Constructing Graphs with Machine Learning

Kira Hodge, Department of Mathematics and Statistics
Supervised by Dr. Natasha Morrison

Introduction

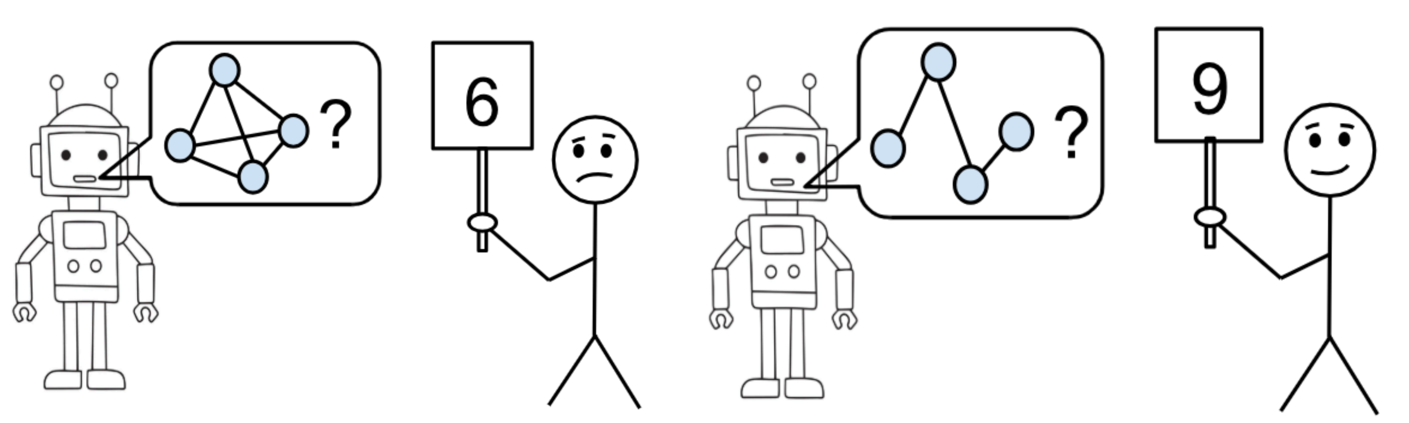
Software has been used in graph theory research for decades, but mainly with programs that allow users to create and edit graphs, and to perform simple calculations. In 2021, Wagner introduced a new method that revolutionized research by using machine learning to construct graphs [4].

The goal of this project is to adapt Wagner's method, using machine learning to create graphs that are counterexamples for existing mathematical conjectures.

Method

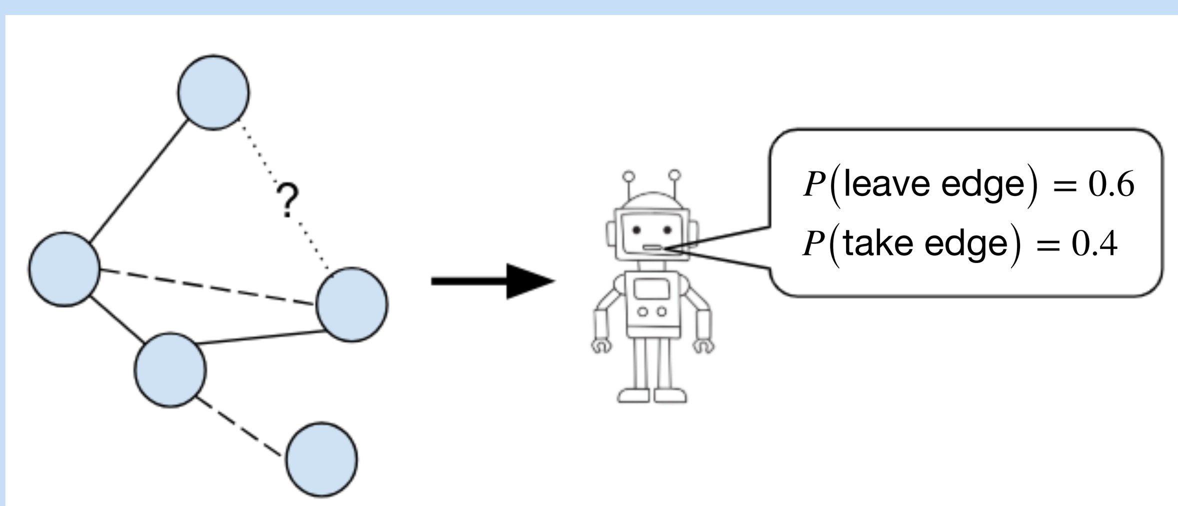
How does it work?

1. The AI agent plays a game, with the goal to score as high as possible
2. It creates a graph and receives a score based on how well the graph meets certain properties
3. The agent doesn't know these properties- it only sees the score
4. Through trial and error, it learns which graphs score higher, improving over time
5. The method succeeds in finding a counterexample if the agent finds a graph with a score exceeding the believed best possible



How does the model create a graph?

1. The model looks at the current state of the graph and the next possible edge
2. It gives the probability of either:
   I. Leaving the edge out
   II. Adding the edge to the graph
3. An option is picked based on these probabilities
4. The process is repeated, continuing to the next edge, until all edges have been looked at
5. The graph's reward score is then calculated



Why does it work?

1. **Explores beyond human intuition**: The AI agent can uncover unexpected patterns and counterexamples.
2. **Efficiently learns from feedback**: It rapidly tests possibilities, refining its approach based on scores.
3. **Easily adapts to different conjectures**: By changing the reward function the agent will create graphs with different properties.

Brouwer's Conjecture

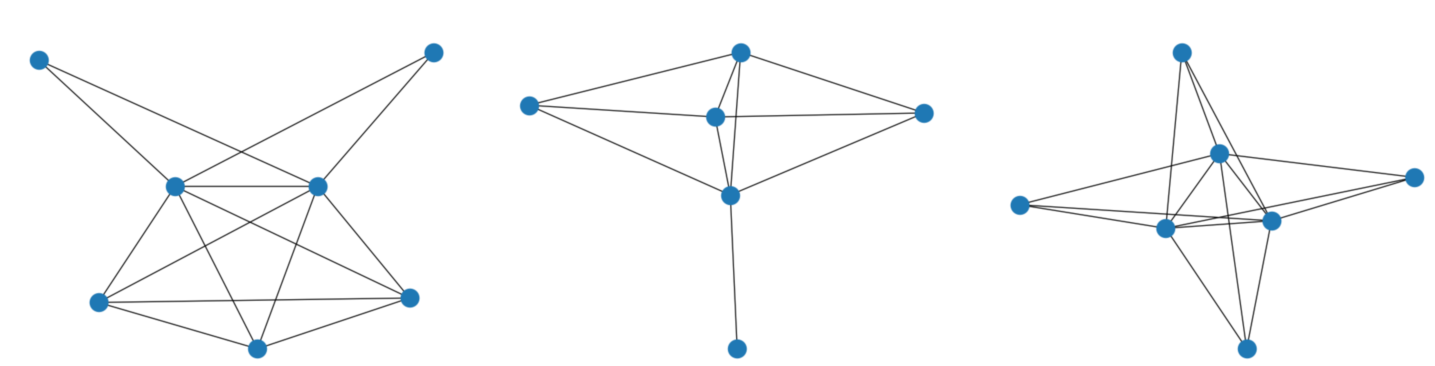
We apply this method in a novel setting to an important and well-studied problem in graph theory: Brouwer's Conjecture (2006) [3]. The goal is to find a counterexample as the conjecture remains unproven in general, despite holding true for certain cases.

Current research

The conjecture states that a certain graph parameter can never be larger than the number of edges plus a small constant. This problem has been approached using exhaustive searches of small graphs and by leveraging properties of specific graph classes. These efforts have significantly narrowed the range of possible counterexamples.

In particular, the graph parameter is believed to reach its maximum in threshold graphs. In this case, the conjecture has been proven and the graph parameter exactly equals the number of edges plus the constant [1].

Examples of threshold graphs:



Reward Function

The reward function scores each graph based on the difference between its graph parameter and the conjectured bound. Since the goal is to find a graph with a parameter exceeding the bound, a positive score indicates a counterexample has been found.

Existing research guides the agent by giving very low scores to graphs from proven classes. This causes the program to explore graphs that differ from those where the conjecture has been confirmed.

Preliminary Results

At first, the AI generated random graphs. As it learned, the reward score quickly increased before flattening as it neared 0, the expected best value.

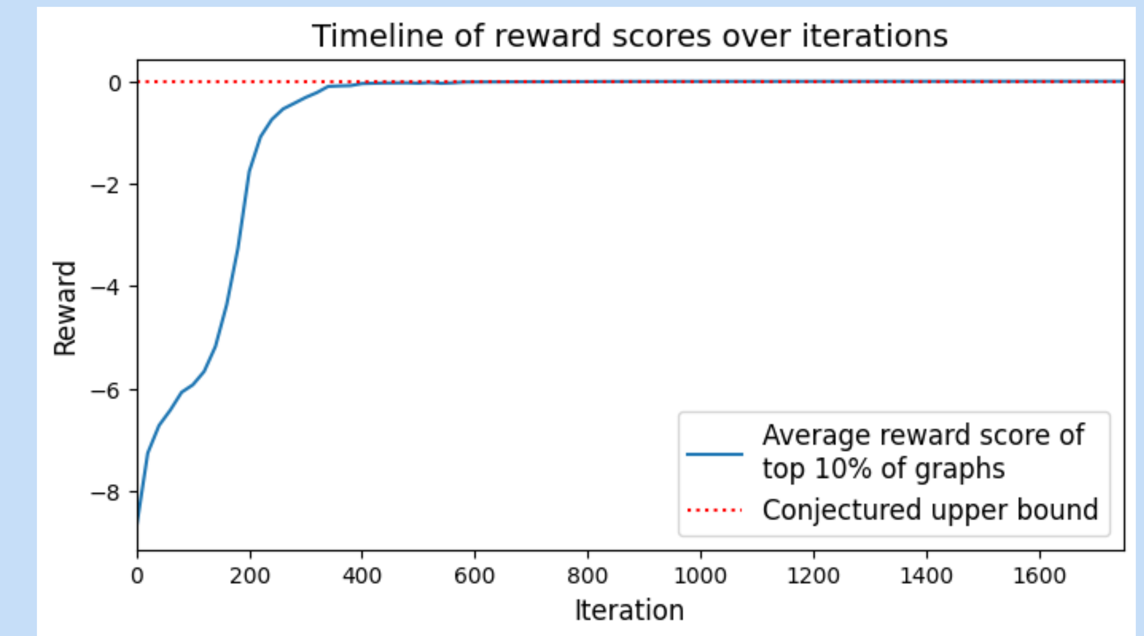


Fig 1. Mean reward score of best performing graphs

As the reward score converged to 0, the constructions closely resembled threshold graphs.

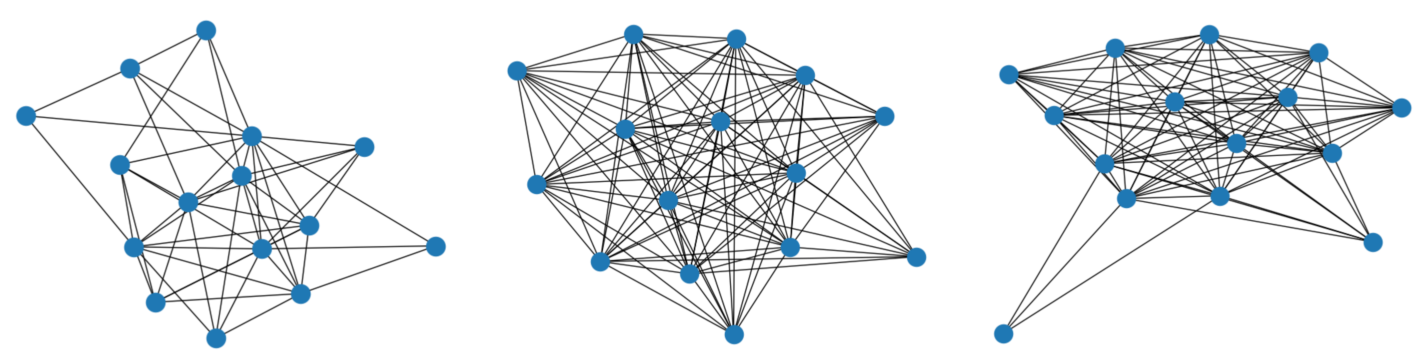


Fig 2. Timeline of best performing graph every 800 iterations

Updating the method

Based on the preliminary results, we refine the method by:

1. **Introducing randomness**: The agent occasionally makes random choices during graph construction. This helps it explore more possibilities and avoid getting stuck on suboptimal solutions [2].
2. **Adding more constraints**: The reward function penalizes graphs with certain properties, narrowing the search to more promising areas.

Future Work

1. **Refine the code** to improve its speed and efficiency.
2. **Apply this method to other conjectures**, including one from 1989 by Powers that upper bounds the i^{th} largest eigenvalue of a graph [3].

References

[1] J. N. Cooper. (2020). *Constraints on brouwer's laplacian spectrum conjecture.* https://doi.org/10.48550/arXiv.2003.03447

[2] M. Ghebleh, S. Al-Yakoob, A. Kanso, D. Stevanovi´c. (2024). *Reinforcement learning for graph theory*. https://doi.org/10.48550/arXiv.2403.18429

[3] L. Liu, B. Ning. (2023). *Unsolved Problems in Spectral Graph Theory*. https://doi.org/10.48550/arxiv.2305.10290

[4] A. Z. Wagner. (2021) *Constructions in combinatorics via neural networks*. https://doi.org/10.48550/arXiv.2104.14516

This research was supported by the Jamie Cassels Undergraduate Research Awards, University of Victoria

Feb 21, 2025