camadas aprendessem. Primeiro, acreditava-se que era preciso fazer com que cada camada aprendesse separadamente, mas, já na década de 2000, começou-se a trabalhar sobre o termo aprendizagem profunda. Percebeu-se que, para que as redes neurais aprendessem em tempo hábil, era preciso uma grande quantidade de dados. Tendo isso, o algoritmo de aprendizagem profunda do Google para reconhecimento de padrões alcançou, em 2012, a capacidade de identificar gatos. Esse foi só o primeiro passo para o desenvolvimento das aprendizagens.

---

[2] The theory has been developed for a hypothetical nervous system, or machine, called a perceptron. The perceptron is designed to illustrate some of the fundamental properties of intelligent systems in general, without becoming too deeply enmeshed in the special, and frequently unknown,conditions which hold for particular biological organisms. The analogy between the perceptron and biological systems should be readily apparent to the reader

Os pesquisadores da aprendizagem de máquina tentaram criar maneiras para que as máquinas pudessem, a partir de bases de dados, aprender sozinhas. Nesse processo, estabeleceram-se três tipos principais de aprendizagem: aprendizagem supervisionada, aprendizagem não supervisionada e aprendizagem por reforço. A primeira é usada para classificar dados rotulados; Já a segunda, permite encontrar padrões em dados não rotulados; enquanto a última aprende a lidar com situações com base em recompensas atribuídas a cada estado e/ou ação.

Nesta pesquisa, escolhemos trabalhar com a aprendizagem por reforço por entendermos que é muito utilizada para resolver problemas de controle. Ao acompanharmos, durante nossa vida, diversos campeonatos esportivos, percebemos que alguns tipos de esportes, que possuem pontuação por cada movimento e transições específicas (como a ginástica rítmica e olímpica), se enquadrariam como um problema de otimização, no sentido de encontrar qual é a melhor coreografia para cada atleta. Ao apresentar critérios e restrições quanto às transições de movimento possíveis, escolhemos trabalhar com o Pole *sport*, por seguir as mesmas características dos demais citados e ainda não contar com tantas pesquisas quanto modalidades mais famosas.

A hipótese inicial foi de que se é possível definir a melhor sequência de ações com base em um sinal escalar de recompensa, então seria possível definir qual a melhor sequência de movimentos obrigatórios a serem escolhidos com base nas regras e pontuações descritos no código de pontos do Pole Sport. Isso porque as regras do campeonato de Pole já têm recompensas pré-definidas para cada movimento, o que poderia ajudar a definir o valor de cada estado. Uma vez que um atleta de pole sport precisa da melhor sequência para ganhar um campeonato, isso se mostra um problema de otimização, pois, dentre as possibilidades, o agente precisa ser controlado na direção da maior recompensa, sem usar uma busca exaustiva.

Esse projeto vem numa tentativa de facilitar a montagem e ampliar as possibilidades de sequências, contribuindo com a melhora do nível esportivo e artístico dos atletas, a fim de colaborar com o reconhecimento do pole como esporte olímpico, além de ser uma ferramenta para os profissionais no planejamento das aulas, permitindo montar sequências direcionadas por foco da aula e nível da turma ou aluno (a). Além disso, buscamos quebrar alguns paradigmas em torno de uma prática que traz bem-estar físico, mental e emocional, não só para mulheres adultas, mas também para os públicos masculino, idoso e infantil.

No capítulo 2, trataremos sobre o que é a aprendizagem por reforço, quais os seus elementos, seu funcionamento e a forma como possibilitou a resolução do problema de dimensionalidade. No capítulo 3, vou apresentar o pole *sport* e o código de pontos. No capítulo

4, demonstrarei o projeto desenvolvido para a navegação de um movimento para outro, com vistas de exemplificar como a técnica Q-learning pode vir a ser útil para a criação de coreografias.

2 APRENDIZAGEM POR REFORÇO

Situado entre supervisionada e não supervisionada, o paradigma da aprendizagem por reforço lida com a aprendizagem em problemas de tomada de decisão sequencial no qual há *feedback* limitado (WIERING; VAN OTTERLO, 2012), sendo bem mais focada em atingir um objetivo, aprendendo qual a melhor ação a ser tomada em cada situação por meio da interação agente e meio. Nesse sentido, o agente é aquele que decide qual ação será tomada e o meio é tudo ao redor e dentro dele, de onde ele pode extrair informações. Na história da aprendizagem por reforço, a ideia inicial é aprender por tentativa e erro, baseado em como as pessoas aprendem. Entretanto, a aprendizagem por tentativa e erro às vezes se confunde com aprendizagem supervisionada, por esta usar o erro (cálculo de distância entre resultado e o valor esperado) para atualizar todos os pesos da rede. Isso porque, na aprendizagem por tentativa e erro, há os aspectos seletivos - envolvendo a experimentação e teste das alternativas, observando as consequências e escolhendo com base nisso - e associativo - no qual cada situação está associada a alternativas específicas -, enquanto na aprendizagem supervisionada não temos o aspecto seletivo. Isso reflete na quantidade de *feedback* que o sistema recebe.

Em aprendizagem por reforço, há apenas uma informação disponível sobre *performance*, em forma de um único sinal escalar, ou seja, o sistema de *feedback* é avaliativo e não instrutivo. Usar esse sinal limitado de *feedback* traz a necessidade de colocar mais esforço em como usá-lo para avaliar e melhorar o comportamento durante a aprendizagem.

De acordo com Sutton e Barto (2014), em seu livro *Reinforcement Learning: An Introduction*,

> Os problemas de aprendizagem por reforço envolvem aprender o que fazer - como mapear situações em ações - de forma a maximizar um sinal numérico de recompensa. Essencialmente, são problemas *closed-loop* pois as ações do sistema de aprendizagem influenciam as próximas saídas. Além disso, não é dito ao aprendiz quais ações tomar, como em muitas formas de aprendizagem de máquina, mas ao contrário, ele tem que descobrir que ações levam a maiores recompensas, tentando-as (p. 2).

No começo de seu desenvolvimento, a aprendizagem por reforço demandava uma base de dados inicial com dados de jogos anteriores de especialistas para aprender as possíveis e melhores jogadas com eles e, a partir disso, ir aprendendo os melhores caminhos. Já os algoritmos mais recentes usam outras formas de pesquisar os possíveis retornos de cada ação, podendo antever qual o resultado das escolhas, mesmo sem nenhum histórico ou professor externo, apenas *self-play*.

Para que essa forma de aprendizagem por *self-play* seja aplicável, o ambiente deve seguir o padrão de um processo de decisão Markoviano, em que a decisão a ser tomada depende apenas do estado atual do ambiente, independente do histórico de como se chegou até ali. “Se usarmos st para denotar o estado no tempo t então para um sistema markoviano poderíamos ter Prob(st+1 = s’ | s1, s2,..., st) = Prob(st+1 = s’ | st)” (WILMOTT, 2019, p. 176).

Os problemas de aprendizagem por reforço, então, envolvem um agente aprendendo a interagir com um ambiente não mapeado, onde não se sabe o resultado de cada ação, de forma que o agente só pode aprender as consequências de suas ações na prática, via tentativa e erro. Para aprender as consequências de cada ação, o agente precisa tomar a ação e observar como isso afeta o ambiente.

A função valor mede se um agente está em uma boa ou má posição para tomar as próximas ações. Se as ações possíveis de serem tomadas a partir do estado atual prometem uma alta recompensa, a função de valor para esse estado é alta e vice-versa. Podem existir dois tipos de função-valor: a que se refere ao estado ou a que se refere a um par ação-estado. É a partir desta função valor que definiremos a política a ser seguida.

> Uma política é um conjunto de regras governando que ação tomar em cada estado. Essa política pode ser determinística. Ou talvez seja aleatória. Basicamente na aprendizagem por reforço queremos que a política maximize o valor a cada estado (WILMOTT, 2019, p. 179)

Precisamos, então, entender o que são ações e o que são estados. Segundo Sutton e Barto (2015), na aprendizagem por reforço, ações são escolhas feitas com base na avaliação dos estados que geram maior valor, não maior recompensa, de forma a trazer maior recompensa a longo prazo. “A única forma de um agente poder influenciar o sinal de recompensa é por meio de suas ações, que podem ter um efeito direto na recompensa, ou um efeito indireto através da mudança do estado do ambiente” (SUTTON; BARTO, 2015, p. 7). Nesse caso, a recompensa é o que você recebe ao tomar uma ação, podendo ser positiva ou negativa a depender da ação escolhida e do estado atual (WILMOTT, 2019), sendo o estado uma representação atual do ambiente.

> O ambiente é tudo que está fora de um agente. No sentido mais geral, é o resto do universo, mas isso vai um pouco além e excede a capacidade até mesmo dos computadores de amanhã, então usualmente nós seguimos o sentido geral aqui (LAPAN, 2020, local do Kindle 896).

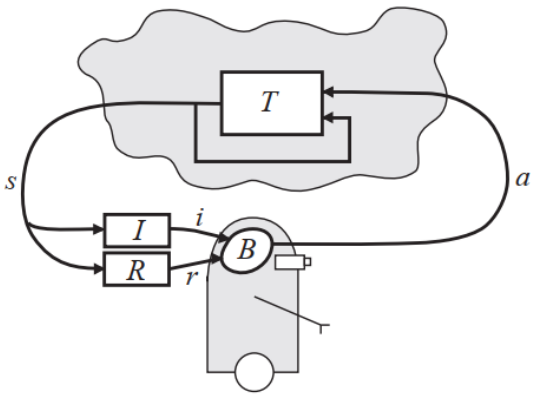
Tal ambiente refletirá as consequências das ações tomadas a cada passo, como afirma Lapan:

> Ações são coisas que o agente pode fazer no ambiente. Em AR, distinguimos entre dois tipos de ações - discretas e contínuas. Ações discretas formam um conjunto de coisas mutuamente exclusivas que um agente pode fazer, como se mover para a esquerda ou para a direita. Ações contínuas têm algum valor atrelado a elas, como a ação de girar o volante com o ângulo e direção do giro (LAPAN, 2020, Locais do Kindle 902-904).

Sendo assim, as três principais características da aprendizagem por reforço são "ser essencialmente *closed-loop,* não ter instruções diretas de que ações tomar e onde a consequência das ações, incluindo sinais de recompensa, ocorre após longos períodos de tempo" (SUTTON; BARTO, 2014, p. 2). A Figura 1 retrata o que compõe o modelo padrão desse modo de aprendizagem.

O agente B interage passo a passo com o ambiente T por meio de ações que podem alterar o estado atual do ambiente s. Assim como um valor de recompensa r, que vem para reforçar a escolha de ações que maximizem os sinais de recompensa ao longo do tempo. As ações, mudando a situação do estado, levam a novos *inputs* e uma recompensa equivalente. Quando o agente recebe qualquer reforço, ele é convidado a fazer uma nova escolha, que leva a um novo estado e recompensa, de forma que ao longo do tempo o agente comece a perceber quais sequências de ações levam a melhores recompensas.

Figura 1. Modelo padrão da aprendizagem por reforço



Fonte: KAELBLING; LITTMAN; MOORE, 1996, p. 238.

A partir da Figura 1, é possível perceber que a recompensa entra como a única forma de estimular ou desestimular uma ação, guiando o sistema para o comportamento esperado. Em alguns casos, como jogos nos quais há estados de ganho, perda ou empate bem definidos, pode ser fácil de especificar a função de recompensa, já em outros casos é preciso que o criador do

sistema tenha cuidado ao especificar recompensas por estados ou ações, e também com seu tamanho relativo para que o agente não encontre uma grande recompensa negativa antes de ganhar uma pequena recompensa positiva, fazendo com que a recompensa positiva seja ofuscada (KAELBLING; LITTMAN; MOORE, 1996).

Apesar de o objetivo da aprendizagem por reforço ser sempre o de conseguir o máximo possível de recompensas, há uma grande variedade de algoritmos que servem a situações diferentes. Primeiro, precisamos entender os binômios *model-based* / *model-free* e *on-policy* / *off-policy.*

Quando temos conhecimento total do comportamento esperado do sistema, em termos de pares de estado-ação possíveis e recompensas, temos o *model-based* - esses problemas são resolvidos fundamentalmente com programação dinâmica. Na falta dessas informações, o agente precisa interagir com o ambiente e coletar amostras explorando os pares de estado-ação disponíveis e estimar as recompensas, caracterizando o *model-free.*

Quanto aos termos *on-policy* e *off-policy*, a diferença está na forma como os dados da experiência do agente são coletados e integrados à política que é usada na escolha das ações. No primeiro caso, apenas experiências atuais são utilizadas para atualizar a política, enquanto no segundo caso, é criado um *buffer* que permite usar amostras obtidas em outras políticas para melhorar a política (KAELBLING; LITTMAN; MOORE, 1996).

O Quadro 1 traz um comparativo das técnicas utilizadas para resolver os problemas de aprendizagem por reforço, trazendo a aprendizagem por diferença temporal como uma combinação dos métodos de Monte Carlo e de programação dinâmica.

Quadro 1. Comparativo das técnicas

	Dynamic Programming	Monte Carlo Methods	TD Learning
Learns from experience, without knowing environment dynamics	X	✓	✓
Uses bootstrapping	✓	X	✓
Updates policies with partial episode trajectories		X	✓

Fonte: BILGIN, 2020, p. 158.

Quanto ao futuro da aprendizagem por reforço, Sutton aponta que "o Futuro da aprendizagem por reforço pode dar um passo a mais para se concentrar nas estruturas que permitem a estimativa da função de valor" (SUTTON *apud* WIERING; MARTIJN, 2012, Local do Kindle 59), ideia que conversa com o construtivismo como é conhecido na psicologia. E, realmente, muitos dos novos desenvolvimentos têm a ver com estruturas que permitem a aproximação da função de valor, assim como propriedades, capacidades e garantias sobre convergência e performance dessas novas estruturas. Frameworks Bayesianos, aproximações lineares eficientes, representação relacional do conhecimento e decomposições de natureza hierárquica e multi-agentes constituem as novas estruturas empregadas na metodologia de aprendizagem por reforço dos dias atuais.

Aprendizagem por reforço é, então, hoje, um campo estabelecido usualmente situado na área de aprendizagem de máquina, no entanto, seu foco em comportamento faz com que se conecte com outros campos de pesquisa como psicologia, pesquisa de operações, otimização matemática e muito mais. É tanto uma metodologia generalista para otimização de comportamento quanto um conjunto de ferramentas para tanto, o domínio de aplicação varia desde robótica e jogos de computador até roteamento em rede sistemas de linguagem natural

Competições podem promover compartilhamento de código e reuso, estabelecer padrões para o campo e ser usadas para avaliar e comparar métodos em domínios desafiadores. Grandes subcampos da aprendizagem por reforço em que muitos resultados foram publicados na última década incluem subcampos como: aprendizagem por reforço evolucionária, abordagens de representação de conhecimento relacional, abordagens hierárquicas, deixando que a piscina de autores reflita a jovialidade do campo e a natureza ativa do campo.

As competições de pole *sport* se parecem, em certa medida, com as competições de aprendizagem por reforço, pois é durante esses eventos que se fortalece o catálogo de movimentos e transições possíveis. Também existe um código compartilhado nas competições de pole *sport* e também existem padrões que precisam ser seguidos, como a altura, tipo e espessura do pole, tempo de apresentação, etc. Em competições desse tipo, os participantes acabam se desenvolvendo e evoluindo em suas práticas pelo simples fato de estarem reunidos.

Tendo em vista que uma coreografia é composta de uma sequência de ações, o problema é precisamente encontrar a melhor política para maximizar uma recompensa, diferente de um problema classificatório ou de busca de um padrão nos dados. Sendo assim, temos uma visão gamificada de uma situação e suas regras, na qual buscamos o comportamento ideal para vencer, o que possibilita enxergarmos a aprendizagem por reforço como a técnica de inteligência artificial ideal para a construção de coreografias de pole *sport*.

3 REGRAS DO POLE SPORT

O Pole Sport tem origem em uma antiga modalidade esportiva nascida no oeste da Índia, chamada *Mallakhamb* (*malla* = homem de força, ginasta, yogi e *khamb* = poste). Esse esporte hindu, que data do século XII, só teve a primeira competição mundial em 2019, depois de ser difundido, nos últimos anos, com eventos menores pela Federação de Mallakhamb da Índia, tendo como um dos objetivos ser reconhecido como esporte (NAG, 2022).

Alguns dos tipos de Mallakhamb são Pole Mallakhamb e Hanging Mallakhamb, modalidades praticadas exclusivamente por homens e garotos, e Rope Mallakhamb, que pode ser praticado por homens, mulheres, garotos e garotas. Essa prática consiste em acrobacias em um poste vertical de madeira, fixo ou suspenso, ou em uma corda, respectivamente. Em todas as categorias, a pontuação se diferencia pelo nível de dificuldade das entradas e saídas dos movimentos e elementos acrobáticos.

Assim como no Mallakhamb, o Pole Sport consiste na execução de sequências de movimentos e acrobacias coreografadas em uma barra vertical, lembrando também a ginástica artística. As roupas utilizadas também são parecidas: shorts e tops são utilizados, pois o contato da pele com o metal é que proporciona aderência para a execução dos movimentos (NAG, 2022).

Atualmente, as federações e associações de Pole Sport buscam, também, que o Comitê Olímpico Internacional reconheça a prática como esporte, assim como sua inclusão nas olimpíadas. Para poder encaixar-se nos padrões olímpicos, o pole *sport* vem tornando cada vez mais rígidas as regras de pontuação, a fim de se equiparar com outros esportes que já funcionam dessa maneira - o que, direta ou indiretamente, contribui também com a melhora do nível dos atletas e do nível de ensino do esporte.

Para universalizar as regras, criou-se uma espécie de manual, o código de pontos, que contém todas as regras e as pontuações atribuídas a cada movimento. De acordo com esse documento, a pontuação final do atleta é composta pela soma da pontuação dos movimentos obrigatórios, dos bônus técnicos da apresentação, das deduções técnicas da apresentação e da apresentação artística e coreografia (IPSF, 2018).

Todos os atletas devem realizar o número exigido de elementos obrigatórios aéreos, a não ser que se especifique o contrário. É concedida uma pontuação de 0.1 a 1.0 caso o atleta execute o elemento seguindo os requisitos mínimos, como segurar a postura por dois segundos, com visualização clara para os árbitros e com ângulo do corpo e aberturas conforme indicado no código. Elementos com valor 0.1 são mais fáceis que aqueles com valor 1.0 e bônus extras

são concedidos caso o atleta execute qualquer um dos bônus compulsórios sobre os elementos obrigatórios. O valor técnico permitido para os elementos obrigatórios depende da categoria de idade e divisão competitiva. Somente a primeira tentativa do elemento é julgada e elementos obrigatórios não podem ser repetidos, já que repetições resultam em dedução por forma incorreta e elemento faltante (IPSF, 2018).

Existem pontuações que podem ser previstas, como as dos movimentos obrigatórios, e outras que não, como os pontos subjetivos ou os bônus e as deduções que só podem ser conferidos no momento da apresentação, pois se referem à performance do atleta no dia do campeonato. Uma das pontuações mais subjetivas é a referente à apresentação artística e coreografia. Nesse quesito, podem ser concedidos até 20 pontos, distribuídos entre os seguintes critérios: equilíbrio, confiança, fluidez, interpretação, originalidade dos elementos, originalidade das transações entre elementos, presença de palco e carisma, originalidade da coreografia e sua composição e originalidade do trabalho de solo. Também podem haver deduções nesse quesito, sendo deduzido um ponto a cada distração emitindo sons vocais, a cada defeito ou distração devido ao figurino, caso o atleta seque as mãos no figurino, corpo, pole ou chão ou caso ajuste o cabelo ou figurino durante a apresentação, além do caso da falta de início ou fim lógicos de apresentação, assim como deduções gerais no caso de determinados erros durante a apresentação. O sistema de pontos encoraja não só uma coreografia equilibrada, mas também um atleta equilibrado (IPSF, 2018).

Nessa seção, entraremos em detalhes nos critérios de pontuação e, consequentemente, de recompensa, detalhando os movimentos que são importantes para a utilização no algoritmo. Apesar de o código de pontos abranger também as regras para competições em dupla, nesse projeto focamos apenas em competições solo.

3.1 BÔNUS TÉCNICOS

É possível garantir um bônus técnico a partir de alguns elementos e combinações, dadas velocidades e alturas mínimas, entre outros critérios. Algumas dessas informações foram utilizadas na alimentação do nosso banco de dados, uma vez que o algoritmo deveria considerá-las nas interações. Pode-se ganhar no máximo 15 pontos.

Quadro 2. Descrição dos bônus técnicos

.

Sigla	Nome	Definição	Critérios	Pontuação
AC (Max 1.0)	Pegada Acrobática	Uma pegada acrobática é definida como a perda de todo o contato com o pole, quando nenhuma parte do corpo toca o pole por um breve instante, e "re-pega" o pole.	O atleta pode se mover para cima ou para baixo no pole e/ou mudar de direção/posição ao realizar a re-pegada. O atleta pode se mover para cima ou para baixo no pole e/ou mudar de direção/posição ao realizar a re-pegada.	+0.5
SP/SP	Combinação de giros com outros giros no pole giratório	Se refere a todos os giros que são combinados com um giro no pole giratório.	A posição de cada giro nesta combinação deve girar 720° completos antes de mudar para a posição do próximo giro (o qual deve incluir a mudança da posição do corpo).	+0.5
SP/SP/SP (Max 2.0)	Combinação de giros com outros giros no pole giratório	Três giros podem ser combinados para este bônus técnico.	Ao se combinar 3 giros de valor 1.0 cada, a posição de cada giro deve ficar fixa e girar por 720°;	+1.0
SP/ST	Combinação de giros com outros giros no pole estático	Refere-se a todos os giros que são combinados com outro giro no pole estático	Cada giro deve ter uma rotação mínima de 360° (e deve incluir a mudança da posição do corpo), e a transição de um para o outro deve ser direta.	+0.5
SP/SP/ST (Máx 2.0)		Três giros podem ser	a posição de cada	+1.0

		combinados para este bônus técnico. Ao se combinar 3 giros de valor 1.0 cada	giro deve ficar fixa e girar por 360°;	
SP/E/ST SP/SP/E/ST (Máx 1.5)	Combinação de giros com outros elementos no pole estático	Se refere a todos os outros elementos combinados com um giro no pole estático	O elemento deve ser mantido por um breve instante em uma posição fixa	+0.5
		Para esse bônus técnico, dois giros podem ser combinados, terminando com outro elemento mantido por um breve instante em uma posição fixa, no pole estático.	Cada giro deve ser mantido em uma posição fixa e girar por 360	+1.0
D	Quedas	Uma descida rápida e controlada no pole, que deve ter pelo menos 1 metro.	Não pode haver contato da mão com o pole durante a queda.	+0.2
DC (Máx 3.0)	Combinações dinâmicas no pole giratório ou estático	Combinações dinâmicas são as combinações de dois movimentos dinâmicos separados (vide definição de combinação dinâmica) onde o corpo tem um controle forte de impulso. O atleta deve demonstrar um alto nível de controle de força centrípeta onde pelo menos uma parte do corpo se solta do pole. Ex: movimentos dinâmicos podem incluir re-pegadas, quedas, pegadas	Eles devem ser executados em sucessão rápida, sem pausa entre os movimentos dinâmicos para que o atleta possa receber este bônus. A repetição de uma combinação dinâmica será concedida no máximo duas vezes. Também apenas 1 re-pegada pode ser incluída nas combinações dinâmicas	+0.5

		acrobáticas, saltos para fora, etc.		
JO (Máx 1.2)	Saltos para fora/ Saltos para o pole (Jump-out / Jump-on):	Salto para fora é um salto executado de uma posição inicial no pole, que termina no solo, longe do pole. Um salto para o pole é um salto executado de uma posição inicial no solo, longe do pole, que termina em uma posição no pole.	Não pode haver contato com o pole entre as posições inicial e final.	0.2
FO (Máx 1.0)	Flip para Fora/Flip para o pole (Flip-out / Flip-on)	Um flip para fora (flip-out) é um flip realizado em uma posição de início no pole que termina no solo. Um flip para o pole (flip-on) é um flip executado de uma posição no chão que termina em uma posição no pole.	Tanto os flips para fora quanto os para o pole devem girar totalmente, com o quadril passando por cima dos calcanhares e sem contato nem com o pole nem com o solo. Os flips podem ser realizados para frente, para trás ou de lado.	1.0
CF (Máx 1.0)	Flip com contato (Contact Flip)	Um flip no pole com contato é um flip realizado a partir de uma posição inicial no pole, que gira para trás, para frente ou de lado e que termina em uma posição no mesmo pole.	Um flip must deve girar completamente (quadril e pés passando acima da cabeça) de forma dinâmica enquanto mantém contato com o pole. Flips no pole com contato podem ser executados para a frente, para trás ou lateralmente.	+0.5

F (Máx 1.5)	Flip no Pole (começando e terminando no mesmo pole)	Um flip no pole é um flip realizado a partir de uma posição inicial no pole, que gira para trás, para frente ou de lado e que termina em uma posição no mesmo pole.	Flips no pole podem ser executados para a frente, para trás ou lateralmente	1.5
RG	Re-grips (Recaptura):	Recapturas são definidas como soltar a barra com uma mão ou com uma trava de braço por um breve momento e intencionalmente recapturar o pole com uma nova pegada de mão ou trava de braço. Ex: trava de cotovelo para pegada da mão. Para que uma recaptura seja reconhecida, o atleta não pode conseguir manter a posição do seu corpo quando solta a pegada. Não pode haver contato do corpo com o pole, apenas as posições de pegada	Para que uma recaptura seja reconhecida, o atleta não pode conseguir manter a posição do seu corpo quando solta a pegada. Não pode haver contato do corpo com o pole, apenas as posições de pegada.	+0.2

Fonte: elaborada pela autora.

3.2 DEDUÇÕES TÉCNICAS

Apesar de serem grandes responsáveis pela pontuação final, todas as deduções técnicas se referem a erros no momento da apresentação, como quedas, má execução ou apresentação dos elementos. Como o foco do projeto é a criação de uma coreografia prévia, não podemos

controlar o que acontece no meio da apresentação ou solicitar que o algoritmo resolva esses problemas.

3.3 DEDUÇÕES GERAIS

Quadro 3. Descrição das deduções gerais

Falha ao girar em pole estático	O atleta será penalizado por não girar no pole estático pelo menos uma vez durante a coreografia	O atleta deve ser capaz de realizar giros em ambos os poles tanto em posição invertida quanto de cabeça para cima	-2.0
Falha ao girar com impulso no pole giratório	O atleta será penalizado por não girar com impulso no pole giratório pelo menos uma vez durante a coreografia	O atleta deve ser capaz de realizar giros em ambos os poles tanto em posição invertida quanto de cabeça para cima	-2.0
Não usar os dois poles de maneira equilibrada	Atletas devem ser hábeis tanto no pole estático quanto no giratório	isso deve ser demonstrado pelo uso equilibrado dos dois poles, sem haver favorecimento de um ou outro pole.	-2.0
Usar menos do que a altura total do pole	Todos os atletas terão pontos deduzidos se não usarem a altura total do pole, por pelo menos duas (2) vezes (em qualquer um dos poles ou em ambos) durante a coreografia.	A altura total do pole refere-se ao uso desde sua base até não menos que 10cm abaixo do seu topo. Os atletas que pretendem usar a altura total do pole devem garantir que eles tenham contato com o pole ou alcancem qualquer parte do corpo até este ponto, com ou sem contato com o pole.	-2.0

Fonte: Elaborado pela autora.

3.4 APRESENTAÇÃO ARTÍSTICA E COREOGRAFIA

Dos bônus artísticos e coreográficos, foram selecionados apenas os que podemos acrescentar ao algoritmo com algum tipo de métrica. Como a originalidade conta, o algoritmo não pode criar a coreografia perfeita e resultar sempre nesta, pois para que ele seja usado por diferentes pessoas é necessário que ele resulte em coreografias diferentes para cada um.

De acordo com o código, o atleta pontua zero caso os juízes considerem que a criatividade estava presente em menos de 20% da coreografia; 0.5 pontos caso esteja entre 20% e 40%; 1.0 ponto entre 40% e 60%; 1.5 pontos para 60% a 80%; e 2.0 pontos para mais de 80% da coreografia. Elementos no pole e fora do pole também são analisados para essa contagem.

Quadro 4. Descrição dos bônus artísticos

Equilíbrio (Máx 2.0)	Equilíbrio se refere à habilidade de criar uma coreografia bem equilibrada que apresente quantidades iguais de diversos elementos.	O atleta deve criar uma coreografia equilibrada com diferentes tipos de truques, ex: flexibilidade, força, giros, movimentos dinâmicos, movimentos acrobáticos, trabalho de palco e transições, integrando todos em uma coreografia com presença de palco e conteúdo artístico, usando o pole giratório e o estático igualmente.
Originalidade da apresentação como um todo (Máx 2.0)	Se refere à originalidade e criatividade do desempenho global de truques e combinações, elementos e movimentos originais dentro e fora do pole e a originalidade coreográfica em toda a apresentação	O atleta deve criar combinações originais de truques e criar novos temas na coreografia. Os árbitros não estão à procura de apenas um ou dois truques e combinações exclusivos, mas de originalidade geral em todos os componentes da coreografia.
Originalidade dos elementos (Máx 2.0)	Se refere à originalidade e criatividade do desempenho global de truques , elementos e movimentos no pole em toda a apresentação.	Os atletas receberão dedução de pontos se os elementos, truques e movimentos se tornarem repetitivos (por exemplo, se as mesmas subidas, JO ou RG for usada repetidamente ao invés de

		uma variedade de elementos, truques e movimentos).
Originalidade das transições entrando e saindo dos elementos (Máx 2.0)	Se refere a originalidade de todas as transições, entradas e saídas de truques e combinações durante toda a coreografia	O atleta deve criar movimentos novos e originais para suas transições entrando e saindo de truques e entrando e saindo do pole

Fonte: Elaborado pela autora.

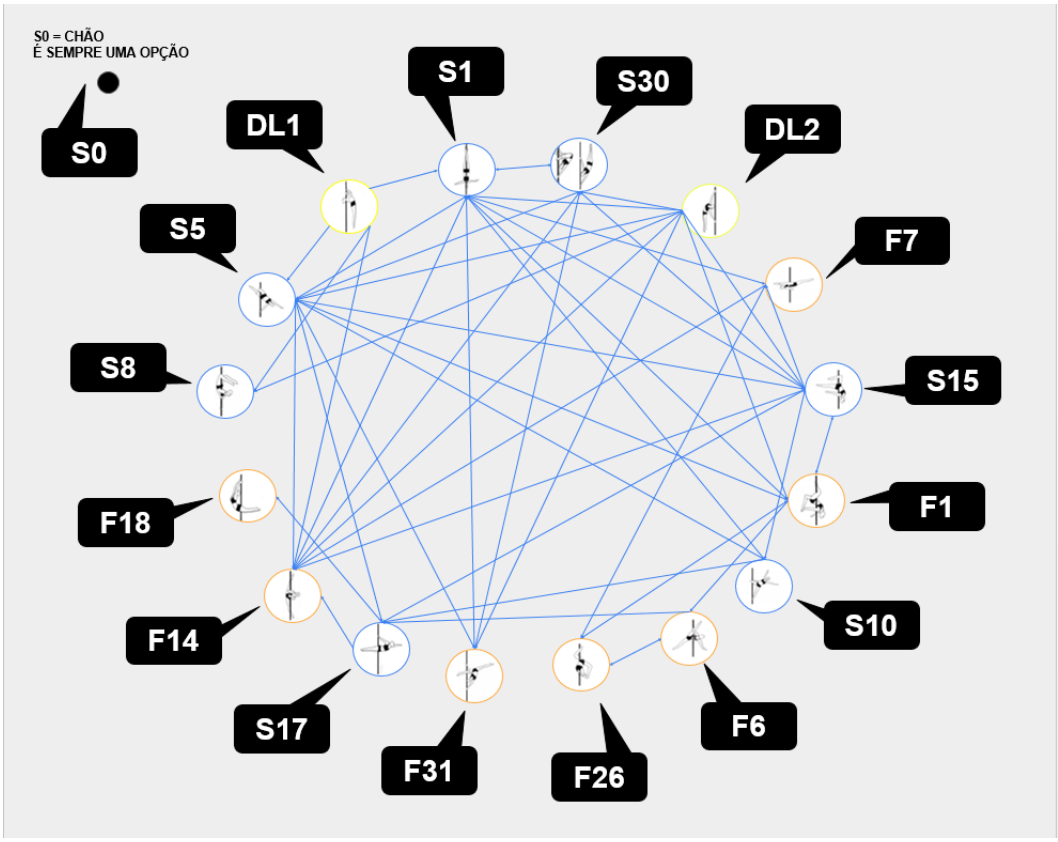
Para a competição, o atleta deve selecionar seus elementos obrigatórios. Cada nível pede uma quantidade específica de elementos, de variados tipos, no nível de dificuldade da divisão (Amador, profissional ou elite). A pontuação de cada elemento varia de 0.1 a 1.0, de acordo com o nível de dificuldade.

Na divisão Amador, os atletas devem escolher nove elementos de pontuação entre +0.1 e +0.5, sendo três elementos de flexibilidade, três elementos de força, um giro no pole giratório, um giro no pole estático e um *deadlift*. A pontuação total dos elementos obrigatórios de uma coreografia da divisão Amador deve estar entre 1.1 pontos e 4.5 pontos (IPSF, 2018).

Elementos de flexibilidade referem-se à flexibilidade das pernas, costas, quadril e ombros. Giros no pole estático são uma rotação de ao menos 360° mantendo o corpo na posição determinada. No pole giratório, a posição deve ser mantida por 720º. Os elementos de força demonstram força nos braços, pernas e core. *Deadlift* é a suspensão do corpo, sem impulso, movimento de balanço, ou ajuda do pole. Os atletas não podem escolher o mesmo elemento duas vezes, mesmo se executados em ângulos diferentes (IPSF, 2018).

Dentre os elementos descritos no código de pontos, escolhemos para esse projeto sete elementos de força (S), sete elementos de flexibilidade (F) e dois *deadlifts* (DL) que permitem uma gama de transições. Eles foram escolhidos pois possuem entre eles uma variedade de transições para que o algoritmo pudesse produzir coreografias variadas, além de reduzir o nosso banco de dados para a validação do algoritmo.

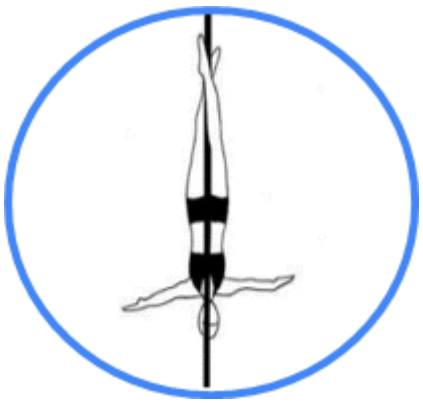
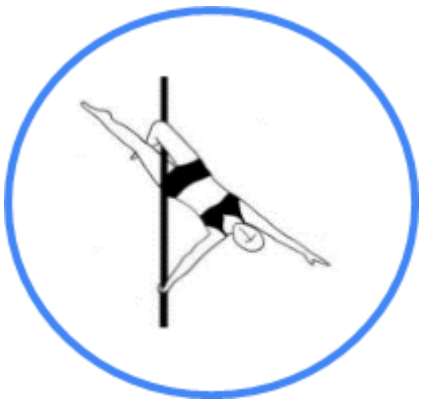
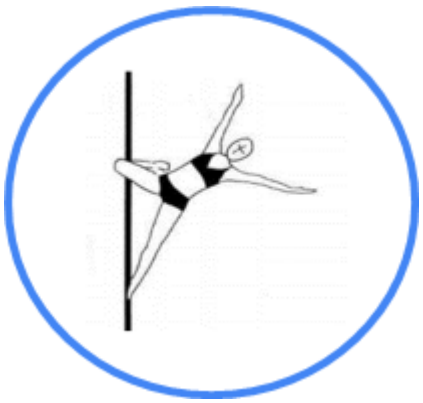
Figura 2. Grafo das transições possíveis



Fonte: Elaborada pela autora.

Da lista completa de elementos obrigatórios do código, foram escolhidos:

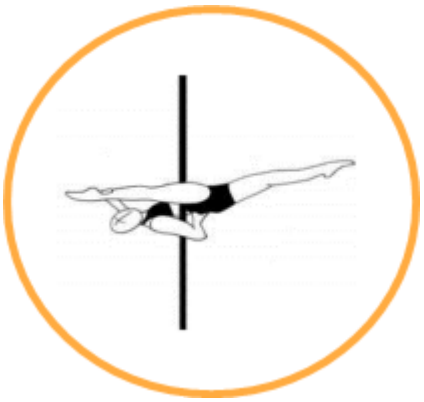
Quadro 5. Elementos de Força

S1 - Basic invert no hands (+0.1) 	*S5 - Outside knee hook, passé (+0.1)* 
S8 - Basic brass monkey (0.2) 	*S10 - Cupid (+0.2)* 

S15 - Outside knee hang back passé (+0.2) 	*S17 - Basic superman (+0.3)* 
S30 - Handspring straddle/pencil (+0.4) 	

Fonte: Elaborado pela autora.

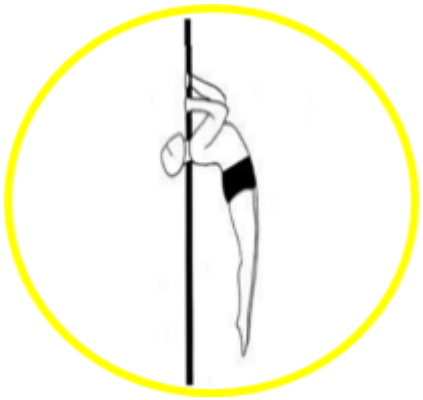
Quadro 6. Elementos de Flexibilidade

F1 - Inside leg hang (+0.1) 	*F6 Allegra split 1 (+0.3)* 
***F7** - Hip hold split (+0.3)* 	*F14 - Cross bow elbow hold (+0.4)* 

***F18** - Superman crescent (+0.4)* 	*F26 - Inside leg hang back split (+0.5)* 
F31 - Split grip leg through split (+0.5) 	

Fonte: Elaborado pela autora.

Quadro 7. Elementos do tipo *Deadlift*

DL1b - Deadlift from shoulder mount grip (+0.3)	*DL2b - Deadlift from twist grip (+0.3)*
	

Fonte: Elaborado pela autora.

3.5 PONTUAÇÃO POR DIFICULDADE

O atleta pode conquistar até 10 pontos pelo nível de dificuldade (LOD - *level of difficulty*) dos elementos e combinação de elementos escolhidos. São analisadas as subidas, os movimentos de flexibilidade, as transições, os giros e os elementos de força e a pontuação em cada quesito é a média da dificuldade dos elementos. A variação nessa pontuação é especialmente relevante para o projeto de otimização, uma vez que é preciso que a solução encontrada se adeque ao corpo do atleta em questão.

O código aponta 5 níveis de dificuldade:

Quadro 8. Níveis de dificuldade

Simples	Moderado	Difícil	Muito Difícil	Extremamente Difícil
0	0.5	1.0	1.5	2.0

Fonte: IPSF, 2018, p. 7.

Subidas: São contabilizadas na execução de ao menos 2 movimentos de subida em sequência (IPSF, 2018).

Movimentos de Flexibilidade - Se refere à flexibilidade das pernas, costas, quadril e ombros. Para pontuar o atleta precisa executar os movimentos, combinações e transições com máxima extensão e mobilidade (IPSF, 2018).

Transições no Pole - Se refere às transições para dentro e para fora dos elementos, incluindo movimentos que partem ou levam ao chão. Para máxima pontuação o atleta deve apresentar movimentos fluidos e posicionamento preciso do corpo (IPSF, 2018).

Giros - Na barra fixa, a posição do giro deve se manter durante uma rotação de ao menos 360º e na barra giratória por 720º (IPSF, 2018).

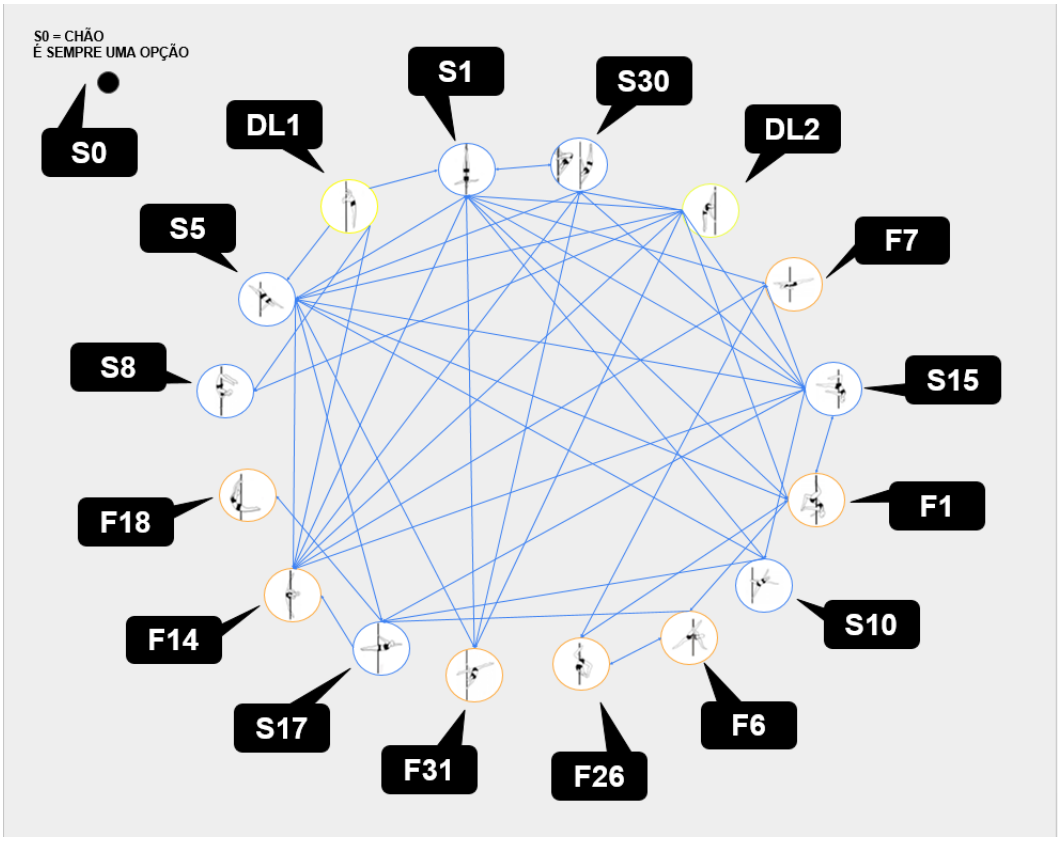
Elementos de Força - Para apresentar maior LOD para os juízes, o atleta deve executar elementos que demonstrem força nos braços, pernas e core. Segurar e controlar um movimento de força como a bandeira por, por exemplo, 2 segundos, resulta em uma maior pontuação por LOD (IPSF, 2018).

Tendo em mente as regras apresentadas, pode-se perceber que já existe um sistema de recompensas desenhado de forma a avaliar e classificar as performances dos Atletas. Por isso, no próximo capítulo, reuniremos as informações da aprendizagem por reforço e do código do pole *sport* na tentativa de entender como a técnica *Q-Learning* pode ajudar a criar coreografias de maior pontuação para os atletas.

4 ALGORITMO PARA POLE *SPORT*

Diante do que foi descrito nos capítulos anteriores, definimos o nosso problema como a criação de uma sequência com ponto de partida e de chegada definidos pelo usuário, dentro das inúmeras possibilidades de sequências de movimentos. Para testarmos a aplicabilidade do algoritmo *Q-Learning* para este problema, foram selecionados dezesseis movimentos, dentre eles sete do tipo força, sete do tipo flexibilidade e dois *deadlifts*, estes movimentos foram escolhidos de forma a haver uma riqueza de possibilidades de transições, resultando em 112 transições no total, conforme apresentado na Figura 3.

Figura 3. Estados e transições



Fonte: Elaborada pela autora.

Quando nos deparamos com o problema proposto, percebemos que a interpretação do que compõe a lista de estados possíveis e de ações possíveis depende muito de como encaramos ele. Pensando no *framework* proposto pela aprendizagem por reforço e pensando em que informações poderiam compor uma observação do estado atual, começamos desenhando uma visão que incluía informações como altura no pole, tipo de pole usado no momento, localização no palco, poderíamos até mesmo levar em consideração o cansaço aumentando ao longo da

execução da sequência. Mas, para resolver o problema de sequenciar os movimentos, podemos usar uma visão do ambiente que inclui exclusivamente o movimento que está sendo executado no tempo t. Tentamos simplificar para dar foco ao que sabemos que podemos controlar, que é sequenciar as ações (transições) possíveis, de forma a atingir um objetivo.

A seguir, descrevemos a lista de transições possíveis a partir de cada estado e seu nível de dificuldade (LOD).

Partindo de S1

S1	para	S30	=	0.5
S1	para	S5	=	0
S1	para	F14	=	0.5
S1	para	S10	=	0.5
S1	para	S15	=	0
S1	para	F7	=	0

Partindo de S30

S30	para	S1	=	0
S30	para	S5	=	0
S30	para	F14	=	1
S30	para	F31	=	1.5
S30	para	S15	=	0.5
S30	para	DL2b	=	1

Partindo de DL2b

DL2b	para	S30	=	0.5
DL2b	para	S1	=	0.5
DL2b	para	S5	=	0.5
DL2b	para	S8	=	1
DL2b	para	F14	=	1
DL2b	para	F31	=	1.5
DL2b	para	F1	=	0.5

DL2b para S15 = 0

Partindo de F7

F7 para S1 = 0.5
F7 para F14 = 0.5
F7 para F26 = 0
F7 para S15 = 0

Partindo de S15

S15 para F7 = 0
S15 para S30 = 0.5
S15 para S1 = 0
S15 para S5 = 0
S15 para F14 = 0.5
S15 para S17 = 0.5
S15 para S10 = 0.5
S15 para F1 = 0.5

Partindo de F1

F1 para S15 = 0.5
F1 para S1 = 0
F1 para S5 = 0.5
F1 para F26 = 0
F1 para F6 = 0.5

Partindo de S10

S10 para S1 = 0.5
S10 para S5 = 0.5
S10 para S17 = 0.5
S10 para S15 = 0.5

Partindo de F6

F6	para	F1	=	0
F6	para	S17	=	0.5
F6	para	F26	=	0

Partindo de F26

F26	para	F6	=	0.5
F26	para	F1	=	0
F26	para	F7	=	0.5

Partindo de F31

F31	para	DL2b	=	2
F31	para	S30	=	1
F31	para	S1	=	0
F31	para	S5	=	0.5

Partindo de S17

S17	para	S10	=	0.5
S17	para	F6	=	0.5
S17	para	F6	=	0.5

Partindo de F14

F14	para	S5	=	0.5
F14	para	S1	=	0.5
F14	para	S1	=	0.5
F14	para	S30	=	0.5
F14	para	S30	=	0.5

F14 para F7 = 0.5
F14 para S15 = 0.5
F18 para S17 = 0.5

Partindo de S5

S5 para DL1b = 1.5
S5 para S1 = 0
S5 para S30 = 0.5
S5 para DL2b = 1.5
S5 para S15 = 0
S5 para F1 = 0.5
S5 para S10 = 0.5
S5 para F31 = 0.5
S5 para S17 = 0.5
S5 para F14 = 0.5

Partindo de DL1b

DL1b para S8 = 0.5
DL1b para F14 = 0.5
DL1b para S5 = 0

Partindo e Chegando em S0

Partindo de S0

S0 para DL1b = 0
S0 para DL2b = 0
S0 para F1 = 0
S0 para F6 = 0
S0 para F7 = 0
S0 para F14 = 0
S0 para F18 = 0

S0	para	F26	=	0
S0	para	F31	=	0
S0	para	S1	=	0
S0	para	S5	=	0
S0	para	S8	=	0
S0	para	S10	=	0
S0	para	S15	=	0
S0	para	S17	=	0
S0	para	S30	=	0

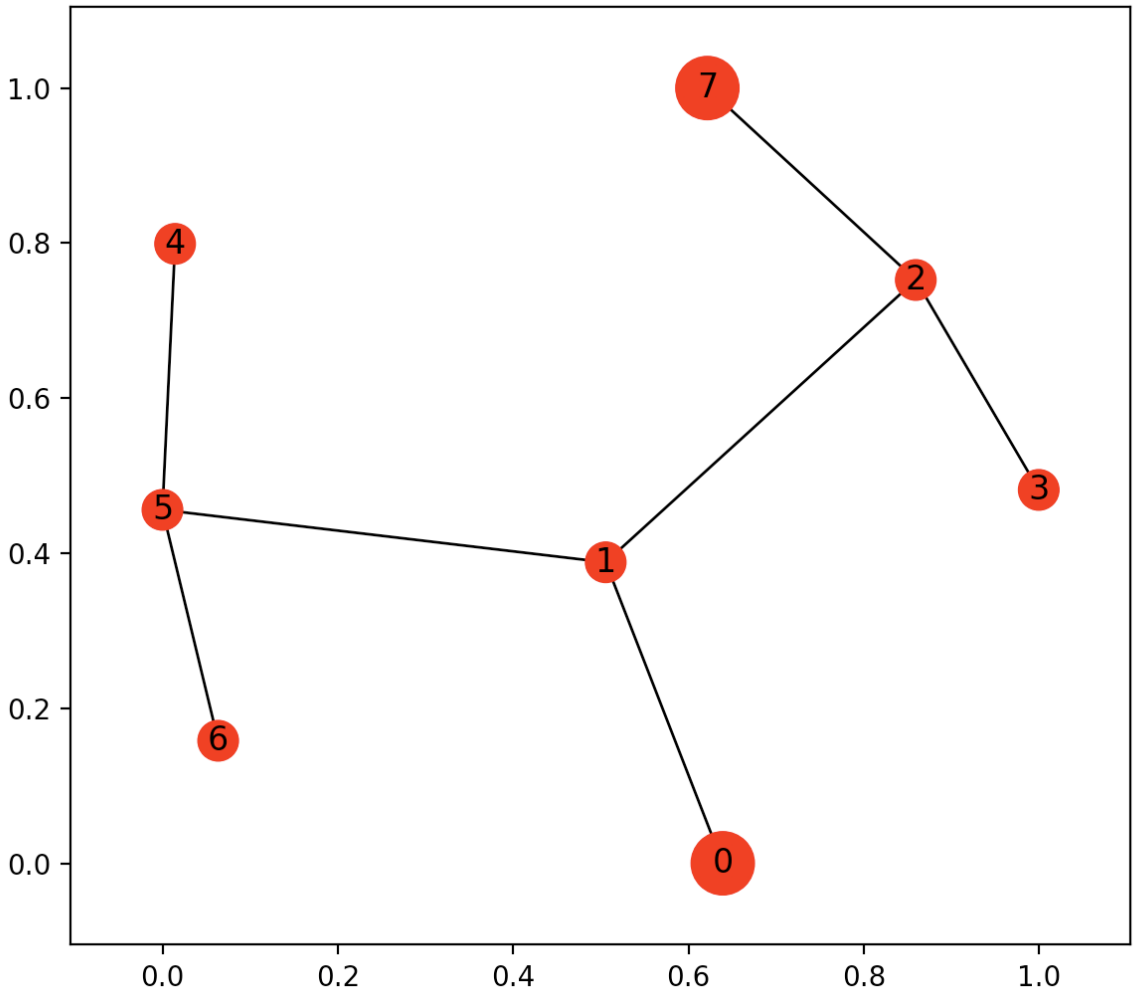
Chegando em S0

DL1b	para	S0	=	0
DL2b	para	S0	=	0
F1	para	S0	=	0
F6	para	S0	=	0
F7	para	S0	=	0
F14	para	S0	=	0
F18	para	S0	=	0
F26	para	S0	=	0
F31	para	S0	=	0
S1	para	S0	=	0
S5	para	S0	=	0
S8	para	S0	=	0
S10	para	S0	=	0
S15	para	S0	=	0
S17	para	S0	=	0
S30	para	S0	=	0

Considerando o problema apresentado, identificamos depois de extensa pesquisa que o programa que melhor adequava-se às nossas necessidades era o desenvolvido por Manuel Amunategui, disponível no site ViralML, inspirado por postagem original no site *The Beginner Programmer*. No caso apresentado por Manuel, ele usa Q-*learning* para encontrar o caminho

mais curto entre dois pontos, a partir de um mapa de lista de pontos que representa que caminhos o *bot* pode tomar, pois assim é possível criar gráficos que mostrem todas as direções possíveis.

Figura 4. Caminho entre dois pontos



Fonte: AMUNATEGUI, 2017, 2:37.

Apesar de o algoritmo selecionado assemelhar-se muito ao nosso problema, algumas adaptações foram necessárias, tanto na etapa de treinamento, em que ajustei o tamanho da matriz de qualidade, as transições possíveis e as recompensas referentes a cada transição, quanto na etapa de teste, em que ajustei o código para retornar uma sequência de 9 movimentos, mesmo que ele já tenha atingido o objetivo.

A matriz atualizada apresenta a nova quantidade de conexões entre os pontos, resultando numa tabela Q inicial de dezessete linhas (quantidade de estados possíveis) por dezessete colunas (quantidade de ações possíveis).

Como estamos lidando com movimentos que podem ou não ser realizados de maneira sequencial, foi necessário mapear quais transições são realizáveis na prática. Assim, na matriz de recompensa, pontos que representam ações impossíveis na prática foram valorados com -1 para indicar que não sejam simuladas, e todas as outras com uma pontuação correspondente à dificuldade da transição.

Segundo o código, o atleta deve escolher entre 9 e 11 movimentos obrigatórios para executar, por isso, uma vez que o algoritmo tende a tomar o caminho mais curto para atingir o objetivo, realizamos a modificação para que a sequência continue até atingir 9 movimentos.

A máquina de estados desenvolvida no presente trabalho é constituída de 17 estados possíveis, desses movimentos, quatro devem ser escolhidos pelo usuário: o ponto de partida, o de objetivo, e uma transição (englobando dois movimentos obrigatórios), e os outros 5 movimentos serão então definidos pela máquina, totalizando uma seleção de 9 movimentos após o treinamento e teste do modelo. A forma de induzir o algoritmo a passar pela transição escolhida pelo usuário é atribuindo para essa uma recompensa maior (200)**.** Já para estimular a chegada ao estado definido como destino, todas as transições que levam até ele recebem uma recompensa intermediária, entre as transições comuns e a escolhida como preferencial (150).

A seguir, apresentaremos o código como estamos utilizando hoje, em Python. Na primeira parte do código é definida a matriz de recompensas populada com -1, e então atribuímos a recompensa de cada transição possível. Na intenção de não haver becos sem saídas, adicionamos o ponto 16 como um ponto neutro que representa o chão e para o qual a maioria dos movimentos pode entrar e sair. É aqui que definimos uma recompensa maior para a transição que desejamos incluir na sequência. Na segunda parte do código, atribuímos uma recompensa de 150 para as transições partindo e chegando no ponto de destino.

Na terceira parte, ficam definidas as funções para iterar e atualizar a matriz de qualidade, sendo elas: *available_actions*, *sample_next_action* e *update*. A primeira recebe o parâmetro estado e retorna as ações disponíveis, a segunda recebe o conjunto de ações disponíveis como parâmetro e retorna uma ação escolhida de forma aleatória. A terceira recebe a ação atual, a ação escolhida e o parâmetro *gamma* e retorna uma recompensa, assim como utiliza a fórmula do *Q-Learning* para a atualização da tabela Q.

Na quarta parte, as funções descritas anteriormente são executadas em 2000 episódios, resultando numa matriz Q que é usada na parte cinco para ajudar na navegação de um ponto A para um ponto B, numa sequência de nove movimentos que começam no ponto inicial selecionado pelo usuário e passam pela transição e pelo ponto de objetivo selecionado. Por último, realizamos uma modificação para que cada ação fosse selecionada apenas uma vez.

Matriz de Recompensas (R)

Tamanho da matriz = 17

Movimentos impossíveis ganham -1

Cada transição ganha recompensa referente ao nível de dificuldade

Recompensa do ponto de destino

Todas as transições relacionadas ao ponto de chegada valem 150 pontos

Matriz de Qualidade (Q)

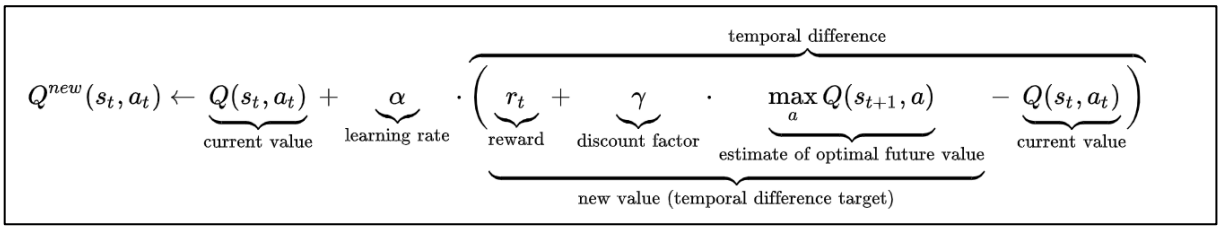
Função *available_actions*, usada para verificar as ações disponíveis passando o estado como parâmetro

Função *sample_next_action*, usada para obter uma próxima ação de forma aleatória dadas as ações disponíveis

Função *update*, usada para atualizar a matriz de qualidade com base no estado atual, ação selecionada e parâmetro gamma

Q[current_state, action] = R[current_state, action] + gamma * max_value

Figura 5. Decomposição do algoritmo Q-Learning



Fonte: WIKIPEDIA, 2022.

Treinamento da Matriz Q

Durante 2000 episódios, o código inicia a execução em um estado aleatório

Verifica quais as ações disponíveis para aquele estado

Escolhe uma dessas ações para seguir e atualizar a matriz Q

Utilização da matriz treinada

Seleciona-se 9 passos, iniciando a sequência no estado definido pelo usuário, e a partir de então escolhendo sempre a ação com valor máximo de Q

4.1 RESULTADOS

Para exemplificar o funcionamento do algoritmo, traremos a seguir duas propostas de entradas do usuário ao algoritmo, a fim de mostrar o que o sistema retorna ao final da aprendizagem. Em todos os casos, o programa levou menos de meio minuto para retornar a tabela Q final e, consequentemente, a melhor sequência.

Exemplo 1:

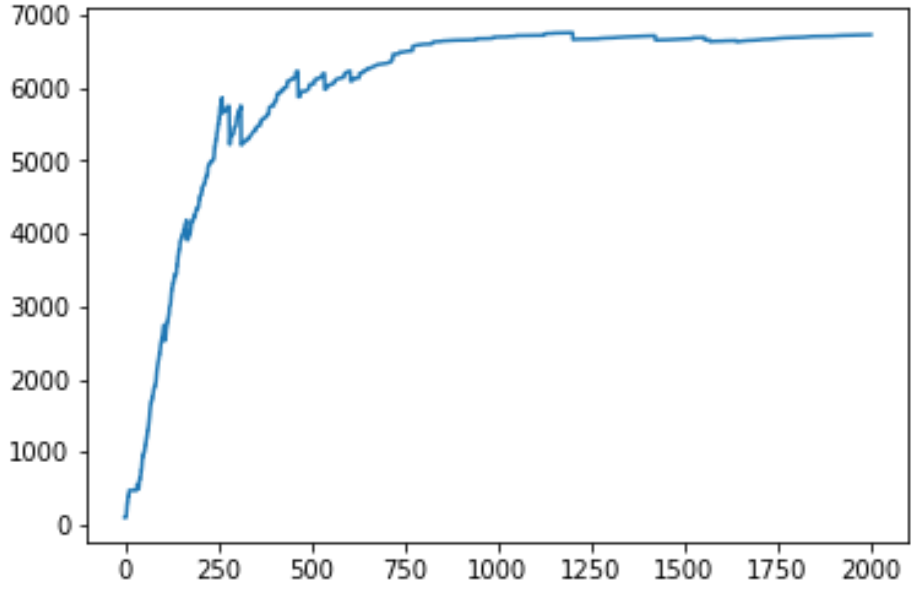
Um atleta deseja construir uma coreografia que parta do movimento S0 (estado 16) e chegue no movimento F31 (estado 8), passando pela transição F1 para F26 (ação (2, 7)). Podemos realizá-la, então, alterando:

```
current_state = 16
goal = 8
R[(2,7)] = 200
```

Most efficient path:
[16, 2, 7, 3, 16, 8, 1, 10, 13, 9]

No momento da utilização da matriz treinada, o programa gera um gráfico com a recompensa (eixo y) obtida por episódio (eixo x). Nele, pode-se observar a partir de quantos episódios a recompensa obtida começa a convergir para um valor máximo. Para este primeiro exemplo, ele nos mostra um crescimento rápido até o episódio 250, que desacelera até o episódio 750, quando as recompensas começam a se estabilizar.

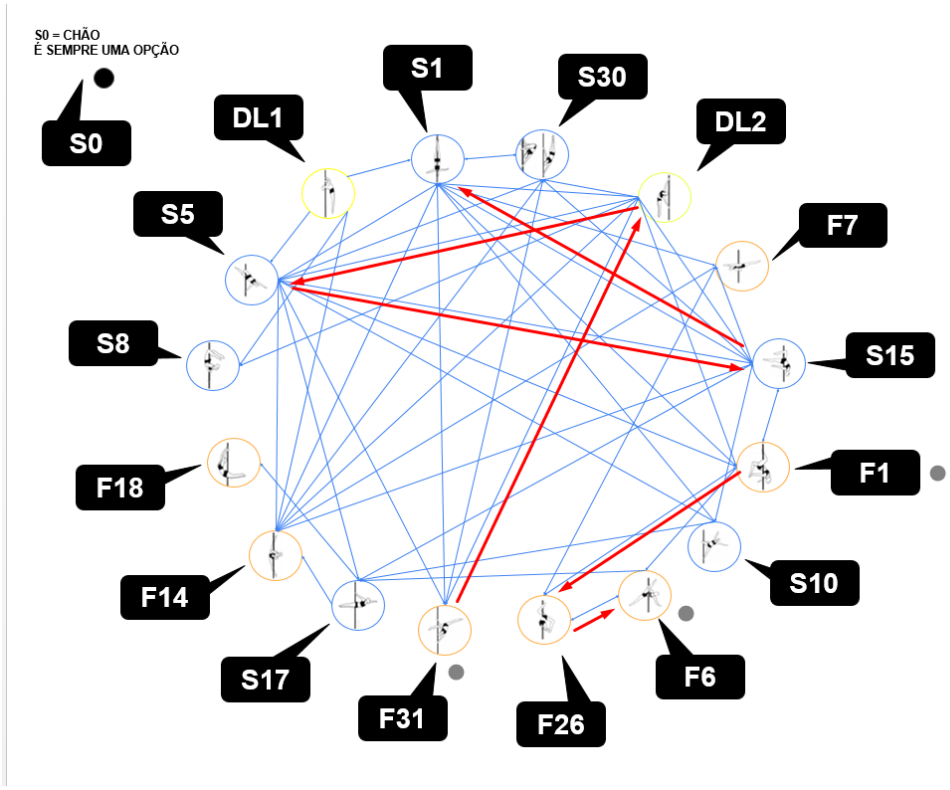
Gráfico 1. Recompensa por episódio do exemplo 1



Fonte: Elaborada pela autora.

Sequência obtida: [S0, F1, F26, F6, S0, F31, DL2b, S5, S15, S1]

Figura 6. Coreografia resultante do exemplo 1



Fonte: Elaborada pela autora.

Exemplo 2:

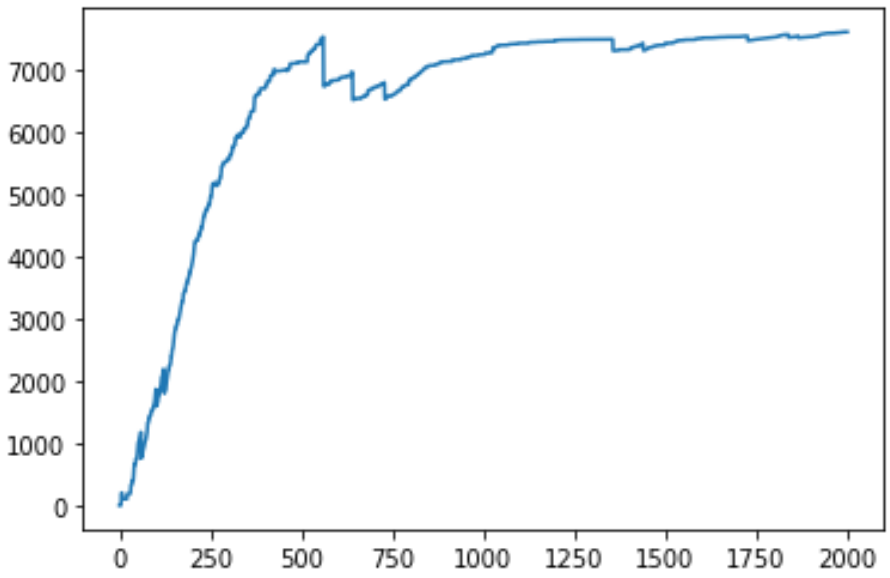
Uma atleta do nível intermediário deseja começar sua coreografia do campeonato no movimento S10 (estado 12) e ir ao movimento S30 (estado 15) passando pela transição S15 para F1 (ação (13, 2)). Para isso geramos as seguintes modificações no algoritmo:

```
current_state = 12
goal = 15
R[(13,2)] = 200
```

Most efficient path:
[12, 13, 2, 10, 15, 8, 1, 5, 9, 16]

No gráfico obtido no segundo exemplo, da mesma maneira explicada no Exemplo 1, pode-se observar a partir de quantos episódios a recompensa obtida começa a convergir para um valor máximo. Agora, atinge um primeiro pico no episódio 500, sofrendo uma pequena queda até 750 e estabilizando a partir de 1000 episódios.

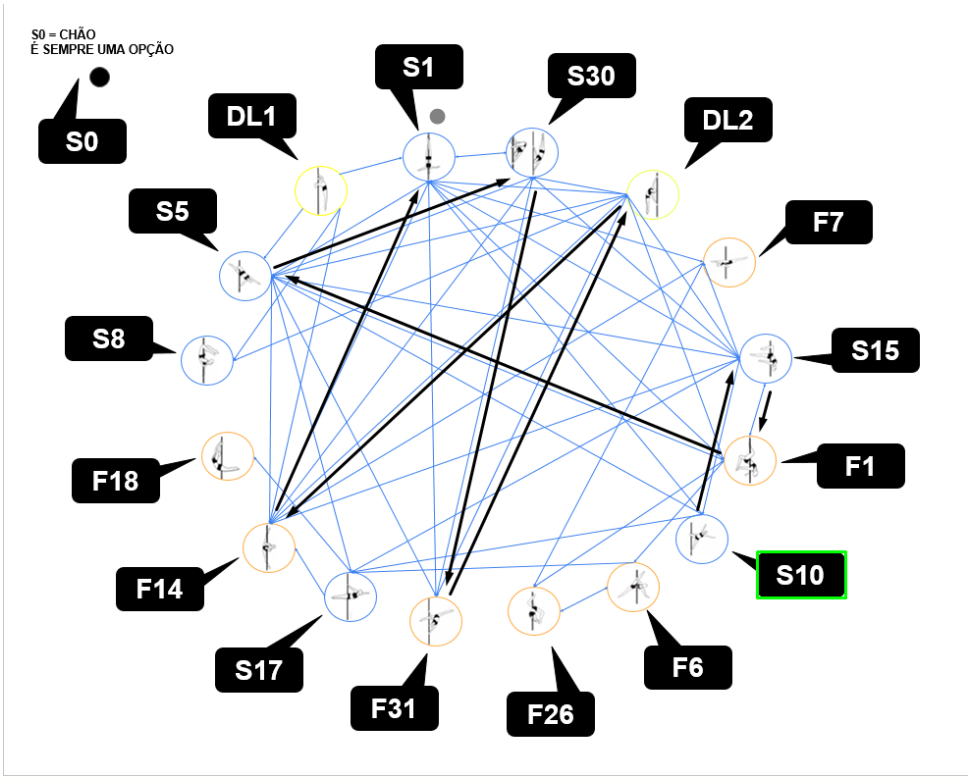
Gráfico 2. Recompensa por episódio do exemplo 2



Fonte: Elaborada pela autora.

Sequência Obtida: [S10, S15, F1, S5, S30, F31, DL2b, F14, S1, S0]

Figura 7. Coreografia resultante do exemplo 2



Fonte: Elaborada pela autora.

Entretanto, essa atleta informou que ainda não sabe executar o movimento F31 (estado 8), e nos solicitou que acrescentássemos a condição de que o algoritmo não executasse tal movimento. Sendo assim, fizemos mais uma modificação:

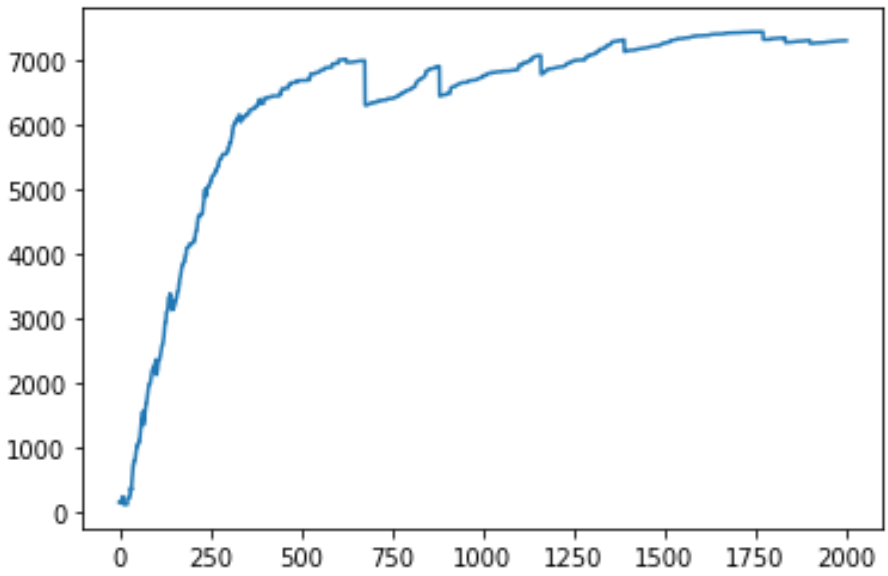
```
excluir = 8

for point in points_list:
    print(point)
    if point[1] == excluir:
        R[point] = -1

    if point[0] == excluir:
        R[point[::-1]] = -1
```

Most efficient path:
[12, 13, 2, 16, 15, 5, 9, 4, 7, 3]

Nessa nova simulação a exploração inicial parece se dar de forma parecida até 300 episódios, mas com vários máximos locais até convergir somente após 1300 episódios. Esse resultado pode indicar alguma dificuldade em manter o aprendizado adquirido, os momentos

de exploração do algoritmo ou que o modo como as recompensas estão distribuídas para indicar o objetivo não seja ideal.

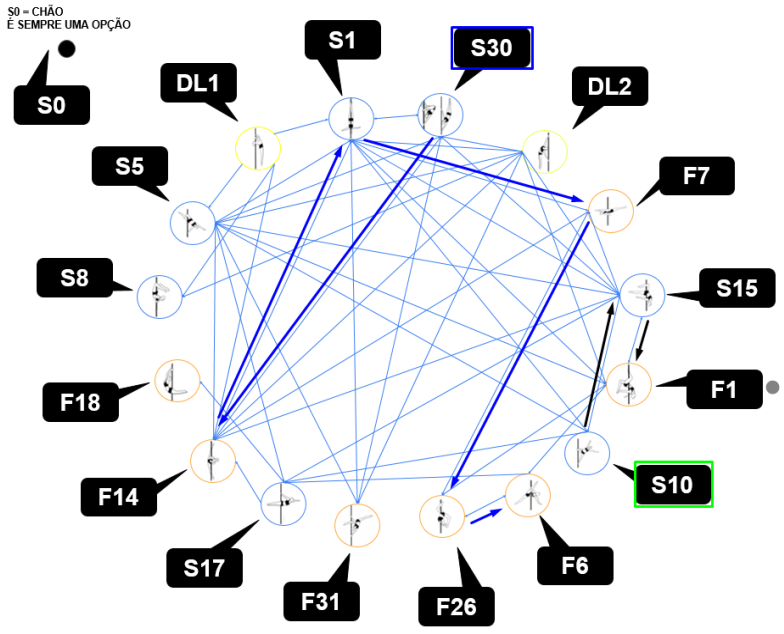
Gráfico 3. Recompensa por episódio do Exemplo 1 após remover o ponto 8



Fonte: Elaborada pela autora.

A nova sequência retornada foi: [S10, S15, F1, S0, S30, F14, S1, F7, F26, F6]

Figura 8. Coreografia resultante do exemplo 2 após remover o ponto 8



Fonte: Elaborada pela autora.

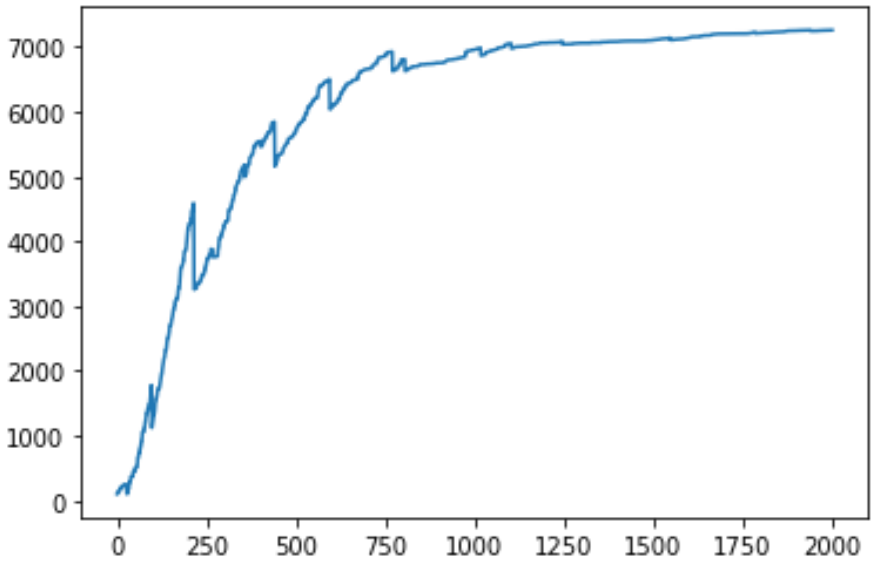
A atleta em questão gostou desta coreografia, porém, chamou nossa atenção para a questão da criatividade da sequência, já que do movimento inicial, já passa para a transição obrigatória e volta para o chão entrando diretamente no movimento objetivo. Apesar de a sequência representar um bom valor de recompensas, uma das pontuações mais relevantes no concurso é a da criatividade - questão que não podemos controlar com o algoritmo, mas podemos adaptar às percepções de cada usuário. A atleta prefere não sair do estado 12 direto para o 13. Precisa que o algoritmo encontre outra forma. Para isso, atribuímos recompensa negativa para a transição 12, 13.

```
R[(12,13)] = -1
```

Most efficient path:
[12, 9, 15, 13, 2, 16, 1, 5, 4, 7]

Essa última simulação apresenta um padrão parecido com o do gráfico anterior, com alguns picos instáveis parecendo indicar momentos de exploração do algoritmo. Desta vez, crescendo mais lentamente, mas estabilizando mais cedo, ao redor do milésimo episódio.

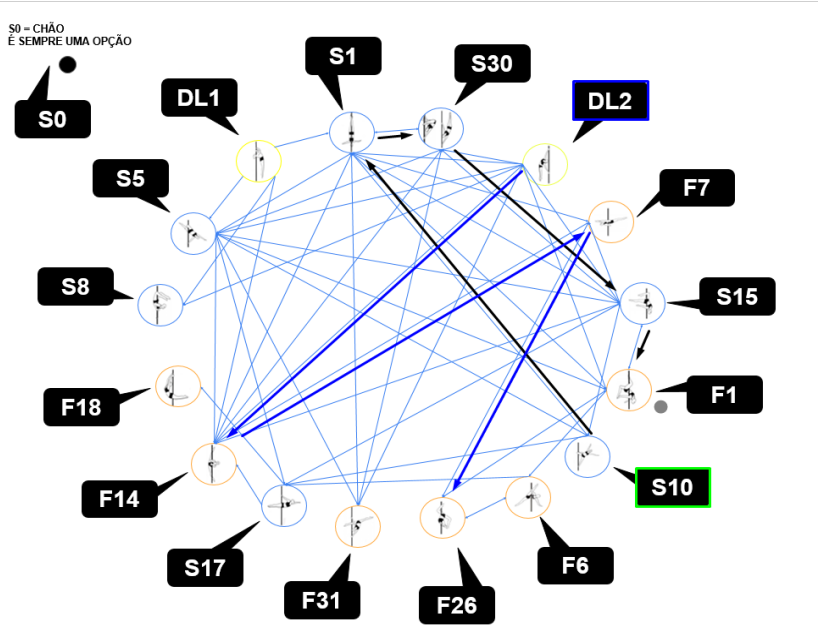
Gráfico 4. Recompensa por episódio do Exemplo 1 após remover o ponto 8 e transição 12,13



Fonte: Elaborada pela autora.

A nova sequência retornada foi: [S10, S1, S30, S15, F1, S0, DL2b, F14, F7, F26]

Figura 9. Coreografia resultante do Exemplo 2 após remover o ponto 8 e transição 12,13



Fonte: Elaborada pela autora.

Chegamos então a uma sequência realizável, de boa pontuação, e que se adequa às particularidades do usuário. A sequência, apesar de não levar em conta a recompensa de cada estado, traz uma coreografia que pontuaria muito bem, tanto em termos de conexão entre os movimentos, quanto de nível de dificuldade, além, é claro, de incluir os movimentos e transições que se desejava explorar.

Algumas regras, como quantidade de movimentos de cada tipo ainda não estão sendo levadas em conta, mas coincidentemente, foram selecionados quatro movimentos de força (borda azul), quatro de *flex* (borda laranja) e um *deadlift* (borda amarela), assim como uma passagem pelo chão. Essa coreografia poderia vencer um campeonato nível profissional ou elite, bastaria acrescentar os giros separadamente.

Percebemos que o algoritmo está funcionando bem, passando por todos os pontos selecionados. Observe que mesmo que o objetivo seja atingido muito antes, a indicação é sempre de 10 movimentos encadeados, incluindo passagem pelo chão, na tentativa de trazer os 9 movimentos obrigatórios necessários para inscrição no campeonato. No último exemplo, um dos dez movimentos foram o chão, os outros 9 seriam os elementos obrigatórios da coreografia.

Essa experiência fez-nos perceber que, caso o algoritmo vire um aplicativo, deveria ser mais fácil montar uma coreografia somente com movimentos simples, que se adequem ao nível de desenvolvimento do aluno ou atleta. Além de ser possível excluir os movimentos, é possível também a criação de um fator de confiabilidade (a capacidade do atleta de executar a ação) a

partir das informações de dificuldade. Assim a melhor sequência retornada pelo algoritmo refletiria uma nova noção de qualidade.

O mesmo algoritmo que apresentamos em Python no *Google Colab* foi também implementado em VBA no Excel em que ficou mais clara a influência dos parâmetros *alpha*, *gamma* e *epsilon* na aprendizagem e a forma como a tabela Q evolui a cada iteração. Quanto maior o *epsilon*, mais valorizada a exploração de novos caminhos em detrimento do caminho até então mapeado como o melhor; quanto maior o *gamma*, mais valorizada a recompensa futura em detrimento da recompensa imediata; quanto maior o *alpha*, mais valorizadas as informações novas em detrimento da base de conhecimento já adquirida sobre o atual estado. O ideal seria variar o valor dos parâmetros gradativamente.

No *Excel* foi possível perceber que a depender da escolha dos parâmetros o treinamento não convergia para um resultado mesmo depois de muitos episódios, por perder a memória do que aprendeu ou não conseguir explorar todas as possibilidades. É preciso explorar os parâmetros por tentativa e erro até encontrar uma combinação que atualize a matriz direcionando-a para o resultado desejado. Em cerca de 15 tentativas foi possível achar os valores corretos para cada parâmetro, possibilitando que o algoritmo pronto consiga rodar em 2 minutos.

5 CONSIDERAÇÕES FINAIS

A aprendizagem por reforço vai se desenvolvendo e aprende ao longo do tempo, assim como os atletas. Em muitas das aplicações não existe um mapeamento das recompensas por pares de estado-ação como mostrado aqui. Mas sim por meio de aproximadores de função, unindo vários tipos de inteligência artificial em um só projeto. Nesse sentido, trabalhamos com a intenção de solucionar o problema por meio da aprendizagem por reforço pura, sem mesclá-la com outros métodos.

Um ponto que se mostrou importante nessa pureza foi a possibilidade de usar a aprendizagem por reforço com um viés criativo, que geralmente é considerada ruim em aplicações que requerem previsibilidade, mas nesse caso é uma coisa boa porque soluciona um problema também de ordem criativa. Sendo assim, pudemos explorar um lado ainda pouco utilizado da inteligência artificial.

Apesar de termos feito uma versão simplificada do algoritmo, com menos movimentos, essa provou-se útil para englobar todas as variáveis que compõem o ambiente de um campeonato de pole *sport*. Ao incluir novas variáveis no algoritmo, visualizamos a possibilidade de torná-lo, consecutivamente, mais complexo e mais confiável para a utilização profissional de atletas. Da mesma forma, como são áreas similares, os mesmos procedimentos para criar o algoritmo para o pole *sport* também podem ser usados para criar algoritmos de outros esportes, como a ginástica rítmica, ginástica olímpica, e etc.

Dito isso, acreditamos que uma grande contribuição desse trabalho para a aprendizagem por reforço é o uso dessa para solucionar problemas criativos e com regras contextualizadas no mundo real, para além de sair do ponto x e chegar no ponto y. Utilizando a técnica *Q-Learning*, em seu estado puro, provando ser possível sequenciar as melhores ações a serem tomadas com base em um sinal de recompensa.

REFERÊNCIAS

AMUNATEGUI, Manuel. **Reinforcement Learning - A Simple Python Example and A Step Closer to AI with Assisted Q-Learning**. Vídeo digital: 16’18”. 30 de setembro de 2017. Disponível em: https://www.youtube.com/watch?v=nSxaG_Kjw_w&ab_channel=Crypto-FintechMeanderings. Acesso em: 30 de agosto de 2022.

BILGIN, Enes. **Mastering Reinforcement Learning with Python**. Birmingham, Mumbai: Packt>, 2020

Data Science Academy. **Deep Learning Book**, 2022. Disponível em: https://www.deeplearningbook.com.br/ . Acesso em: 10 Janeiro. 2022.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep learning**. Book in preparation for MIT Press org, v. 1, 2016. Disponível em: http://www. deeplearningbook.

IPSF - International Pole Sports Federation. **Campeonatos de Pole Sports: Código de Pontos 2018/2019**. 2018. Disponível em: www.polesports.org. Acesso em: 10 de Janeiro de 2022.

KAELBLING, Leslie Pack; LITTMAN, Michael L.; MOORE, Andrew W. Reinforcement Learning A Survey. **Journal of Artificial Intelligence Research**, n. 4, 1996, p. 237 - 285.

KLEINA, Nilton. Colossus: Herói de guerra e um dos primeiros computadores do mundo. **Tecmundo: descubra e aprenda tudo sobre tecnologia**, v. 25, 2020. Disponível em: https://www. tecmundo. com. br/historia/40576-colossus-heroi-de-guerra-e-um-dosprimeiros-computadores-do-mundo. htm

LAPAN, Maxim. **Deep Reinforcement Learning Hands-On**. Birmingham, Mumbai: Packt>, 2020.

LOSANO, Mario Giuseppe; JOFFILY, Bernardo. **Histórias de autômatos: da Grécia Antiga à Belle Époque**. Comp. das Letras, 1992.

MINSKY, Marvin; PAPERT, Seymour. **Perceptrons**: An Introduction to Computational. MIT press, 1988.

NAG, Utathya. **Mallakhamb** – Know the rules, history and origin of this ancient Indian sport. Olympic Games, 28 de Abril de 2022. Disponível em: https://olympics.com/en/news/what-is-mallakhamb-rules-history-origin-india Acesso em: 19 de Novembro de 2022.

PUCRS Online. **Inteligência Artificial:** conceito e como dominá-la. Publicado em 6 de Outubro de 2020. Disponível em: https://online.pucrs.br/blog/public/inteligencia-artificial-conceito. Acesso em: 18 de Novembro de 2022.

ROSENBLATT, F. The Perceptron: a probabilistic model for information storage and organization in the brain. **Psychological Review** Vol. 65, No. 6, 1958, p. 386 - 408. Disponível em: http://134.208.26.59/INA/A%20probabilistic%20model.pdf. Acesso em 18 de Novembro de 2022.

SHANNON, Claude E.; WEAVER, Warren. **The mathematical theory of communication** (4 udg.). Urbana: The University of Illinois Press, 1969.

SUTTON, Richard S.; BARTO, Andrew G. **Reinforcement Learning:** An Introduction. Cambridge, London: The MIT Press, 2014 - 2015.

WIDROW, Bernard; HOFF, Marcian E. **Adaptive switching circuits**. Stanford Univ Ca Stanford Electronics Labs, 1960.

WIERING, Marco A.; VAN OTTERLO, Martijn. Reinforcement learning. **Adaptation, learning, and optimization**, v. 12, n. 3, p. 729, 2012.

WIERING, Marco; VAN OTTERLO, Martijn. **Reinforcement Learning**: State-of-the-Art. Berlim, Heidelberg: Springer, 2012

WIKIPEDIA. **Q-Learning.** 18 de Novembro de 2022. Disponível em: https://en.wikipedia.org/wiki/Q-learning Acesso em: 30 de Dezembro de 2022.

WILMOTT, Paul. **Machine Learning:** An Applied Mathematics Introduction. [*S. L.*]: Panda Ohana Publishing, 2020.