BLACKJACK SIMULATOR

A PYTHON BASED DASHBOARD APPLICATION

Kelly “Scott” Sims | ksims35@gatech.edu

Abstract

Blackjack is one of the most popular games at any casino. This is because of all games, it usually has the best odds of winning for any given player. In most casinos, the house’s edge is only 1 percent. In this work I have created a full fledge black jack simulator. It is executed in Python and results are displayed in a multi paged web based dashboard application. I then utilize this simulator to determine if it is in one’s interest to follow the same playing strategy that is imposed on the dealer by house rules.

1. Motivation

A standard casino in Las Vegas has approximately 15 general games. Each can have their own spin off versions with unique rules. However, no other game gives the best odds of winning than Blackjack. Most casino’s rules give the house only a 1% edge over the ordinary player. Because of this, Blackjack is one of the most, if not the most popular gambling games at casinos. The rules are simple, you are initially dealt two cards and the dealer is dealt two cards. It is important to note that you can only see one of the dealer’s two cards. The objective is to reach a total sum of cards in your hand that is greater than the dealer’s total sum, without going over a cumulative total of 21. All number value cards are worth their actual number. All faces cards are worth 10 points, but aces are special. They can be worth either 1 point or 11 points depending on whichever the player chooses. After the cards are dealt, a player can request to take a “hit”, which signals the dealer to deal then an extra card. Or the player can request to “stand”, which signals to the dealer that the player is satisfied with their current total. If at any time a player requests a “hit” and the dealt card brings their total to over 21, they have gone “bust” and they automatically lose. Once a player elects to stand, the dealer then revels their second dealt card, thus showing their current total. Most house rules say that the dealer must continue to hit until their total is 17 or higher. If at any time their total goes over 21, they lose and the players sitting at the table, if they have not gone bust, automatically win. In the following sections, I will briefly describe my implementation of the blackjack simulator, as well as further explore if it is wise for a player to follow the house strategy of “17 or higher”

2. The Simulator

The simulator I created was written in Python 3.7 and is a full scale dashboard application. It primarily uses Numpy, Plotly, and Dash modules to execute configured simulations. It was implemented in Object Oriented Programming style (OOP) and gives the user the ability to create many different scenarios to be simulated. There is a complete user’s manual in the project itself that goes into further details on how to setup and run the simulation, as well as the description for all parameters and configurables that exist. For the sake of brevity in this paper, I will only discuss high level details and refer the reader to the user’s manual for further details.

2.1.1 Configuration YAML

All simulations need to be configured prior to running. In the main directory of the project folder exits a config.yml file. This is where a user can define many rules and strategies for the game to be simulated. Things such as table minim and maximum bets allowed, the number of other players sitting at the table, the number of starting betting chips, and even the betting and playing strategies are configurable in the config.yml file. The user’s manual goes into full detail on how to configure each parameter as well as any extra constraints that may or may not be passed to certain variables. The table below shows everything that exists in the config.yml as well as the description for each.

Parameter	Values	Description
RANDOM_SEED	Any Valid Integer	Sets a random seed generator for statistical repeatability
NUMBER_OF_TRIALS	Any Integer > 0	How many times to repeat the simulations
NUMBER_OF_DECKS	{1,2,3,4,5,6,7,8}	The number of standard 52 card decks to be used at the table. 1 is the minimum, resulting in a card shoe of 52 cards to deal from. 8 is maximum, resulting in a card shoe of 416 cards to deal from.
SHOE_CUT_PERCENTAGE	0.0 <= X <= 1.0	This specifies what percentage of cards must be dealt from the card shoe before the dealer reshuffles the deck. For example, a value of 0.75 means that for a shoe utilizing 8 decks of cards, 312 cards will be dealt before all cards are shuffled and reintroduced to the shoe to be dealt again.
TABLE_MIN_BET	0 < X	The required minimum bet to play at the table. Any valid integer greater than 0 is allowed
TABLE_MAX_BET	TABLE_MIN <= X	The maximum allowable bet at the table. Any valid integer greater than TABLE_MIN_BET is allowed
PLAYER_NAME	<string>	The unique name to give your player at the table to disambiguate from any other players that may be sitting at the table
NUMBER_OF_HANDS	0 < X	The maximum number of hands to be dealt for each simulation trial. Each trial will progress until either the maximum hand has been reached or your player has gone broke. After either of these criteria have been met, everything will be reset and the next trial will commence
CHIPS	0 < X	The amount of chips every player sitting at the table will start with for each simulation trial
BETTING	See ***Strategies*** section	The betting strategy you wish to use for the simulation trials
PLAYING_STRATEGY	See ***Strategies*** section	The playing strategy you wish to use for the simulation trials
OTHER_PLAYERS	See ***Other Players*** section	Betting and Playing strategies for any active simulated players sitting at the table with your player.

Table 1 Table from the User's Manual highlighting all simulation configurables and their description

Once a simulation is configured, it can be ran by executing the main.py script. Please refer to the user's manual for more details. After it has finished running for all trials, a web based application should launch, revealing many graphics and statistics from the resulting sim. If for whatever reason it does not launch, a user can simply open a web browser and navigate to http://127.0.0.1:8050/. This is the localhost port in which the server is running.

3. Copy the House?

For the rest of this paper, I explore whether it is in a player's best interest to copy the house's playing strategy, as well as trying to identify the best betting strategy to maximize profits. As stated earlier, most house rules dictate that the dealer must continue to "hit" until their total is 17 or higher. As an anecdotal example, if the dealer's first two dealt cards are a 3 and Jack, their total is 13 and they must "hit". Conversely if their cards are an Ace and a 9, their total is 20 (using the ace as an 11 instead of 1) and the dealer must "stay". There are other concepts to the game which weren't previously covered, such as doubling, splitting, and surrendering. However, with respect to the dealer, they are not allowed to do this anyways, so these features were not implemented.

3.1 Do You Win More?

As stated earlier, in most casinos, the house odds is only at 1% advantage. Therefore, you'd expect to see a win/loss ratio, momentarily ignoring the betting portion, very close to 50% if it is just you and the dealer playing at the table. To test this out, I ran a simulation with the following parameters:

Table Min	$100
Table Max	$1000
Number of Other Players	0
Number of Simulation Trials	2000
Number of Playing Decks	8
Shoe Cut Percentage	0.75
Number of Hands per Trial	50

Where the first 3 constrains are self-explanatory. The number of playing decks, with a value of 8, means that 8 decks of a standard 52 card deck was used at the table for a total of 416 cards. This is done by casinos to make it harder for players to "count cards" at the table. The cards are kept in an object called a "shoe". Once cards are dealt from the shoe, they are not returned, meaning after each game, the amount of cards to be dealt from, reduces. This is true until the shoe cut percentage constraint has been met.

Typically at the start of a fresh table, all cards are put in the shoe, and then the dealer randomly places a "cut card" somewhere between the cards in the shoe. Once all cards have been dealt to the point where this cut card is revealed, the dealer then reshuffles all the cards and reintroduces all them back to the shoe. In the simulation, the shoe cut percentage mimics this cut card. For my parameter of 0.75, this means that 75% of the 416 cards must be dealt out before all cards are shuffled and reintroduced back into the shoe.

Finally, the number of hands per trial parameter signals that for each of the 2000 trials, a trial ends whenever a player has played 50 hands or the player has gone broke (lost all their chips), whichever comes first. In the event that a player never goes broke, in this scenario, a total of 100k games will be played across all simulation trials. To facilitate efforts in the simulation so that the player doesn't go broke, I set the betting strategy such that the player will always bet the table minimum to keep losses to a minimum. Figure 1 shows, as expected, that the win/loss percentage is slightly above 50%.

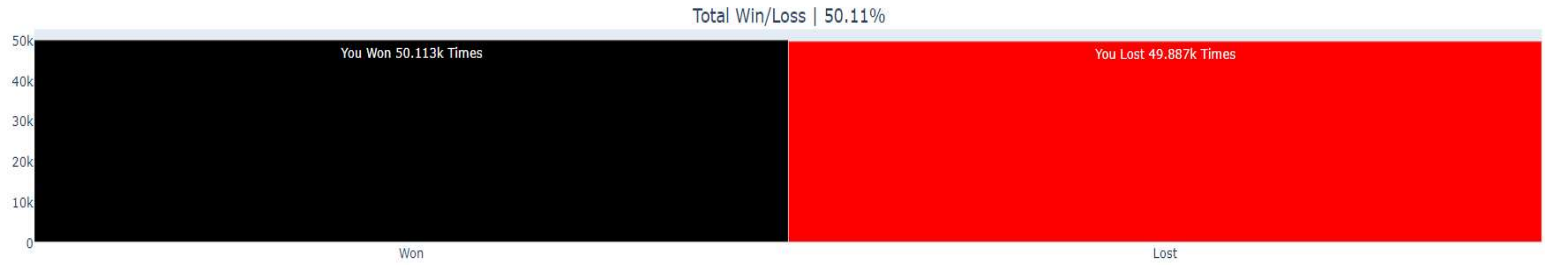


Figure 1 Dashboard output from the simulation showing at win percentage of 50.11%

This is one of the outputs from the analytics dashboard generated from the simulation. We can see that 50.113k games were won out of the 100,000. We can also see from another analytical output from the dashboard in Figure 2 that following the house strategy puts a player at risk of busting about 31% of the time. Recall that a player has gone bust if their cards sum to any value greater than 21.

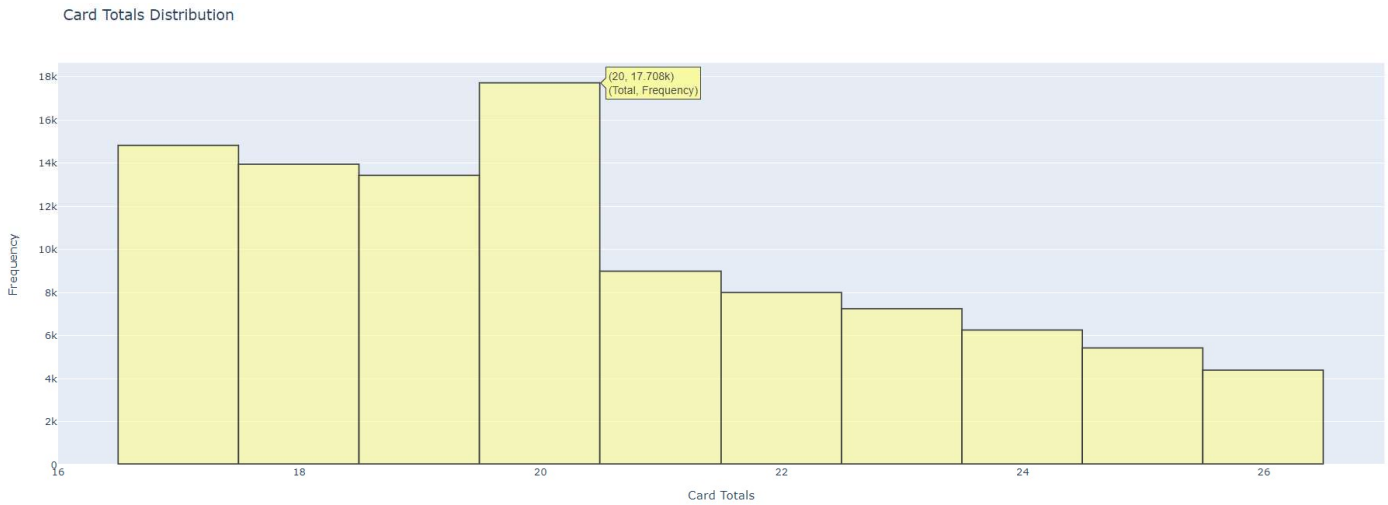


Figure 2 Dashboard output from the simulation showing the frequency of specific card totals per game. A total of 20 was the most prevalent, happening 17% of the time. Any value greater than 21 (bust) accounted for 31% of the outcomes.

It also reveals interestingly enough, that you're more likely to get a total of 20 more often than any other total at a rate of 17%. Sadly, the winning value of 21 happens only at rate of 9%. But this is only if it is you and the dealer sitting at the table. I reran the simulation 7 more times, each time adding an extra player, to see how it affects win percentages. For each player added, I assigned them both a random playing

style as well as a random betting style to introduce more stochasticity to each trial. It is also an attempt to mimic a real world situation where there can be varying-experience level players sitting at the table with you. The table blow shows the results.

Scenario	Win Percentage (%)
Dealer + You	50.11
Dealer + You + 1 Other	50.01
Dealer + You + 2 Others	49.69
Dealer + You + 3 Others	50.23
Dealer + You + 4 Others	50.11
Dealer + You + 5 Others	50.16
Dealer + You + 6 Others	49.90
Dealer + You + 7 Others	50.01

Table 2 Win Percentage using the dealer's playing strategy with different amounts of "other players" at the table

We can see, pretty consistently, regardless of how many people you have sitting at the table, and what their strategy is, you can still expect very close to 50/50 odds on winning at blackjack if you follow the house rules for the dealer. In all the above scenarios, the bust percentage still hovered around the 30-31% mark as well

3.2 Minimize Risk of Busting

As we saw in the previous section, following house rules, we can expect to lose almost 50% of the time. Approximately 60% of those losses were due to a player going bust. So in an attempt to reduce the risk of busting, I've modified the house rules of standing on totals of 17 or higher, to stand on values of 16 or higher, to see how much that reduces the chance of going over while still trying to score higher than the dealer. I reset the simulation to just the single player and the dealer, no other players and reran the simulation with the modified betting strategy. The figures below show the results of the simulation.

Figure 3 Dashboard analytics for simulated run with playing strategy of standing on 16 or higher. The results do show a decrease in bust rate, but not a significant jump in win rate

We can see that the win % has increased ever so slightly to 50.42%, but this might be an insignificant increase. More trials should be ran with other players before making this conclusion. However, the bust percentage did decrease significantly. We have gone from 30-31% to now approximately 22% bust rate. This is significant, however with the win percentage being what it was, it appears that we have merely traded losses due to going bust with more losses simply because the dealer scored higher than us. So initially there doesn't seem to be an extreme advantage in picking 16 or 17 as the cut-off criteria. Out of pure curiosity, I ran the simulation again, using 18 as the cut-off criteria. As one would expect, it isn't a great strategy. The win percentage dropped significantly to 47% and the bust rate jumped up to over 41%. For completeness, I ran the simulation on the remaining strategies I created. The full explanation for each strategy can be found in the user's manual.

Playing Strategy	Win Percentage (%)	Bust Percentage (%)
Stand on 16 or Higher	50.42	22
Stand on 17 or Higher	50.11	31
Stand on 18 or Higher	47	41
50/50	33.97	28
D+10	47.02	28.5
Always Stand	41.29	0

Table 3 Win Percentage using various playing strategies

Overall, it appears that standing on sums of 16 or higher and 17 or higher is in a player's best interest. The house is on to something with their dealer strategy.

4. Maximize Profits with House Strategy

Now that we have identified what appears to be the best strategy to use (stand on 16 or higher), in this section, we work on methods of maximizing profits while playing with this strategy. In the dashboard analytic after running a simulation, there is a tab called "Chip Total Per Hand". This graphic shows all player's chip total after every single hand. The separate trial runs are divided by vertical black lines. The starting chip total as configured by the user is represented by the horizontal dashed line. In instances that chip total curves are above the dashed line, this means the player has positive winnings. Anytime it is below, this signifies a player has negative winnings (losing money). For simulations that have several hundreds of trials (or more), give this window some time to load as it has to render several SVG objects in the browser. Figure 4 shows an example of this graphic

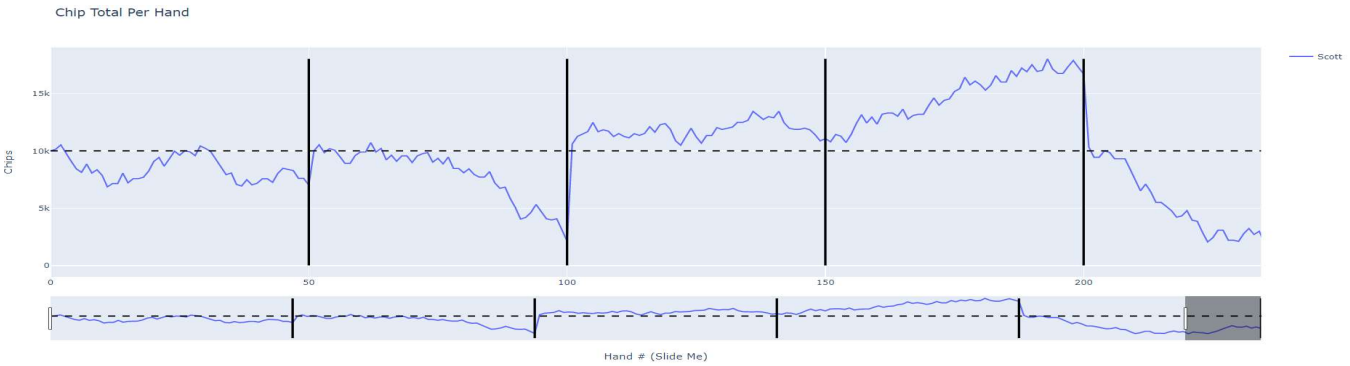
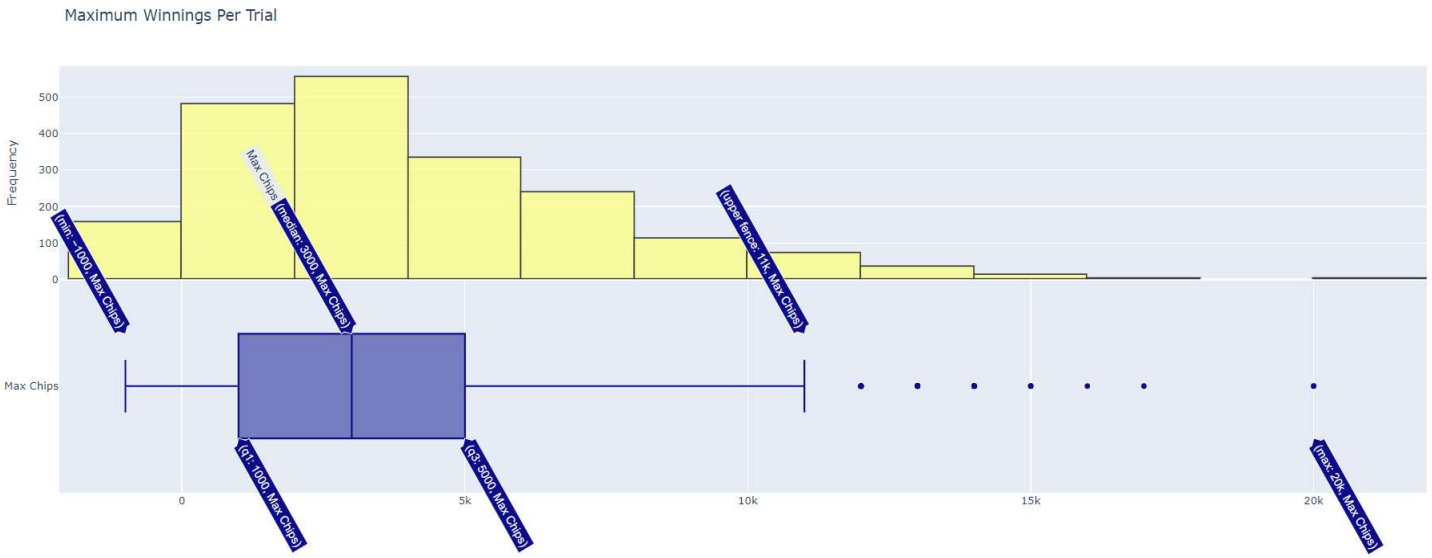


Figure 4 Graphic from the dashboard that shows players' chip total after each hand for each trial in a simulation. The slider at the bottom allows a user to conveniently scroll for long graphs

This graphic visually shows the trend and overall results of each trial in a single instance. It also allows a user to quickly see how often a player went bust, as well as the overall maximum chip count a player was able to achieve using a certain combination of betting and playing strategies. However, to better analyse maximum wining potential, there exists a tab called maximum winnings per trial. This tab is the aggregation of the maximum chip total minus the starting chip total (maximum winnings) for every trial at any instance during that trial. It doesn't matter if the maximum winnings for a trial was seen in the first, middle, or last hand of the trial. It helps illustrates the point of maximum potential earnings at which a player should walk away from the table. Attempting to increase their winnings past a certain point comes at a higher probability of losing more money. Figure 5 shows an example of this tab.



Went Broke: 777 (39.0%) | Finished Negative: 697 (35.0%) | Finished Positive: 526 (26.0%)

Figure 5 Distribution of maximum winnings achieved per trial regardless of when this maximum occurred within the trial. This gives a user an idea of when they should walk away after they have won a certain amount.

In this instance of 10k starting chips and a betting strategy "Always bet table maximum", the median max winnings seen across 2000 simulation trials (100,000 games played) was $3000. The table max bet for this simulation was $1000. Therefore, this is indicative that a player using this strategy in this scenario should almost always walk if at any point they're up $3000. If they wish to risk more, the 75th percentile of max winnings was $6000 and the absolute max winnings seen was $21,000, but this is clearly shown to be an outlier. The chances of them getting any higher than median max winnings (50th percentile) quickly falls off.

It is also important to note however, when implementing this strategy, that the player either went broke or finished in the negative 74% of the time. So it's probably never a good idea to bet the table maximum every single hand.

4.1 How to Bet

In this section, we look at a few betting strategies and the winnings results from several simulations. The user's manual contains full information on all implemented betting strategies. Some of them are self-explanatory such as "Always Bet Max" and "Always Bet Min". For the strategies that mimic a distribution, a random number is sampled from the specified distribution. This number should always be a number between [0, 1]. That number is then multiplied by the table maximum as the applied bet. In the table below, we can see the results from the previous simulation configuration of 2000 trials, 8 decks, no other players, shoe cut percentage 0.75, min bet of $100, max bet of $1000, and 50 hand max per trial. This blackjack simulation pays 2 to 1. This means that a winning bet of $100 will net you $200.

Betting Strategy	25%	50%	75%	Went Broke (%)	Finished Negative (%)	Finished Positive (%)
Always Bet Max	1000	3000	6000	39	35	26
Always Bet Min	100	300	600	0	73	27
Uniform Random	620	1670	3320	5	68	26
Triangular(0, 0.2, 1.0)	500	1360	2675	1	74	25
Triangular(0, 0.5, 1.0)	580	1510	3145	3	72	25
Triangular(0, 0.8, 1.0)	700	1860	3615	9	67	24
Normal(0.2, 0.3)	370	1015	1985	0	72	28
Normal(0.5, 0.3)	590	1530	2980	4	72	24
Normal(0.8, 0.3)	810	2190	4280	21	54	25
Expo(0.2)	230	760	1530	0	73	27
Expo(0.5)	505	1600	2960	3	70	28
Expo(0.8)	670	1955	3675	10	65	25

This table shows 25, 50, and 75 % thresholds for the distribution of maximum winnings accumulated from all trials. Again, it doesn't reflect when the maximum was achieved. For a given trial, the player could have been up $6000 yet ended the trial being broke. We are simply trying to identify the maximum point in which it is no longer beneficial to try and increase earnings. Each betting strategy utilized the same random seed generator so all trial outcomes were the same across all simulated versions.

As one would expect, always betting the table maximum provides the highest potential reward, but at the highest risk of going broke at 39%. Therefore, in this scenario, if you were up anywhere from $1000 to $3000, it would be extremely smart for you to just walk away. The odds are not in your favour for long. Conversely when implementing a strategy of always betting minimum, the risk of losing all your money is virtually non-existent, yet you stand to gain almost nothing. There's no point in even playing if this is the strategy you wish to take. In fact, no strategy is the clear favourite after all simulations.

Without taking into account how many previous wins or losses has occurred or what the current chip total is before betting, the results simply show that conservative or aggressive random betting yields conservative or aggressive results at the probability of low or high risk respectively. However, as intended, it is a good reference in highlighting when a player should walk away if they are up a certain amount. No matter what, in the long run, the house always wins.

5. Future Considerations

In the development of this simulation, only the absolute basics were implemented and simulated. For future versions, giving the player the ability to split, double down, and surrender will be implemented. Also, more dynamic betting and playing strategies will be added. Currently, betting happens randomly from the supplied distribution if not "always bet" max or min. Because of this, past outcomes bear no weight on current decisions. This is not an entirely realistic scenario in the real world. A player who has won 4 or 5 in a row is more likely to slowly start to increase how much they're betting instead of randomly choosing from a distribution. Also, because all players at the table can see each other's cards, more probabilistic decisions could be made as a part of the strategy. If you're sitting in the last position at the table and you can see that virtually no one else has been dealt a 10 or higher, if you're total is 14, there's a high probability that you will be dealt a 10 or face card and go bust. So the smart decision would be to stand. Giving strategies the ability to make these type of probabilistic decisions would also be a great feature to add.