Utilizando Aprendizado por Reforço no treinamento de uma IA visando localizar contraexemplos para conjecturas contidas na Teoria dos Grafos

Lucas Freire de Siqueira
Bacharelado em Ciência e Tecnologia
UNIFESP - Universidade Federal de São Paulo
São José dos Campos, Brasil
Lucas.Freire@unifesp.br

***Abstract*—Existem diversos exemplos do auxílio que uma IA pode trazer quando utilizada para resolução de problemas matemáticos. Dentro da Teoria dos Grafos existe uma grande quantidade de conjecturas a serem provadas ou descartadas e, através deste trabalho, para auxiliar a criação de contraexemplos, visando separar conjecturas com potencial de serem verdades daquelas que não as são, um algoritmo que ensina uma Inteligência Artificiai (IA) utilizando o paradigma de Aprendizado por Reforço apresenta possibilidades interessantes no cumprimento desse objetivo. Neste artigo são relatadas e explicadas as técnicas utilizadas para tais treinamentos e posteriormente são discutidos os resultados obtidos.**

***Index Terms*—teoria, grafos, aprendizado, reforço, conjectura, inteligência, artificial**

I. Introdução

Sendo um dos ramos da matemática mais explorado e estudado na atualidade, a Teoria dos Grafos apresenta grandes avanços em suas aplicações, porém diversas conjecturas contidas em seu escopo, continuam sem uma prova cabal. Nos últimos anos a Inteligência Artificial (IA) demonstrou interessantes possibilidades na tentativa de assumir o papel de ferramenta nas tentativas de criação de contraexemplos de diversas conjecturas. Como o próprio nome já diz, a Teoria dos Grafos é o campo de estudo das propriedades e relações dos Grafos, que por sua vez são estruturas matemáticas compostas de vértices e arestas e que, geralmente, são utilizados na modelagem de problemas reais.

Este artigo tem como proposta o acompanhamento da inicialização, do treinamento, dos testes e da utilização de uma IA que seja capaz de identificar padrões de grafos, fazendo como que através de seu treinamento, seja possível demonstrar contraexemplos para conjecturas pertencentes a Teoria dos Grafos. Criar contraexemplos para conjecturas sólidas não é um trabalho trivial, mas existem diversos exemplos para problemas menores, e são esses, principalmente, que serão explorados. A técnica escolhida para o treinamento desta IA será a Aprendizagem por Reforço, pois a mesma já demonstrou em outros estudos grandes avanços nessa área de estudo de Teoria dos Grafos, e o m.

O potencial de exploração dessa área é amplo e pode vir a se provar muito útil em um futuro próximo. Como os grafos estão sendo cada vez mais utilizados em atividades de modelagem de problemas do mundo real, criar uma profunda compreensão de suas propriedades, de seus relacionamentos, de seus comportamentos e características, não apenas torna útil esse aprendizado, como também o torna necessário pois diversos serviços da sociedade estão cada vez mais integrados a sistemas inteligentes de administração, havendo, desta maneira, a necessidade da elaboração de um sistema robusto que seja composto por uma base sólida e clara de entendimento de suas funcionalidades.

II. Conceitos Fundamentais e Trabalhos Relacionados

Para que haja um bom entendimento da proposta deste artigo, será dada uma breve explicação de alguns conceitos pertencentes a área de Inteligência Artificial, em especial sobre a área de Machine Learning e sobre o método de Reinforcement Learning (Aprendizado por Reforço). Além disso será explicado de maneira sucinta, o que são Redes Neurais Convolucionais, que é uma técnica utilizada utilizada nesse código e também sobre o Deep Cross-Entropy Method, que é uma técnica algarítmica utilizada para o treinamento da IA nesse trabalho.

Dadas as explicações desses temas, serão realizados rápidos resumos de alguns artigos que formam a base de pesquisa deste trabalho, pois cada um deles envolve pelo menos um assunto que seja pertinente ao desenvolvimento de uma IA que aprende por reforço e/ou utiliza redes convolucionais e/ou o Deep Cross-Entropy Method.

A. Aprendizado por Reforço

A técnica de Aprendizado por Reforço, simula o método com o qual grande parte dos seres humanos aprende, principalmente na infância. A formalização matemática utilizada para representar essas tomadas de decisões é chamada de Processo de Decisão de Markov (PDM) e é através dele que acontece a estruturação necessária para que os problemas encarados pela IA possam ser resolvidos.Dentro do PDM, temos os seguintes componentes:

- Ambiente;

- Agente;
- Estado
- Ação
- Recompensa/Punição;

Ao decorrer do tempo, o agente, que no começo está em um estado inicial, interage com o ambiente por meio de ações que são determinadas por uma policy pi. Ao realizar uma ação, o agente muda de um estado (St) para um novo estado (St+1). Ao atingir este novo estado, o agente ganha uma recompensa baseada em quão boa foi sua decisão de tomada de ação no estado anterior. O objetivo do agente é, não apenas obter a melhor recompensa em cada estado, como também ter a maior recompensa possível acumulada através do tempo, passando pelo menor número de estados possíveis. Esse processo de troca de estados por meio de ações e que obtém recompensas, chama-se trajetória, e define o mapa percorrido pelo agente até o seu estado final. Quando o agente consegue a melhor recompensa possível passando pelo melhor caminho através dos estados possíveis, diz-se que o a Trajetória Otimizada foi encontrada.

Como cada estado está contido em um conjunto de Estados (chamaremos este conjunto de S) e cada ação está contida em um conjunto de Ações (chamaremos este conjunto de A), o par gerado por (St, At), ou seja, a ação tomada em cada estado ao decorrer do tempo, terá como resposta um valor r que está dentro de um Conjunto R de recompensas. Assim, uma trajetória pode ser assumida da seguinte forma:

S0,A0,R1,S1,A1,R2,S2,A2,R3,S3,A3...Rn,Sn,An,Rn+1.

Levando em conta que os conjuntos de estados e recompensas contam com um número finito de variáveis aleatórias, é possível afirmar que ambos têm como característica distribuições probabilísticas bem definidas. Ou seja, cada valor associado a um par Rt e St, tem uma probabilidade associada a ele.

Por fim, um agente não conhece de fato as regras do jogo. Ele apenas executa ações baseadas em uma política PI e obtém através disso recompensas ou punições. Quem de fato conhece as regras, no caso da IA implementada nesse trabalho, é uma função Score, um função de ”pontuação” do jogo. Ela diz o quão perto ou distante o agente está de seu objetivo, e é por meio do retorno dessa função que o agente consegue aprender ao explorar o ambiente, no caso a inserção ou não de arestas no grafo.

B. Redes Neurais Convolucionais

Comumente conhecidas por sua sigla em inglês ’CNN’ (Convolutional Neural Networks) são redes neurais artificiais inspiradas em processos biológicos, do tipo feed-foward e que tentam utilizar o mínimo possível de pré-processamento, sendo esse um grande diferencial quando comparada com outros algoritmos de classificação. Pelo fato das CNNs terem uma arquitetura que melhor trabalha com dados que são estruturados como grades, sendo matrizes um desses tipos de estruturas, elas são uma ótima ferramente para se utilizar quando queremos analisar matrizes de adjacências que são representações de grafos, por exemplo.

Apesar de apresentarem muitas das características de redes neurais mais primitivas, como retropropagação, funções de ativação não lineares (sigmoide, ELU, ReLU, etc), gradiente descendente, as CNNs tem como grande diferencial a sua camada convolucional. Essa camada utiliza filtros convolucionais para percorrer a imagem de entrada, aplicando uma operação de multiplicação ponto a ponto em cada região local da imagem. Esses filtros são aprendidos automaticamente durante o treinamento da rede neural, ajustando seus pesos para capturar padrões visuais relevantes.

A partir desse passo, uma série de mapas de características são criados com o objetivo de destacar características ou a ausência delas, na imagem gerada como input. Após isso é aplicado, em camadas posteriores, o pooling, que é o método utilizado para reduzir a qualidade da imagem, através da redução de sua dimensionalidade, dessa forma fazendo com que as características mais importantes sejam preservadas.

Com essas ações, as CNNs aprendem quais características devem ser detectadas e como elas devem ser classificadas. Camadas que são totalmente conectadas têm seus pesos ajustados durante os treinamentos, justamente como forma de melhorar essas detecções.

C. Deep Cross-entropy Method

O Deep Cross-Entropy Method é um algoritmo de aprendizado por Reforço que utiliza uma mistura de Deep Neural Networks (no caso desse trabalho, as CNNs), com o método de Entropia Cruzada, um método Monte Carlo muito utilizado na área de Otimização.

De acordo com a Entropia Cruzada, existem duas distribuições de probabilidade, sendo p (verdadeira) e q(estimada), para um mesmo conjunto de eventos. Ele realiza um medição do número médio de bits necessários para se identificar um evento e gerar uma medida capaz de quantificar a diferença entre essas duas probabilidades q e p. Dessa maneira, quanto menor for a entropia cruzada, mais similares são as distribuições, assim facilitando o trabalho de predição realizado pelo algoritmo. A treinarmos uma Deep Neural Network utilizando esse método, eventualmente é possível criar um modelo de previsão bastante eficiente.

Geralmente esse método é usado com loss function em algoritmos de treinamento em problemas de aprendizado de máquina.

D. Grafos

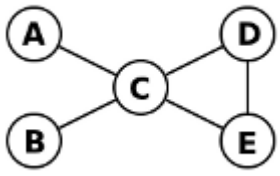

Fig. 1. Exemplo da estrutura de um grafo

Um Grafo é uma estrutura composta por vértices e arestas, sendo descrito por um par (V/E), onde V é o conjunto de vértices e E é o conjunto de arestas. Ele representa uma estrutura de um conjunto de objetos, informações, e das relações que existem entre si. As arestas que existem para representar essas relações podem ter custo e/ou definirem direções/sentidos a serem percorridos, sendo assim chamados de grafos dirigidos ou direcionados, e caso não as arestas não tenham direção definidas, os grafos são comumente chamados de não dirigido. Na Fig. 1. pode-se observar um exemplo de um grafo não dirigido com 5 vértices e 4 arestas.

Os grafos são muito utilizados em trabalhos que exigem algum tipo de modelagem de um sistema, podendo ser ele real ou não, e uma das maneiras mais fáceis e práticas de representar um grafo é através de uma matriz de adjacência, que é o modelo de representação escolhido para este trabalho.

	A	B	C	D	E
A	0	0	1	0	0
B	0	0	1	0	0
C	1	1	0	1	1
D	0	0	1	0	1
E	0	0	1	1	0

Fig. 2. Matriz de Adjacências representando o grafo da Fig. 1.

As linhas de uma matriz de adjacências MxN, (representadas por i, tal que i varia de 1 até M, onde M = número total de vértices contidos no grafo) representam os vértices e suas colunas (representadas por j, tal que j varia de 1 até N, onde N = número total de possíveis relações de dado vértice com outro vértice) representam quais vértices têm ligações através de arestas entre si. O valor de 0 na posição ij da matriz MxN representa a não relação entre os vértices, enquanto o valor 1 na posição ij da matriz de adjacências MxN representa a ligação de vértices através de uma aresta.

E. Trabalhos relacionados

Foi realizada uma pesquisa de artigos que poderiam estar relacionados, de alguma maneira, com os assuntos pertinentes a este trabalho, variando desde a abordagem do funcionamento de algoritmos de Aprendizado por Reforço, passando por alguns métodos específicos deste tipo de aprendizado (Cross Entropy Method), até chegar no estudo de Redes Neurais profundas que foram utilizadas para o estudo do dinamismo dos grafos. O único artigo que, necessariamente, aborda exatamente o mesmo assunto é o artigo [3] de Adam Z. Wagner, " Construction in Combinatorics via neural networks", de ordem baseada na ideia deste trabalho.

Segue um breve resumo do ponto principal de cada artigo pesquisado para a realização deste trabalho, seguindo a lista de referências contidas no final do trabalho.

No primeiro artigo [1] os autores discutem o uso de aprendizado de máquina na matemática para auxiliar na descoberta de padrões, formulação de conjecturas e teoremas. Destaca-se que os matemáticos têm utilizado computadores desde a década de 1960 para esse propósito, citando a conjectura de Birch e Swinnerton-Dyer como um exemplo bem conhecido. O texto apresenta exemplos de novos resultados matemáticos que foram descobertos com a ajuda do aprendizado de máquina, demonstrando como ele pode auxiliar os matemáticos a propor novas conjecturas e teoremas.

No segundo artigo [2] é estudado um problema de controle ótimo com restrições. O objetivo é encontrar um controlador que maximize uma função objetivo enquanto satisfaz uma restrição, dado um modelo de sistema dinâmico com estados e ações contínuas. O texto destaca que esse tópico tem sido estudado há décadas dentro da comunidade de controle, mas ainda apresenta desafios práticos.

No terceiro artigo [3] aborda-se as formalizações de provas assistidas por computador, incluindo resultados importantes, como a prova do teorema das quatro cores em 1976 por Appel e Haken, e a prova da conjectura de Kepler em 1998 por Hales. Afirma-se ainda que recentemente, houve progresso significativo na área de algoritmos de aprendizado de máquina, e eles se tornaram ferramentas interessantes, sendo especificados os avanços recentes no campo do Aprendizado por Reforço que levaram os computadores a alcançar um nível de jogo sobre humano em jogos como Atari e Go.

No quarto artigo [4] atesta-se que o objetivo principal do campo da inteligência artificial é resolver tarefas complexas a partir de entradas sensoriais não processadas e de alta dimensionalidade. O artigo afirma que, recentemente, houve progresso significativo ao combinar avanços no aprendizado profundo para processamento sensorial com aprendizado por reforço, resultando no algoritmo "Deep Q Network" (DQN) capaz de alcançar desempenho humano em muitos jogos Atari usando pixels não processados como entrada. Para isso, aproximadores de função de redes neurais profundas foram usados para estimar a função valor de ação.

No quinto artigo [5] é abordado o DeepStruct que é uma estrutura que combina modelos de aprendizado profundo e teoria dos grafos, permitindo a imposição de várias estruturas de grafo em redes neurais ou a extração de estruturas de grafo a partir de modelos treinados. Ele fornece modelos de redes neurais profundas com diferentes restrições baseadas em um grafo inicial e oferece ferramentas para extrair estruturas de grafo de modelos treinados.

No sexto artigo [6] é atestada a importância de aprender representações úteis de grafos para o sucesso de muitos algoritmos de aprendizado de máquina. Muitos problemas do mundo real podem ser formulados como grafos e sua aplicação abrange diversas áreas de pesquisa, como classificação, previsão de links, análise de redes dinâmicas, entre outras e embora existam conjuntos de dados estruturados disponíveis em domínios como redes sociais, redes de transporte, redes financeiras e redes cerebrais, poucos algoritmos conseguem lidar com grandes quantidades desses dados estruturados e

fazer interpretações úteis a partir deles. A pesquisa atual em aprendizado de máquina e aprendizado profundo geralmente se concentra no uso de conjuntos de dados não estruturados, como imagens, áudios e texto. No entanto, a capacidade de interpretar e obter conclusões significativas a partir de conjuntos de dados estruturados, principalmente aqueles em forma de grafos, é limitada. O objetivo é preencher essa lacuna desenvolvendo um modelo de rede neural que combine redes neurais convolucionais e redes neurais recorrentes (redes LSTM) para capturar tanto a estrutura quanto a dinâmica temporal dos grafos.

No sétimo artigo [7] é estudada a abordagem de aplicação de uma IA, através de um deep learning, para classificação de semigrupos finitos. O objetivo é conseguir determinar as classes de isomorfismo de semigrupos de determinados tipos. O estudo considera se um processo projetado para aprender a fazer provas usando redes neurais pode aprender a produzir provas ”melhores”, medida pelo número de nós na árvore de prova.

No oitavo artigo [8] é dito que a aprendizagem baseada em modelo oferece várias aplicações potenciais, incluindo aprendizagem de políticas, compreensão de cenas, exploração, motivação intrínseca e raciocínio contrafactual. Os autores apresentam o Simulated Policy Learning (SimPLE), um sistema que utiliza técnicas estocásticas de previsão de vídeo e treina uma política para jogar jogos dentro do modelo aprendido. Por meio da agregação de conjuntos de dados e iterações, o SimPLe alcança eficiência de amostragem e joga com sucesso jogos no ambiente real. Avaliações empíricas mostram que o SimPLe supera uma versão ajustada do algoritmo Rainbow em eficiência de amostragem na maioria dos jogos. No regime de baixa quantidade de dados, o SimPLe alcança resultados comparáveis ao Rainbow, sendo que ambos os métodos possuem pontos fortes em jogos diferentes.

No nono artigo [9] é discutida a aplicação da programação linear e solucionadores de PL como um método valioso no estudo e na refutação de conjecturas abertas na combinatória extrema. Embora não seja uma abordagem nova, argumenta-se que este método pode ser útil na determinação da probabilidade de uma afirmação ser verdadeira ou falsa em conjecturas complexas. Os autores utilizam a programação linear para abordar várias conjecturas, perguntas e problemas em aberto em diferentes domínios.

No décimo artigo [10] é proposto um modelo profundo completo, embutido com o método de entropia cruzada (CEM), para registro tridimensional não supervisionado, modelando a tarefa de registro de nuvem de pontos como um processo de decisão de Markov. O modelo consiste em um módulo de rede de amostragem e um módulo CEM diferenciável. No módulo de rede de amostragem, dado um par de nuvens de pontos, a rede de amostragem aprende uma distribuição de amostragem prévia sobre o espaço de transformação. A distribuição de amostragem aprendida pode ser usada como uma inicialização ”boa” do módulo CEM diferenciável.

III. Objetivos

Este trabalho tem como objetivo a reprodução da execução de um algoritmo implementado e descrito por Adam Z. Wagner em seu artigo [3] ”Construction in Combinatorics via neural networks”. O experimento visa implementar uma IA que aprenderá a realizar uma tarefa, através de Aprendizado por Reforço, de como construir um grafo que contenha N vértices e qual o valor máximo de arestas possam ser adicionadas, sem que sejam formadas estruturas triangulares. Serão utilizadas redes neurais convolucionais (CNNs), que agirão como agentes atuantes no ambiente e que terão como resultado a construção do ambiente (grafo) a ser estudado, além de também utilizar-se do Deep Cross-Entropy Method, como forma de distribuição de probabilidades e método preditivo. Espera-se com isso obter-se diversas formas de grafos que se adéquem dentro de tais parâmetros e que através da análise de tais resultados, seja possível entender os funcionamentos das técnicas e métodos que foram abordados e utilizados, além de criar um discussão sobre a possibilidade de adequação da função Score para provar conjecturas contidas dentro da Teoria dos Grafos.

IV. metodologia experimental

O ambiente utilizado para a compilação dos trechos de código foi o Google Colaboratory. A línguagem de Programação utilizada foi o Python 3 e as seguintes bibliotecas foram utilizadas para compilação do código:

- networkx
- random
- numpy
- statistics
- pickle
- time
- math
- matplotlib.pyplot
- torch
- itertools
- numba

Funções de cada uma dessas bibliotecas foram utilizadas em algum trecho ou mais, do código. E a seguir falaremos um pouco sobre a base de dados ou a falta dela, já que o trabalho utilizou um ambiente artificial construído de maneira ”aleatória” no início da execução do programa.

A. Base de Dados

Por se tratar de um algoritmo de aprendizado por reforço, os dados de entrada, sendo esse o ambiente a ser construído (grafo), era primeiramente iniciado como uma String baseada em um alfabeto binário composto apenas pelos caracteres ’0’ e ’1’. Um grafo contendo N vértices, pode ser representado pela seguinte String de tamanho ((N*(N-1))/2). O primeiro valor a ser inserido nessa String é aleatório e os valores seguintes seguem o método de entropia cruzada, como distribuição pirobalística. Assim que uma dessas strings, que são chamadas de sessões, é finalizada, existe uma recompensa de uma função Score (a conjectura a ser testada), que atua como caixa-preta

e da qual as redes neurais não sabem como funciona e nem o que ela tenta resolver. Para cada iteração do jogo, são realizadas 5000 construções de strings e para cada iteração subsequente, apenas um pequeno percentual de pares de Ações e Estados e também de construções que atingiram um maior nível de recompensa são mantidos e reutilizados, sendo o resto descartado. Na iteração seguinte inicia-se o mesmo processo, tendo como inputs os dados herdados da iteração anterior.

B. Diagrama de Blocos

A Fig. 3. representa de maneira básica o pipeline de execução do código.

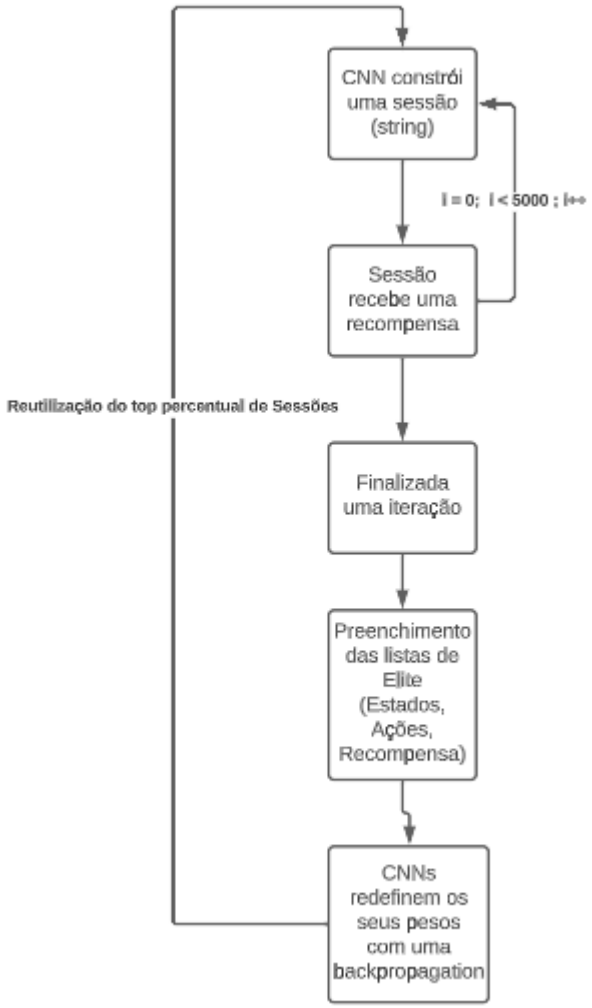

Fig. 3. Listas de Ações, Estados e Recompensas

Inicialmente são alocadas as listas que irão guardar os valores de elite dos Estados, Ações e Recompensas, nessa ordem, de iterações anteriores. Todas são alocadas vazias no início e são preenchidas novamente de acordo com os resultados de cada iteração. Nos testes realizados foram geradas as construções de 5000 strings (grafos) por iteração, e o número de iterações necessárias para atingir o score máximo varia pra cada estrutura gerada no input inicial. Após cada iteração e realizado o descarte de todas as sessões que ficaram fora do percentual selecionado (elite), as CNNs, que atuam como agentes nesse algoritmo, atualizam seus pesos, aprendendo com as construções salvas. O preenchimento das sessões iniciais da próxima iteração é realizado com construções da iteração anterior e ao fazer isso o algoritmo garante que a próxima iteração sempre será melhor em termos de recompensa que a iteração anterior. Os grafos são reutilizados, até que fiquem fora do percentual de elite da iteração atual e sejam eventualmente descartados. No final de cada iteração, a sessão que obteve o melhor valor de recompensa é transformada em uma matriz de adjacências e tem a sua estrutura de grafo plotada. Os valores de tempo estimado para execução de 5000 sessões e de tempo real de execução de 5000 sessões de cada iteração também são plotados, assim como o número de arestas contidas nas 10 construções com maior valor de recompensa na iteração atual.

V. RESULTADOS E DISCUSSÕES

Como o preenchimento da primeira sessão dentro da primeira iteração é realizado de maneira aleatória, a velocidade com a qual o algoritmo consegue chegar em uma estrutura de valores máximos de arestas para um dado grafo de N vértices, sem estruturas triangulares, depende muito dessas sessões iniciais.

O teste realizado e que aqui será exposto, foi feito para um grafo com 10 vértices. O valor máximo possível de arestas, sem estruturas triangulares, é de 25 arestas.

```
Tempo previsto: 0.4351258277893866, Tempo jogado: 4.651834593582153

0. Os melhores indivíduos: [18. 17. 16. 16. 16. 16. 15. 15. 15.]
Melhor matriz de adjacencia no passo atual:
[[0 1 0 0 0 1 0 1 0 0]
 [1 0 0 1 1 0 0 0 0 1]
 [0 0 0 0 1 1 0 0 0 1]
 [0 1 0 0 0 0 1 1 0 0]
 [0 1 1 0 0 0 1 0 1 0]
 [1 0 1 0 0 0 1 0 1 0]
 [0 0 0 1 1 1 0 0 0 1]
 [1 0 0 1 0 0 0 0 1 0]
 [0 0 0 0 1 1 0 1 0 1]
 [0 1 1 0 0 0 1 0 1 0]]
```

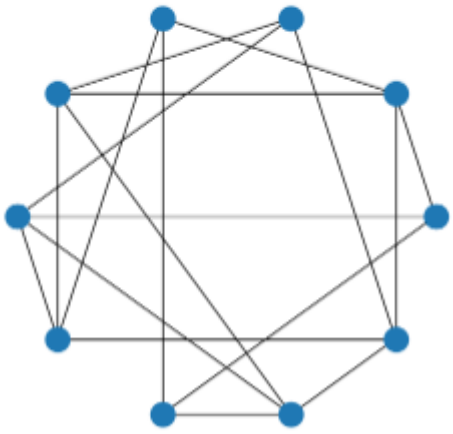

Fig. 4. Estrutura inicial do Jogo 01

É possível observar a aleatoriedade do preenchimento dos valores da matriz de adjacências que obteve o maior valor

de recompensa entre as 5000 construções dentro da primeira iteração.

```
Tempo previsto: 0.2748876915740967, Tempo jogado: 0.3944973945617676

5. Os melhores indivíduos: [18. 18. 18. 18. 17. 17. 17. 17. 17. 17.]
Melhor matriz de adjacencia no passo atual:
[[0 1 0 0 0 1 0 1 0 0]
 [1 0 0 0 1 0 1 0 0 1]
 [0 0 0 0 0 0 0 1 0 0]
 [0 0 0 0 1 0 1 0 1 1]
 [0 1 0 1 0 1 0 1 0 0]
 [1 0 0 0 1 0 1 0 1 1]
 [0 1 0 1 0 1 0 0 0 0]
 [1 0 1 0 1 0 0 0 1 1]
 [0 0 0 1 0 1 0 1 0 0]
 [0 1 0 1 0 1 0 1 0 0]]
```

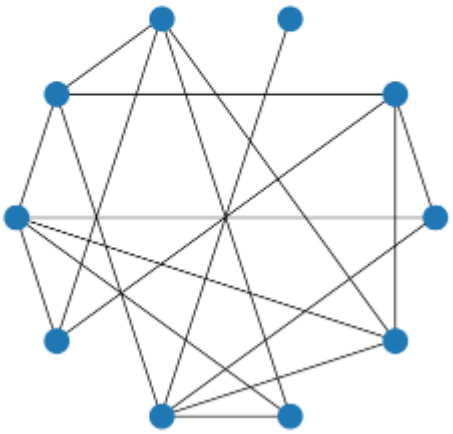

Fig. 5. Estrutura após a quinta iteração

Após 5 iterações e 25000 sessões construídas no total, obteve-se 4 sessões que alcaçaram estruturas com 18 arestas para 10 vértices. A Fig. 5. demonstra a melhor estrutura da iteração atual.

Com 25 iterações concluídas e com 125000 sessões construídas, já é possível observar na Fig. 6. uma certa ordenação na matriz de adjacências da sessão que obteve o melhor valor da função Score. Duas construções obtiveram 19 arestas sem estruturas triangulares. A partir deste ponto, o algoritmo começa a aprender de maneira acelerada a como atingir estruturas que garantem um maior valore de recompensa.

Apenas duas iterações depois, na vigésima sétima iteração, o algoritmo já alcançou uma construção com 20 arestas. A matriz de adjacências da Fig. 7. já está bastante simétrica em relação a sua diagonal principal, que é uma característica de quando as estruturas começam a chegar perto do valor máximo de arestas.

Por fim, na iteração de número 33 contida na Fig. 8. é possível observar que todas as 10 primeiras sessões com maiores valores de recompensa já possuem 25 arestas em suas estruturas. Sua matriz de adjacências também apresenta simetria em seus valores separados pela diagonal principal.

Como dito anteriormente, o tempo de aprendizagem do programa varia bastante e depende de como as sessões iniciais estão constituídas. O valor total do tempo de execução desse teste, partindo da iteração inicial e chegando até a trigésima terceira iteração, foi de aproximadamente 13 segundos e 59 milésimos. O mesmo teste, com o mesmo número de vértices,

```
Tempo previsto: 0.1781768798828125, Tempo jogado: 0.2299797534942627

25. Os melhores indivíduos: [19. 19. 18. 18. 18. 18. 18. 18. 18. 18.]
Melhor matriz de adjacencia no passo atual:
[[0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [0 0 0 1 0 1 1 1 0 0]
 [0 0 1 0 0 0 0 1 1 1]
 [0 0 0 0 0 1 1 0 0 1]
 [1 1 1 0 1 0 0 0 0 0]
 [1 1 1 0 1 0 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0]
 [1 1 0 1 0 0 0 0 0 0]
 [1 1 0 1 1 0 0 0 0 0]]
```

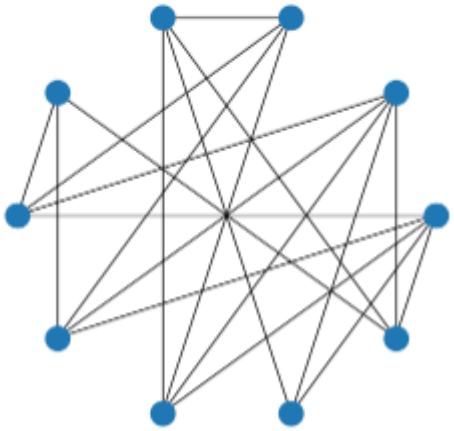

Fig. 6. Estrutura após a vigésima quinta iteração

```
Tempo previsto: 0.18652749061584473, Tempo jogado: 0.23720717430114746

27. Os melhores indivíduos: [20. 19. 19. 19. 19. 19. 19. 19. 19. 18.]
Melhor matriz de adjacencia no passo atual:
[[0 0 0 0 0 0 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 0 1]
 [0 0 0 0 0 1 1 1 0 0]
 [0 1 1 1 1 0 0 0 1 0]
 [1 1 1 1 1 0 0 0 0 0]
 [1 1 1 1 1 0 0 0 0 0]
 [1 1 1 0 0 1 0 0 0 0]
 [1 1 1 1 0 0 0 0 0 0]]
```

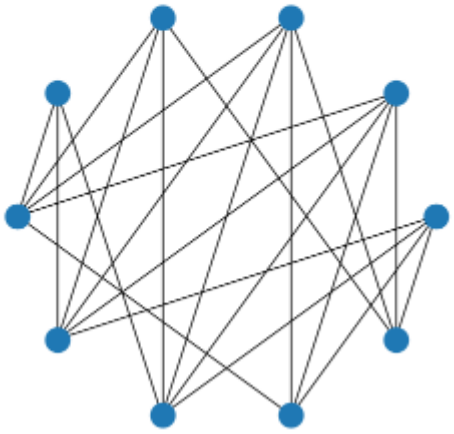

Fig. 7. Estrutura após a vigésima sétima iteração

```
Tempo previsto: 0.28857415962219238, Tempo jogado: 0.2442793846130371

33. Os melhores indivíduos: [25. 25. 25. 25. 25. 25. 25. 25. 25. 25.]
Melhor matriz de adjacencia no passo atual:
[[0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [0 0 0 0 0 1 1 1 1 1]
 [1 1 1 1 1 0 0 0 0 0]
 [1 1 1 1 1 0 0 0 0 0]
 [1 1 1 1 1 0 0 0 0 0]
 [1 1 1 1 1 0 0 0 0 0]
 [1 1 1 1 1 0 0 0 0 0]]
```

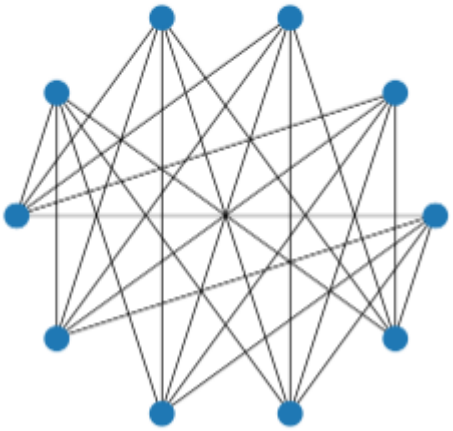

Fig. 8. Estrutura balanceada com o número máximo de arestas

porém realizado para uma sessão inicial completamente diferente, levou mais de 7 horas para obter 25 arestas.

VI. Conclusões e Trabalhos Futuros

Ao reproduzir um dos experimentos do pesquisador Adam Z. Wagner, foi possível concluir o quão poderosa pode ser a mistura de CNNs com métodos de aprendizado por reforço, tal qual o Cross-Entropy. Na atualidade existem diversos outros métodos algorítmicos que são mais famosos e muito mais utilizados que o Deep Cross-Entropy, porém ele provou ser uma ferramenta que tem como característica uma relativa facilidade em da compreensão de como utilizá-lo , não dependendo de algoritmos muito complexos ou de diversas políticas, além de demonstra uma eficaz capacidade de atuação sobre ambientes mais simples, como são os grafos. Para o futuro deste trabalho em específico, espero conseguir implementar diferentes conjecturas com o mesmo sucesso, e também adaptar outros métodos de aprendizado por reforço, para quem sabe, conseguir uma ferramenta que seja multi-uso não atuando apenas na Teoria dos Grafos, mas outras áreas da Matemática.

References

[1] DAVIES, Alex; VELICKOVIC, Petar; BUESING, Lars. et al. “Advancing mathematics by guiding human intuition with AI”, Nature 600, 70 – 74, 01/12/2021. https://doi.org/10.1038/s41586-021-04086-x

[2] M. Wen, U. Topcu, “Constrained Cross-Entropy Method for Safe Reinforcement Learning”, publicado em IEEE Transactions on Automatic Control (Volume: 66, Issue: 7, Julho 2021).

[3] WAGNER, Adam Z., “Construction in Combinatorics via neural networks”, Cornell University – 29/04/2021. https://doi.org/10.48550/arXiv.2104.14516

[4] LILLICRAP, Timothy P.; HUNT, Jonathan J.; PRITZEL, Alexander; et al. “Continuous control with deep reinforcement learning”, Cornell University – 09/09/2015. https://doi.org/10.48550/arXiv.1509.02971

[5] STIER, Julian; GRANITZER, Michael; “Deepstruck – linking deep learning and graph theory”. Elsevier. Software Impacts, 11. Fevereiro/22. https://doi.org/10.1016/j.simpa.2021.100193

[6] NARAYAN, Apurva; O’N ROE, Peter H.; “Learning Graph Dynamics using Deep Neural Networks”. Elsevier, Volume 51, Issue 2, 2018, 433-438. https://doi.org/10.1016/j.ifacol.2018.03.074

[7] SIMPSON, Carlos; “Learning proofs for the classification of nilpotent semigroups”. Cornell University – 06/06/2021. https://doi.org/10.48550/arXiv.2106.03015

[8] KAISER, Lukasz; BABAEIZADEH, Mohammad; MILOS, Piotr; et al. “Model-Based Reinforcement Learning for Atari”. Cornell University – 01/03/2019. https://doi.org/10.48550/arXiv.1903.00374

[9] WAGNER, Adam Z. “Refuting conjectures in extremal combinatorics via linear programming”. Elsevier, Journal of Combinatorial Theory, Serias A. Volume 169, 01/2020. https://doi.org/10.1016/j.jcta.2019.105130

[10] JIANG, Haobo; SHEN, Yaqi; XIE, Jin; et al. “Sampling Network Guided Cross-Entropy Method for Unsupervised Point Cloud Registration”. PCA Lab, Nanjing University of Science and Technology, China. 17/09/2021. https://doi.org/10.48550/arXiv.2109.06619

[11] HUANG, Gao; LIU, Zhuang; MAATEN, Laurens van der; WEINBERGER, Kilian Q. ”Densely Connected Convolutional Networks”. https://doi.org/10.48550/arXiv.1608.06993