Using Reinforcement Learning to Develop a Better Strategy for Ramsey Games

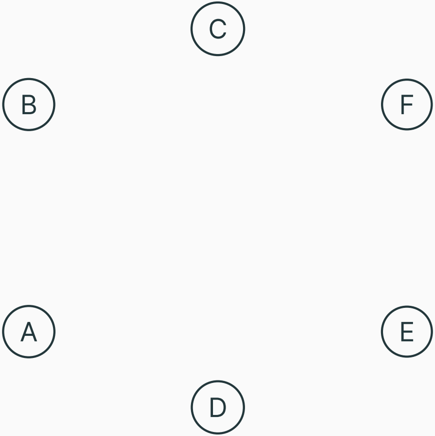
Kandasamy Chokkalingam
Mentored by Dr. William Gasarch

Poolesville High School

Background

Ramsey’s Theorem

Imagine you walk into a party with 6 people.



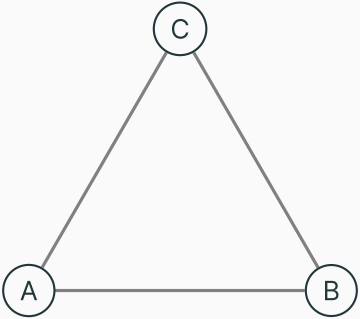
Ramsey’s Theorem

Every pair either knows each other or doesn’t.

Red edges mean they know each other, and blue edges mean they don’t.

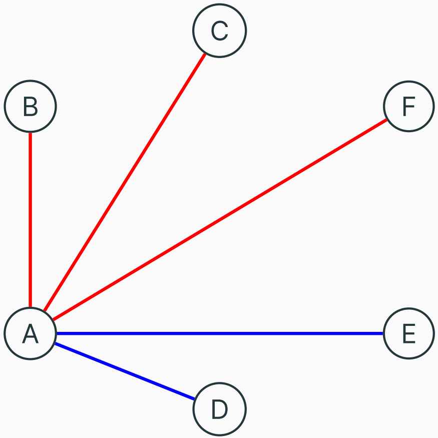
Ramsey's Theorem

For any possible set of relationships in this party, 3 people will either mutually know each other or not know each other.



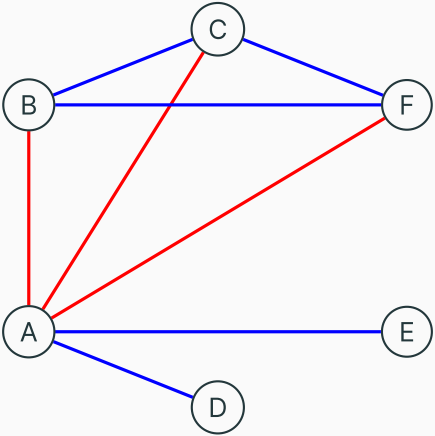
Ramsey's Theorem

Every person must either know or not know 3 people.



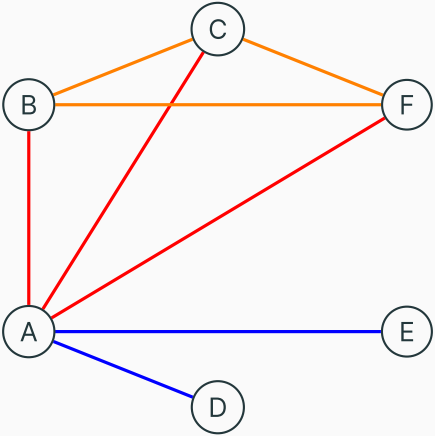
Ramsey’s Theorem

B can’t know C, C can’t know F, and F can’t know B.



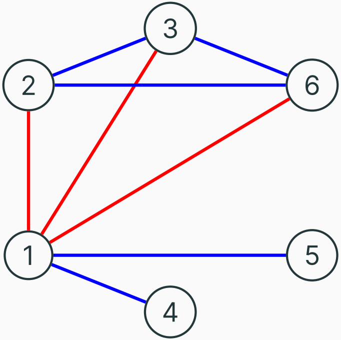
Ramsey's Theorem

We've just proven Ramsey's Theorem.



Ramsey Numbers

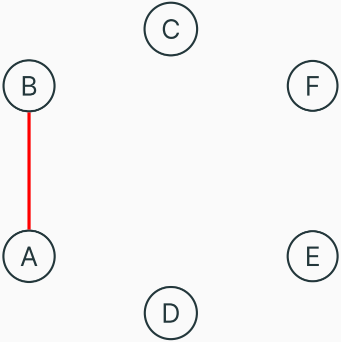
Ramsey Numbers are the minimum graph size needed to ensure a monochromatic group exists.



Bounds on all of these numbers aren't known, and are difficult to find.

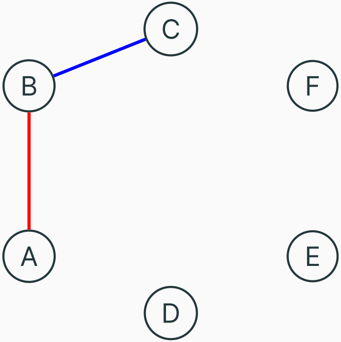
Ramsey Games

Players take turns connecting nodes on a graph.



If it always results in a win or loss, the graph size is a game Ramsey number.

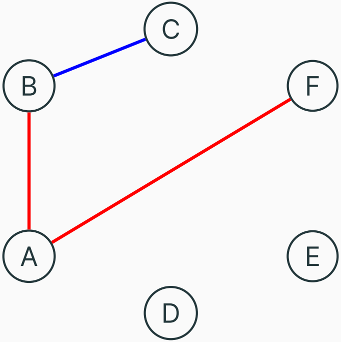
Players take turns connecting nodes on a graph.



If it always results in a win or loss, the graph size is a game Ramsey number.

Ramsey Games

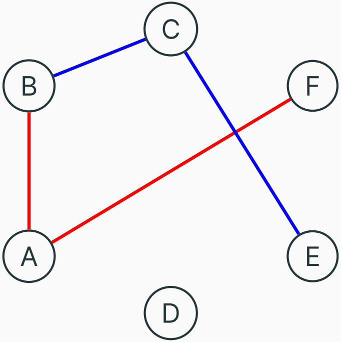
Players take turns connecting nodes on a graph.



If it always results in a win or loss, the graph size is a game Ramsey number.

Ramsey Games

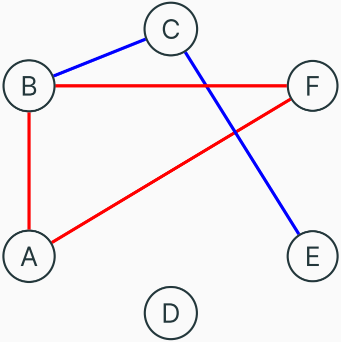
Players take turns connecting nodes on a graph.



If it always results in a win or loss, the graph size is a game Ramsey number.

Ramsey Games

Players take turns connecting nodes on a graph.



If it always results in a win or loss, the graph size is a game Ramsey number.

Reinforcement Learning

Models learn from playing by adjusting their policy.

They use the outcome of the game to identify the best moves.

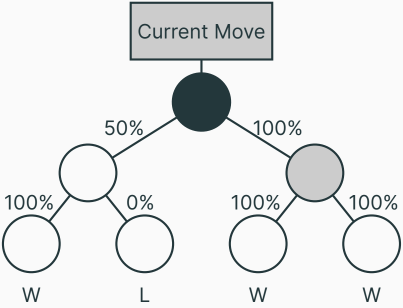
An ideal policy is one that will always produce the best move.

If a draw occurs, a graph size cannot be a game Ramsey number.

Monte Carlo Tree Search - MCTS

MCTS uses a random sampling of possible playouts.

It chooses the move that most often leads to a win.



Q-Learning

Q-Learning assigns a score to every state and action pair.

- Tabular Q-Learning - TQL
    - Stores each value in a table
    - Precise, but takes large amounts of memory
- Deep Q-Learning - DQN
    - Trains a neural network to predict values
    - Not as precise, but is memory efficient
- Graph Q-Learning - GQN
    - Uses GNNs to extract information about groups
    - Similar to DQN in performance

Methods

Model Specifications

Hyperparameters were tuned by performance with Ax.

MCTS had a maximum search depth of 4 moves.

TQL used a python dictionary to store q-values.

DQN used a 3 layer neural network.

Three types of GQN were used:

- GQN-1: With GAT-Conv Layers
- GQN-2: With EdgeConv Layers
- GQN-3: With Graph Convolutional Layers

Training Procedures

Models played against another model of the same type.

Models were tested on graphs of size R(3) and R(4).

Trained for 30,000 games, or at a maximum of 6 hours.

Multiple metrics were collected after every game:

- Number of moves
- Win rate
- Mean time taken to make a move
- Memory used

Testing Procedures

Each model played 500 games against a random agent 3 times.

For each trial, the number of wins was collected.

The top model for both versions was used to study the policy.

Results

Win Rate

The win rate represents how well the model is learning.

Table 1: Win Rate

Metric	R(3)	R(4)
Fastest Growing	GQN-3	DQN
Highest Value	TQL (92%)	DQN (88%)

Average Move Time

The move time represents the time cost of the model.

Table 2: Average Move Time

Metric	R(3)	R(4)
Lowest Value	TQL	DQN
Highest Value	GQN-1	MCTS

Number of Moves

The number of moves represents the performance of the model.

Table 3: Number of Moves

Metric	R(3)	R(4)
Lowest Value	TQL	MCTS
Highest Value	DQN	GQN-1

Memory Usage

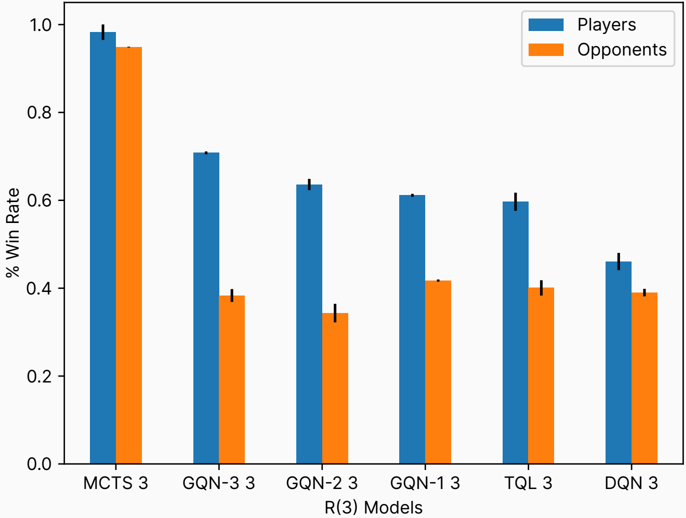
The memory usage represents the memory cost of the model.

Table 4: Average Number of Moves

Metric	R(3)	R(4)
Lowest Value	GQN-1	DQN
Highest Value	TQL	TQL
Most Consistent	GQN-3	GQN-1

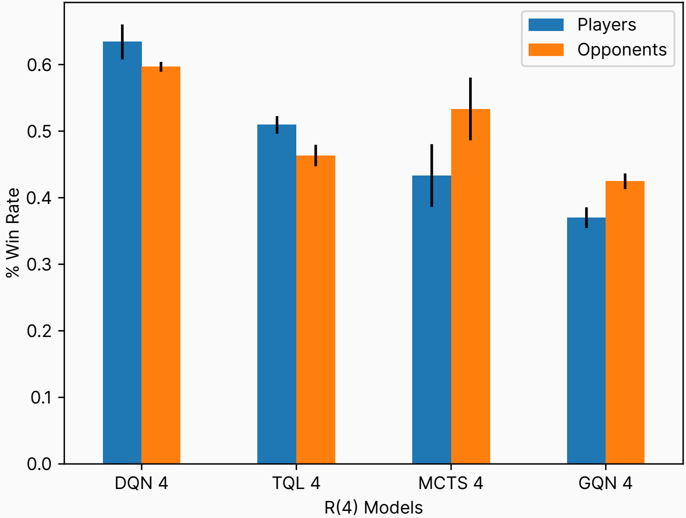
Random Player Win Rate

Fig. 1: Win Rate of Models against a random player on R(3)



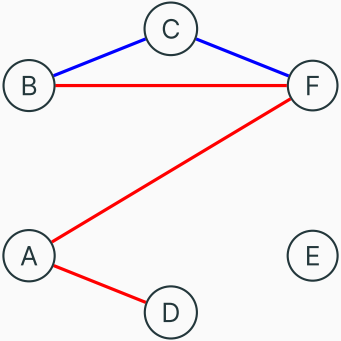
Random Player Win Rate

Fig. 2: Win Rate of Models against a random player on R(4)



Player Strategy

On R(3), the MCTS Player prioritized forking from the same node.



On R(4), more clusters of connections were present.

On R(3), 8/15 moves were played and 65/77 were played on R(4).

Conclusions

Conclusions

GQN models performed the best on R(3) and DQN on R(4).

MCTS showed high performance on R(3) but not on R(4).

Forking was done differently on R(3) vs. R(4).

This method is viable for learning how to play Ramsey games.

What’s Next?

Applying models to higher Ramsey number sized graphs.

Evaluating the performance of newer models that may be more suited for this task.

Acknowledgments

I'd like to thank Dr. William Gasarch and Mr. Joshua Twitty.

I'd like to thank my teachers, friends and family.

Questions?