Monopoly Project Report
CSE 519: Data Science Fundamentals

Table of Contents

1. Objective **2**

2. Post-midterm progress **2**

3. Strategic agent **3**

3.1 BMST decision 3

3.2 RespondTrade 4

3.3 Buy property 5

3.4 Auction property 6

3.5 Competitive Strategies used 6

4. Greedy agent **10**

4.1 BMST decision 11

4.2 RespondTrade 11

4.3 Buy property 11

4.4 Auction property 11

5. Random agent **11**

5.1 BMST decision 12

5.2 RespondTrade 12

5.3 Buy property 12

5.4 Auction property 12

6. Q-learning agent **13**

6.1 State space 13

6.2 Action space 14

6.3 Algorithm 14

6.4 BMST decision 15

6.5 RespondTrade 15

6.6 Buy property 16

6.7 Auction property 16

7. Statistics and Analytics **16**

7.1 Agent analysis 17

7.2 Successful strategies 17

8. Conclusion **19**

9.References **19**

Objective

The purpose of the project is to build a novel representation of the famous board game Monopoly backed by strategic agents which would make intelligent decisions on our behalf. Monopoly is often misjudged as a game of chance. Making the right decisions, buying appropriate properties and choosing the aggressive/defensive strategy is the crux of the game. After delving deep into the problem at hand and understanding the multitudinous game scenarios, our initial blueprint briefed upon the use of a Reinforcement Learning (RL) agent capable of playing against the baseline agents and learning the winning strategies. Accordingly our midterm progress talked about how we designed the state space, reward function for our RL agent and our plan to use OpenAIGym and build an extensive Q-table to record all these moves. We also drafted our baseline bots to be either Greedy or Random in their respective games so as to capture a broader range of preferable/winning moves at each state.

Post-midterm progress

Post midterm we worked on developing our primary agents i.e an RL and a Strategic agent to make one superior against the other. We developed 2 baseline agents: a Random and Greedy agent to test incremental strategies employed in each of our agents. We compared win percentages by running 100 games for all the pairs. Subsequently we analysed and implemented simple yet successful techniques to overthrow opponents and tried moulding our bots to utilise this new found knowledge primarily in our Strategic agent.

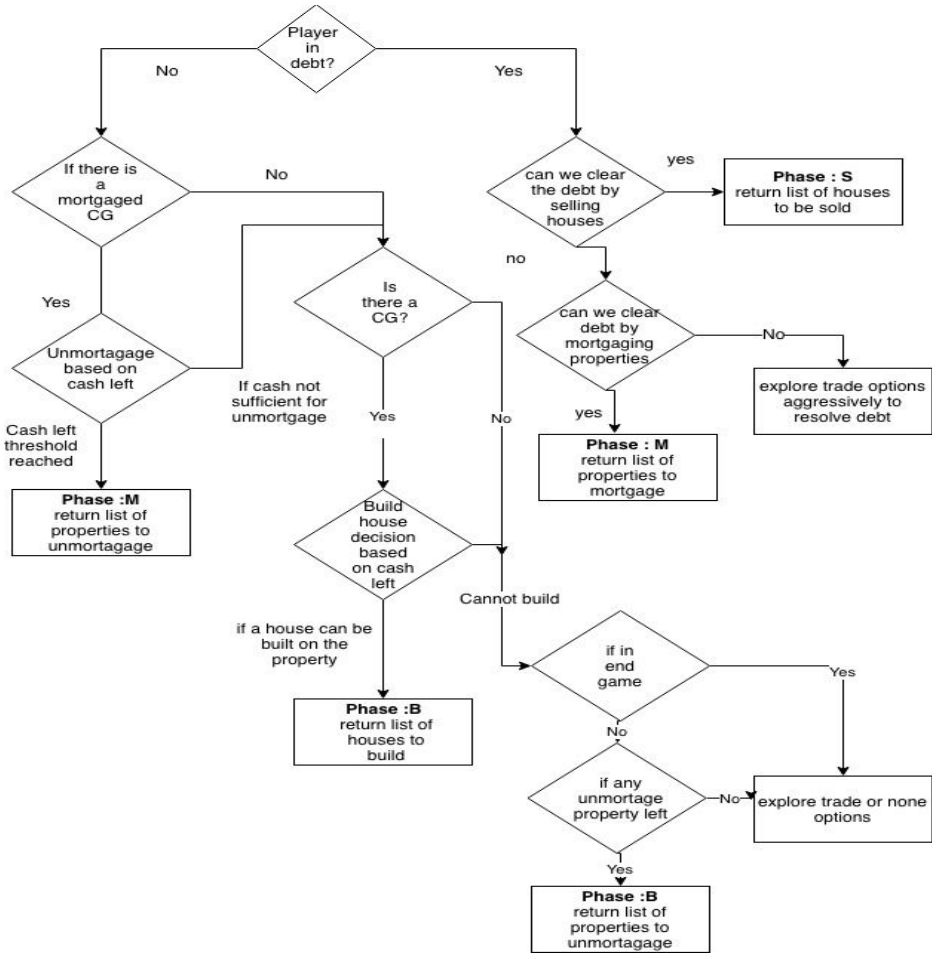
A major differentiating factor between a general monopoly game and the game in this project is that we end the game in a total of 100 turns and declare the winner to be the one with the maximum number of assets and the cash. As you read through this report, you will notice at various places that we used this to our advantage to not spend unnecessarily within the last few moves and acquire properties wisely. Few of our strategies also include the use of probability based decisions. This report speaks in depth about the implementation of the bots and the strategies we found to be useful for a game of this sort.

Strategic agent

Based on the numerous runs between the strategic agent and the greedy, random agents, the logs were generated and analysed to make optimal decisions for the strategic agent. The various phases of the bot have been explained with the use of flow charts below.

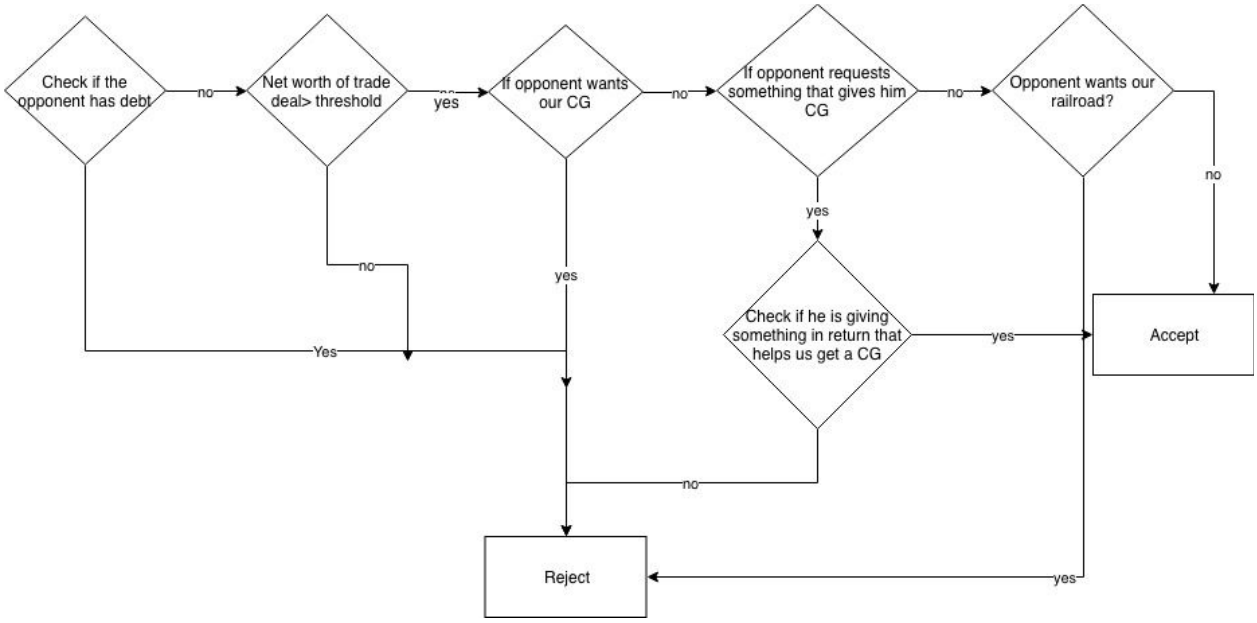
BMST decision

Over the duration of the game design, we realised that the BMST decision making is one of the most crucial decisions to make as this decision is invoked majority of the times in the given limited number of turns.This decision is based on several factors like the player being in debt or choosing to spend when he has money at hand; preferring when to build or unmortgage; versus when to sell a property, mortgage a property or propose a good trade deal based on opponent's behaviour to move out of debt. The flowchart below explains the route we have taken to tackle this complex decision making.



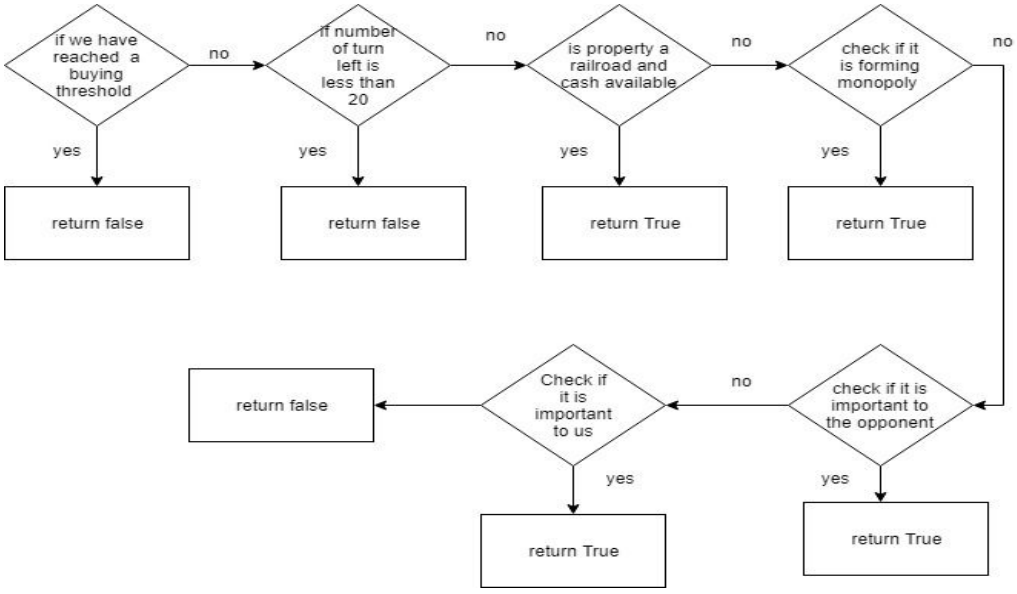
RespondTrade

Our strategic agent responds to an opponent's trade offer by taking into account several factors such as the opponent's debt, net value he gains from the trade, whether properties being requested and offered contribute to monopoly state and the class of the property i.e railroads. The agent chooses an optimal decision after considering all these factors as below.



Buy property

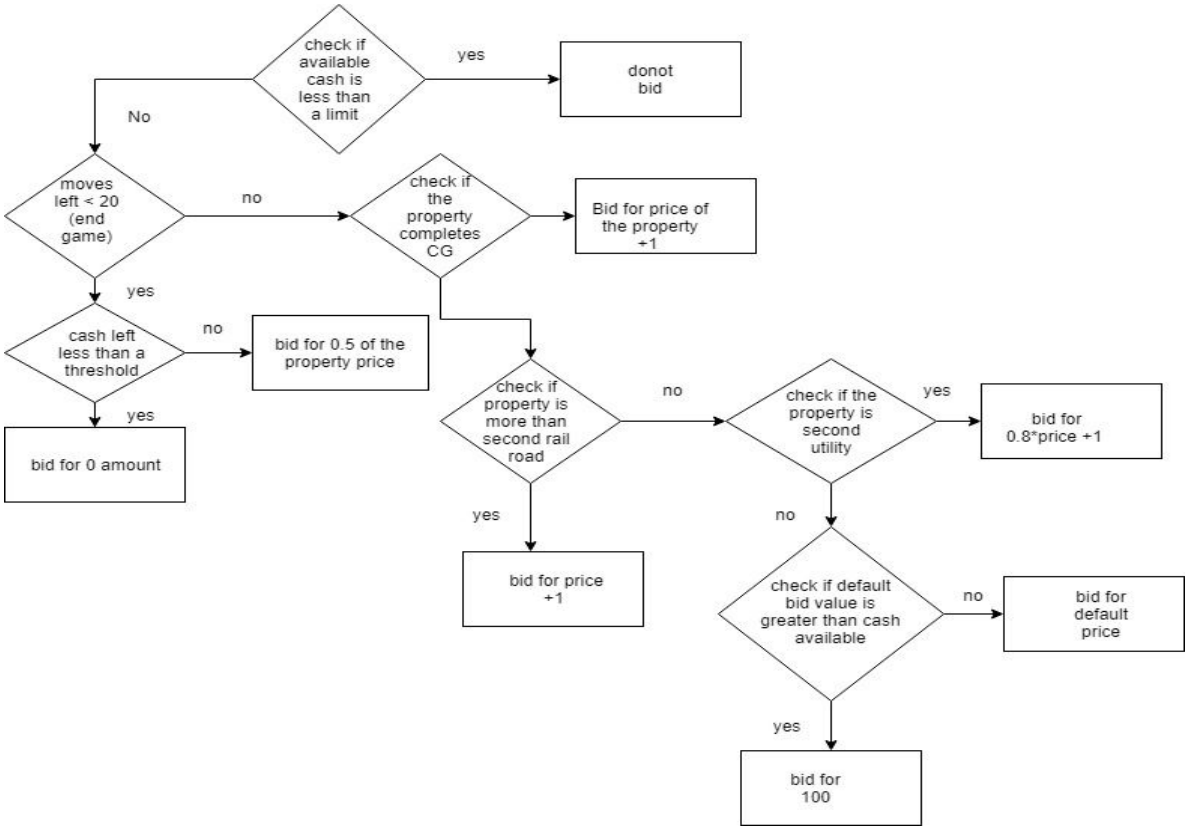
This decision is influenced by several factors like the cash resources at our disposal, the stage of the game, how important the property is to us in terms of the preference order we maintain across the game and lastly, if the property forms a monopoly for the opponent. Given these constraints the optimal buy strategy can be modeled as below.



Auction property

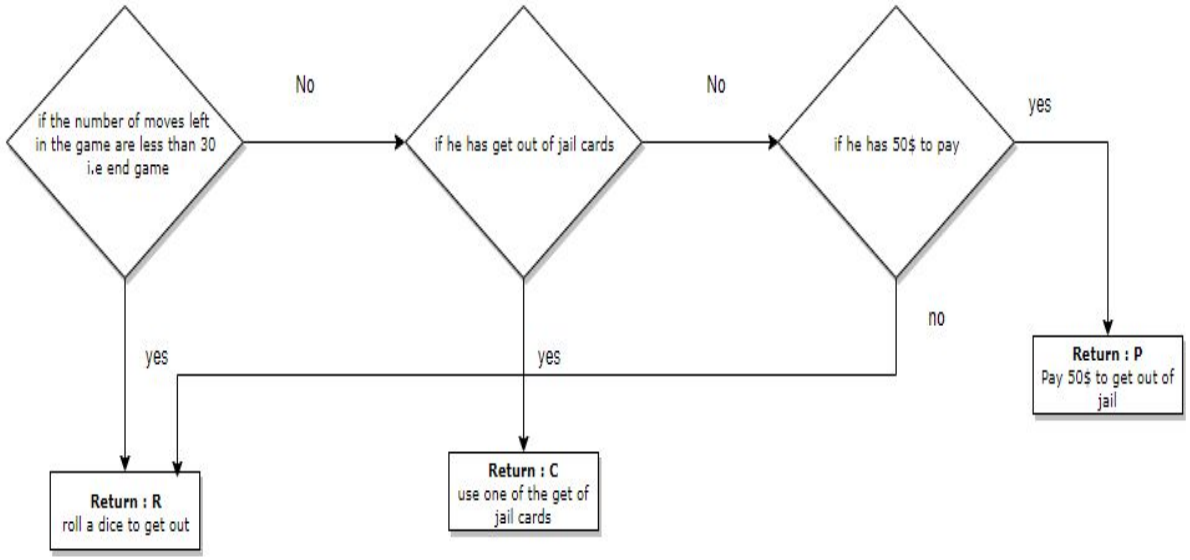
This decision needs to consider several constraints like whether the property being auctioned is important to the player, if we have enough cash resources to make a competitive bid, what stage

of the game is the auction invoked at. Given all of these constraints the auction property strategy is described as below.



Jail decision

From the logs it has been observed that it is not profitable to move around the board in the last phases of the game because it is very unlikely to buy new properties and the opponent landing on these squares is less probable too, so we prefer to stay in the jail. In the initial phases of the game we aggressively try to get of the jail by either using the get out of jail card or by paying 50$. This jail decision strategy can be visualised as below.



Competitive Strategies used

Useful Trivia: How frequently does a user land in jail?

There are several moves by which a user ends up landing in jail. Some of them are:

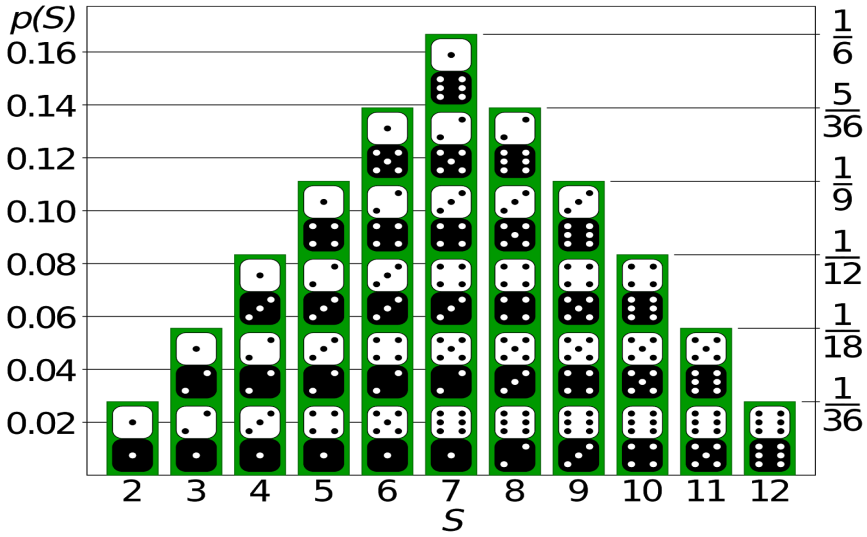
- rolling a double thrice.There are 6 doubles possible, namely (1 + 1, 2 + 2, 3 + 3, 4 + 4, 5 + 5, 6 + 6). In each turn there are 36 possibilities, therefore the probability of getting a double is ⅙. The cumulative probability of getting 3 doubles is ⅙*⅙*⅙ which comes out to be 0.46% but over a game of 100 turns this value increases manifold.
- land in the go-to-jail square
- Community chest card or a Chance card stating you to go to Jail.

Taking into account all the above possibilities, jail is one of the highly visited spots on the board.

Given the limited number of turns in the game we have proved that the following strategies work favourably:

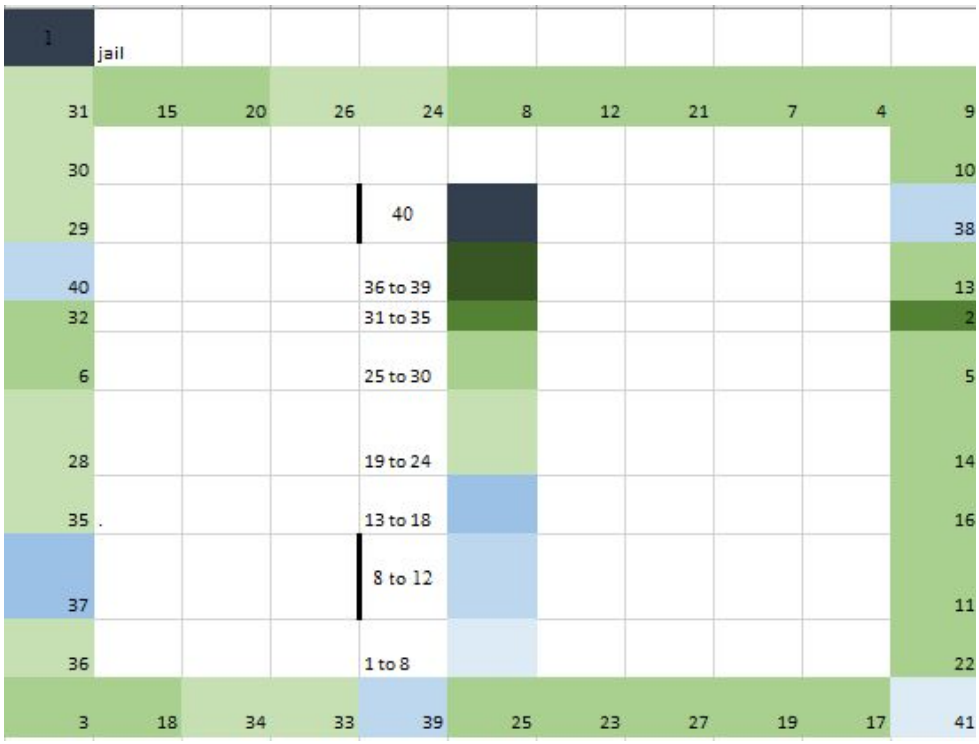
1) **Preference order of colour groups for making decisions based on property class**

Take a look at the diagram below, we can see that the probability of getting a 6, 7 or 8 is higher than any other combinations of the dice roll.



Therefore, we combined the trivia with the above logic to figure out that property squares at 6, 7 or 8 distance from the Jail Square is one of the most likely positions for a user to land on the board after coming out of jail. These squares happen to belong to the Orange color group. Using the same logic for the next move will land us on the yellow or red squares. Therefore our color group preference order for buying or building houses starts with Orange followed by Red and Yellow.

2) Number of visits at each square on the board



The above heat map is generated over 1000 turns and each square is numbered as per the rank while sorting according to the number of visits. Based on the above heat map we modified our preference order for buying the squares which are attracting high amounts of traffic.

3) How many houses to build?

Breakeven (# opponent rolls)

	Single Property	Last Property on Block	1st House	2nd House	3rd House	4th House	Hotel
Med/Bal	1,050.61	313.81	291.84	87.55	29.18	25.01	21.93
Or/Ver/Conn	702.31	173.97	111.01	36.32	11.71	16.02	14.53
StC/St/Vir	549.78	136.87	125.53	37.66	13.02	21.84	27.87
StJ/Tenn/NY	434.52	108.00	77.97	25.63	9.50	17.05	17.05
Ken/Ind/Ill	418.14	103.62	92.65	30.29	11.47	29.49	29.49
Atl/Vent/Marv	442.95	110.75	83.12	24.94	11.85	32.27	32.27
Pac/NC/Penn	442.51	110.63	92.84	28.23	14.71	38.48	42.07
Park/Board	381.01	120.52	81.64	23.59	12.38	35.56	35.56

	First Railroad	Second Railroad	Third Railroad	Fourth Railroad
RRs	264.61	87.10	32.56	13.01

	First Utility	Second Utility
Utilities	198.79	49.58

The above diagram illustrates the number of player turns needed to break even (i.e to get back the amount invested on a property). From the above table it can be inferred that building 3 houses is the most optimal way to get back the investment cost on almost all the Street properties.

4) Not un-mortgaging properties towards the end of the game

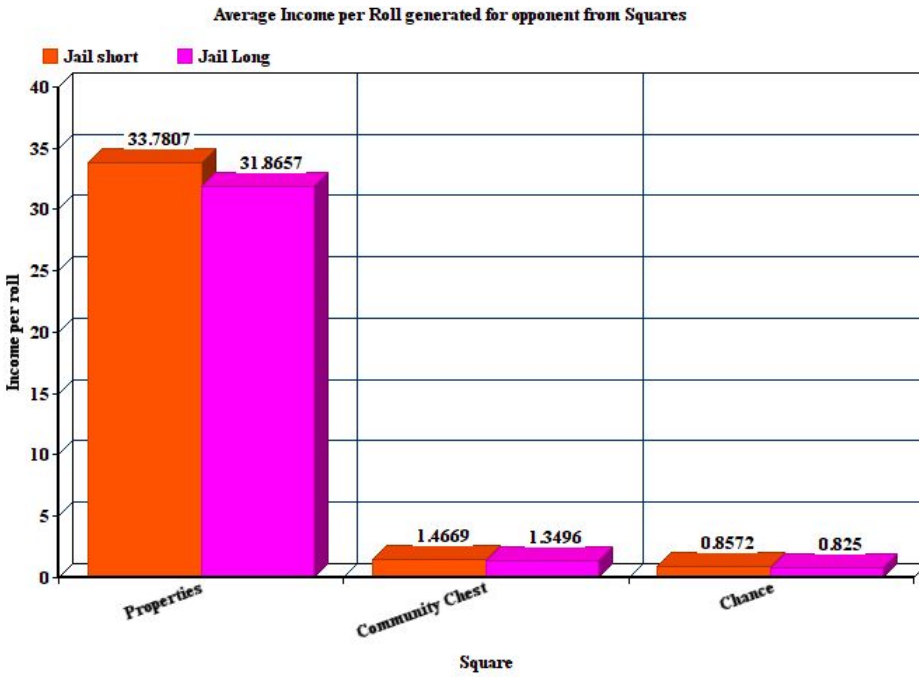
In order to unmortgage a property, we have to pay an additional 10% on the mortgaged price (50% of the original property price) of the property. Therefore, in un-mortgage we pay 55% of the property price and get the ownership back.
Consider a mortgaged property worth $100. Suppose my cash reserves = $100.

Un-mortgaging the property will result in my asset value being,
100 (property value) + 100 - 55 (my cash reserves) = $145.

On the other hand, if I chose not to unmortgage the property, my asset value would be,
50 (property value) +100 (my cash reserves) = $150.

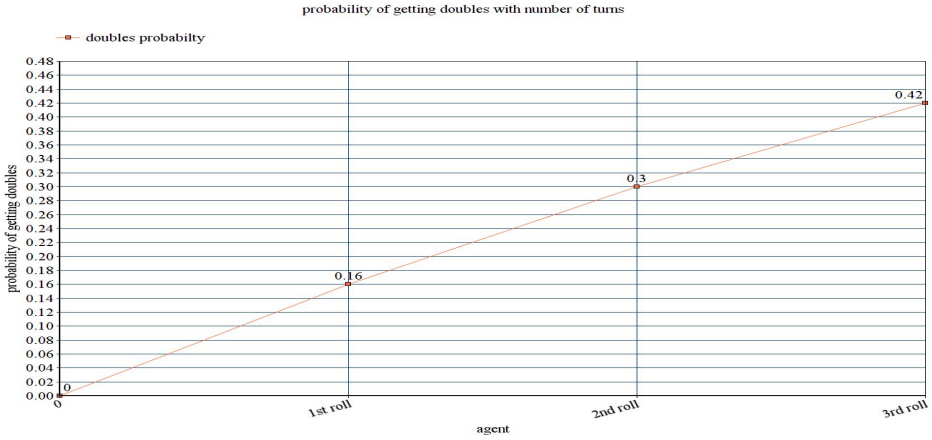
The 3.33% loss associated with un-mortgaging applies throughout the game. However it makes sense to un-mortgage in the initial stages of the game with the hope of gaining cash as rent when the opponent lands on the property or in the hope of forming a CG and building houses later. Towards the end of the game we can chose not to do this to maintain a higher asset value.

5) Prefer to Stay in Jail towards the end of the game



We plotted statistics to see the average income a player generated for the opponent when he preferred to stay in jail for long towards the end of the game. The income a current player generates for the opponent comes from paying rent by landing on opponent's properties or due to chance card or community chest cards that involve a transfer of cash. Our statistics account for the last 20 turns of the game. We therefore included this as a strategic decision in the bot. If the number of turns left in the game is less than 20, we prefer staying in jail to avoid generating income for the opponent.

6) Probability of getting out of jail in 3 turns



It can be observed that the probability of getting out of jail in 3 turns by rolling a double is ⅙ + 5 / 36 + 25/216 which is 42% .This strategy can be used when there is cash crunch instead of immediately getting out of jail by paying 50$.

7) Mortgage preference order

We mortgage properties in the following preference order:

- properties which are of less importance. These can be traded as mortgaged properties later on to win against a bot who might not be checking the status of the property he is accepting.
- properties between go and jail (except in the case we have cg's or we own multiple railroads). In the heat map above, we can see that these properties are less frequently visited
- single utility or single railroad because these properties are beneficial as a group and not individually.
- if we are still in debt we mortgage both the utilities and finally the monopoly groups as well

Greedy agent

The aim of this agent is to acquire as many properties as possible. It is greedy in that it takes all it's decisions with the sole aim of acquiring properties and building houses aggressively.

BMST decision

The greedy agent first looks at the debt the player is in.
If the player is in debt, it chooses first to sell houses due to it's greed for keeping ownership of its properties. It then tries to mortgage properties or trade as a last resort to clear debt.
If the player has no debt, first preference is given to building houses. If he is unable to build houses (i.e when there is no monopoly), the agent will unmortgage his mortgaged properties. If there are no properties to unmortgage, the agent again resorts to exploring trade options or returns None.

Build houses

Our greedy agent tries to maximise the number of houses it builds on each of its properties. It aims at building houses starting in order of the least build cost till there is cash left.

Sell houses

Our greedy agent starts selling houses in order of the least build cost till he clears his debt. Agent is trying not lose the most expensive houses that he has built till now.

Mortgage

Mortgage properties in order of the least value till you clear your debt. Agent is trying to not mortgage the most expensive properties.

Unmortgage

Unmortgage properties in order of the least value till you have cash left. Agent is trying to regain as many properties as he can.

Trade

Our greedy agent tries to trade to form CGs, get utilities and get railroads. In exchange, he offers any property that is not in his CG.

RespondTrade

In respondTrade, the greedy agent accepts all trades that are in net worth profitable to him. It calculates the net worth of the properties by adding the property prices being offered and the cash offered and subtracting the price of the properties being requested and the cash being requested. If it does not gain in the net worth it chooses to reject trade.

Buy property

The greedy agent buys all the properties that it lands on aggressively as long as it has sufficient cash.

Auction property

The greedy agent's aim is to win as many auctions as it can to gain properties. Therefore the bid amount is always equal to the price of the property (provided the agent has sufficient cash). If the agent does not have sufficient cash, it bids an amount equal to its current cash holdings by keeping a buffer (to prevent going into debt in the subsequent turns).

Jail decision

There is no such greedy decision involved in the jail decision for the greedy agent. Therefore this agent simply chooses an option of 'C' if it happens to have the get out of jail cards. If it doesn't have 50$, it chooses to roll out of jail, else it chooses 'P' or 'R' with equal probability.

Random agent

The aim of this agent is to return decisions randomly with uniform probability. The sole purpose of developing this agent was for testing win percentages of other smarter agents against it. We make sure here that the random agent does not return invalid actions.

BMST decision

If the player is in debt, agent first checks if the player has mortgaged properties. If he does, agent checks if there are houses to sell, if yes, a random action between ‘S’, ‘M’ and ‘T” is returned with the valid list of properties. If there are no houses to sell, a random action between ‘M’ and ‘T’ is taken. If player is not in debt, agent first checks if there is a CG and there are mortgaged properties, if yes it randomly chooses between ‘B’,’M’ and ‘T’. If there is a CG and there are no mortgaged properties, agent randomly chooses between ‘B’ and ‘T’. If there is no CG and there are mortgaged properties agent randomly chooses between ‘M’ and ‘T’. If there is no CG and no mortgaged properties agent chooses ‘T’.

RespondTrade

Agent accepts or rejects the trade with equal probability.

Buy property

If player doesn’t have enough cash we choose not to buy else agent buys or rejects the trade with equal probability.

Auction property

Agent returns a random bid amount in the range of 0 to current cash or property price whichever is least.

Jail decision

Jail decision between P and R is taken randomly if the player has the cash sufficient for it. If the player has a chance card agent returns C.

Q-learning agent

This agent employs Reinforcement learning to make better decisions at any point in the game. This agent reads the past history of states and maintains a qtable to make subsequent smarter decisions. The agent was developed using Python’s OpenAIGym library that provides API’s to encode RL environments. Specifically, we developed a custom environment RL_ENV and registered it with OpenAIGym. The rl_env has methods of env.reward, env.step, env.train that were developed.

State space

As described in the paper[2], we translate the state at any point to a vector of [area, finance, position]. Area is vector of 2x10 size, where 2 rows correspond to the two agents, and the 10 rows correspond to the 8 CGs + Utilities + Railroads. Each cell in the area vector represents the fraction of properties of that CG that are owned by that player. Therefore ranging from 0-1. The

finance vector consists of 2 values, specifying the current player's number of properties in comparison to those of his opponents' as well as a calculation of his current amount of money. Specifically concerning the first value, given players a, b and c and the number of properties they own as pa, pb, pc it will be pa/(pa+pb+pc). The second value in the tuple is cash owned by the player and can take many values. We bound this using a sigmoid function. The position variable determines the player's current position on the board in relation to its colour-group, scaled to [0,1] e.g. If the player is in the fourth colour group this value would be 0.4.

Action space

The Qtable is a table that stores q-values for every state [area, finance, position] against an action space of [Spend, Sell, Do nothing]

Qtable

State/Action	Spend	Sell	Do Nothing
[Area, Finance, Position]	...	...	...
[Area, Finance, Position]	...	...	...
...	...	...	...

Algorithm:

1. Reduce the state received to the vector [area, finance, position].
2. Choose randomly between exploring a new action vs reading the action with maximum Q-value from the qtable.
3. Whenever we read from the qtable we match the state we have from step 1 to the states in the qtable. Since the state space is continuous, we use approximation to find the closest state in the qtable. More specifically two states s1 and s2 can be considered as equal when all of the following statements are true:

    - $|(area[0,j]_{s_1} - area[0,j]_{s_2}) + (area[1,j]_{s_1} - area[1,j]_{s_2})| \leq 0.1, 0 \leq j \leq 9$
    - $|finance_{s_1} - finance_{s_2}| \leq 0.1$
    - $|position_{s_1} - position_{s_2}| = 0$

4. Once an action is chosen, calculate the new state after the action is taken and compute the reward using the reward function:

$$r = \frac{\frac{v}{p} * c}{1 + |\frac{v}{p} * c|} + \frac{1}{p} * m,$$

where

- p =2 i.e the number of players
- c is a smoothing factor
- v is a quantity representing the player's total asset value and is calculated by adding the value of all the properties in the possession of the player, minus the properties of all of his opponents
- m is a quantity representing the player's finance and is equal to the percentage of the money the player has to the sum of all players' money

5. We then calculate an updated qvalue as follows:
   `new_value = (1 - alpha) * old_value + alpha * (reward + gamma * next_max)`

   - Old value is the old qvalue against the state in the q table
   - Next_max is the maximum of the q values against the new state obtained after taking the action
   - alpha=0.1
   - gamma=0.6

6. The new_value is then updated against the current state and action chosen in the qtable. This is how the RL training occurs for the agent.

Translating the Q-learning logic to our agent's behaviour. Note that in each decision, we only consider Q-values of actions that are legal and that make sense for that phase in order to find the most profitable action.

BMST decision

If the action returned by the Q-learning algorithm is Spend, we use strategic agent logic and perform actions of Build house/Un-mortgage/Trade.
If the action returned by the Q-learning algorithm is Sell, we use strategic agent logic and perform actions of Sell House/Mortgage/Trade.
Else we choose to do nothing.

RespondTrade

Agent employes Q-learning to see the maximum q-value in the table following the pseudocode described above. If the action chosen by the pseudocode is Do Nothing, we chose to reject the trade. In all other cases, we accept the Trade.

Buy property

If the action chosen by the Q-learning algorithm is Spend, we return True, in all other cases we return False.

Auction property

If the action chosen by the Q-learning algorithm is Spend, we bid an amount equivalent to the property price, in all other cases, we bid an extremely low amount to lose the auction.

Jail decision

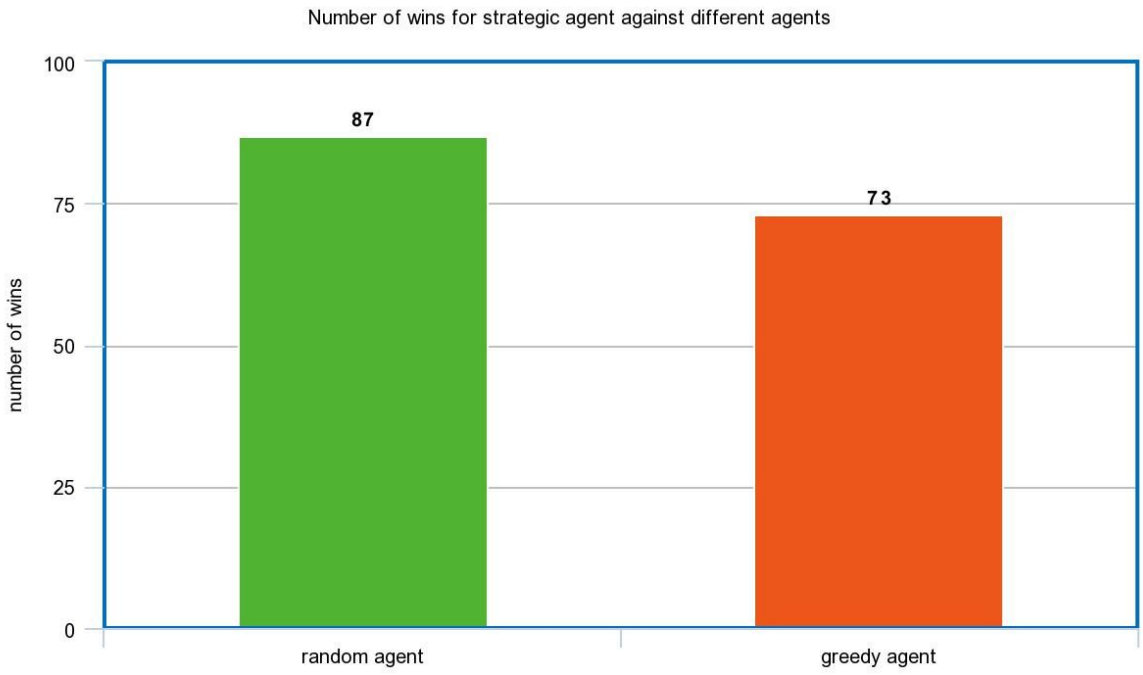
If player has a chance card agent returns C.If the player does not have a chance card, we look at the q-values of ‘Spend’ and ‘Do-Nothing’ to make a decision between P and R.

Challenges

The main challenge that we faced with the RL agent was dealing with an extremely huge and continuous state space. Over a single run of the game, a substantial amount of BMST calls were invoked that involved back and forth transfer and decision making on the basis of the state passed to the agent. Iterating the Q-table at each of these stages and finding the closest state to read the Q-values was an extremely expensive operation. Our midterm progress report highlighted our planned usage of OpenAIGym to help in our RL implementation. Subsequent understanding of the library showed that it provided APIs that allowed comparison of multiple RL agents and did not inherently have built-in capabilities for building and maintaining Q-tables. Our further evaluations consider the performance of the strategic agent as against other baselines.

Agent Analysis

1. **Win Percentage**



2. Successful strategies

2.1 Aggressively buying good properties

Property worth of agents though the course of a game

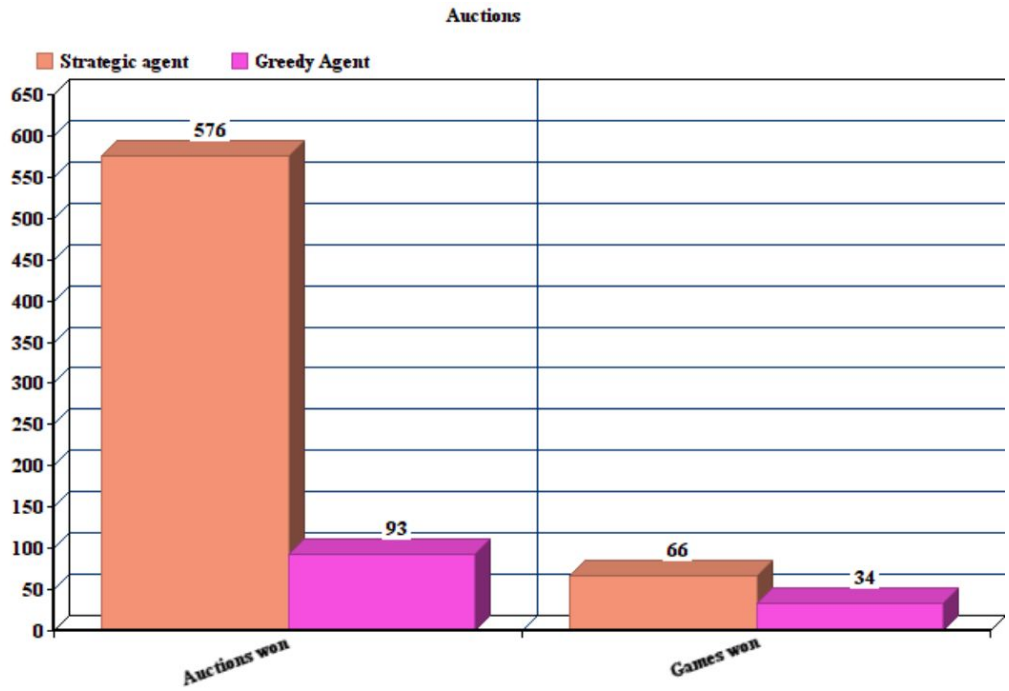
Here we aim to compare the buy logic used by the greedy and strategic agents. The above graph plots the total property worth of both the agents. It can be observed that since the strategic agent buys important properties aggressively the agent quickly breaks even and continues its buying streak resulting in a higher net worth as compared to the greedy agent. Therefore aggressively buying important properties helps rather that just buying all the properties blindly.

2.2 Playing aggressively in the beginning and conservatively towards the end of a game

Cash reserves of agents though the course of a game

As proved in the above graph, our strategic agent consumes its cash reserves aggressively at the beginning of the game by purchasing/auctioning the properties it deems to be important. Therefore, it achieves a breakeven phase at the center of the game whereas the greedy agent's cash reserves touches rock bottom. Towards the end of the game, the strategic agent applies a conservative strategy while conserving its resources whereas the greedy agent ventures out on the board when it is most likely that he may land on opponent properties and tend to pay heavy rent. Finally we see that at the end of the game, strategic agent possesses higher cash reserves than the greedy agent.

2.3 Smart auction strategies



Our agent employs interesting strategies in acquiring properties through auction. The strategic agent buys properties only when they belong to the Orange, Yellow and Red color groups. The agent considers all the other properties as less important and tries to acquire these through auctions by bidding them for a lesser amount. The greedy agent chooses to buy aggressively on the other hand and bids an amount equal to the property price if it has enough cash reserves to do so or else it bids an amount equal to its cash value by keeping a buffer. This causes the greedy agent to initially lose most of its cash reserves in buying properties and subsequently bidding low amounts in the auction due to low cash. The strategic agent wins a huge number of auctions and subsequently wins a large number of games when compared to the greedy agent.

Conclusion

A novel Strategic agent has been proposed and the performance of this agent was compared to baseline Random and Greedy agents. The results clearly demonstrate that the Strategic agent was able to outperform both the baseline agents with a good margin. The winning strategies have been explained in detail.

In the future, we plan to use a combination of the RL strategy for major decision making like the BSMT and the strategic bot for all other decisions by coming up with an optimal state space and overcoming the challenges faced in our Q-learning implementation.

References

1) https://www.inverse.com/article/32781-win-monopoly-using-math

2) http://doc.gold.ac.uk/aisb50/AISB50-S02/AISB50-S2-Bailis-paper.pdf
3) http://www.monopolycity.com/ac_monopoly_strategy.html
4) https://github.com/openai/gym-soccer/tree/master/gym_soccer/envs
5) https://www.learndatasci.com/tutorials/reinforcement-q-learning-scratch-python-openai-gym/