Final Project Write-up: Alpha-Beta Pruning based AI for BANG!

The Game:

BANG is a 4 to 7 player card game where each player is randomly given a role. In a 4 player game, there are 2 Outlaws, 1 Renegade, and one Sheriff. In a 6 player game, the role of Deputy is added, and in a 7 player game, there is an additional Outlaw. With the exception of the sheriff, all player roles remain a secret until they are eliminated, or the game ends. The game ends when either the Sheriff or all the Outlaws and the Renegade are eliminated. The goal of the Sheriff is to eliminate the Outlaws. The goal of the Deputy is to protect the Sheriff and eliminate the Outlaws. The goal of the Outlaws is to eliminate the Sheriff. The goal of the Renegade is to first eliminate the Outlaws, and Deputy if present, then eliminate the Sheriff. The deck consists of 22 different types of cards. “BANG!” cards allow you to shoot another player if they are in range of your gun and “MISSED!” cards can be played to protect against shots. There are also status cards, such as the mustang, which increases your distance, or beer, which restores your health. The real complication comes from being able to play as many cards from your hand as you want during your turn, but at the end of your turn, the number of cards in your hand must be no greater than your remaining health points.

The Problem:

The problem we wished to tackle, was the creation of an AI player that will competently play a game of BANG.

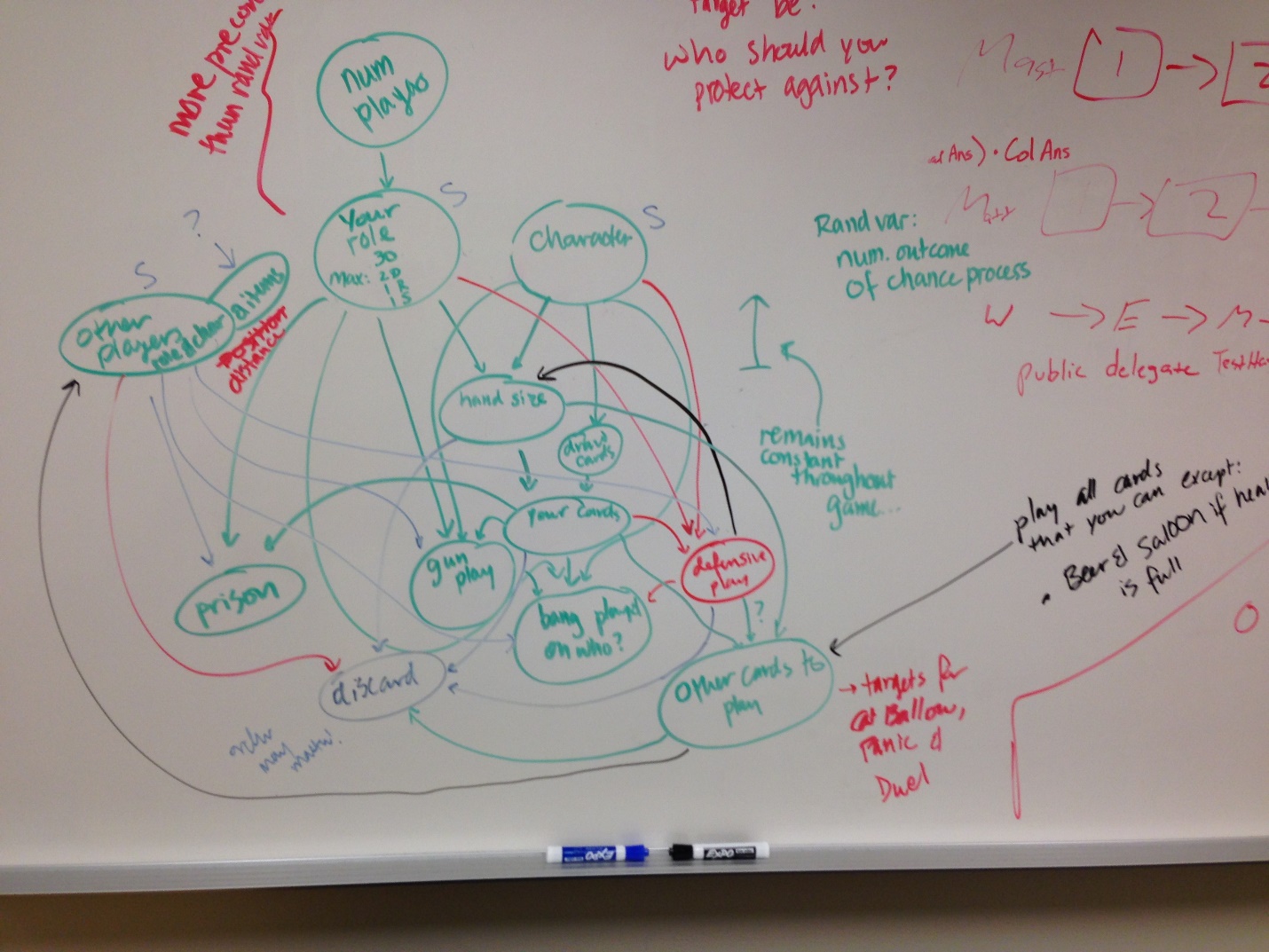
Our Approach:

Our approach to solving this problem, was to implement a Bayesian Network to guesstimate the roles of the other players. A Bayesian Network is a probabilistic graphical model, formed from a directed acyclic graph which represents a set of variables and their conditional dependencies. We chose this method because BANG is a game of incomplete information. The information is incomplete because all roles, excepting the Sheriff, remain hidden until the player is eliminated. With roles hidden, the AI cannot know for certain the goals of the other players, excepting the Sheriff.

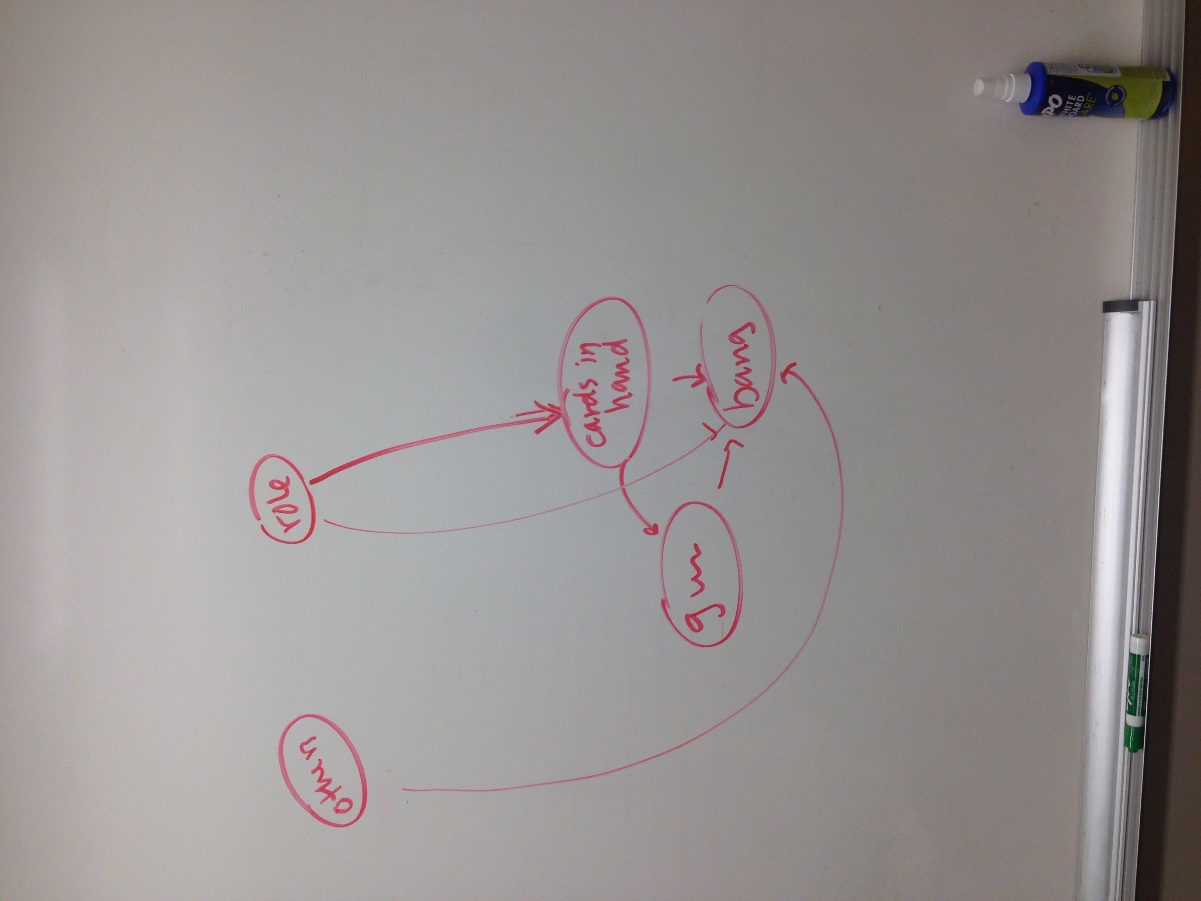
The language we chose to use for this project was python. This was mainly due to the large number of libraries available and the flexibility of the language. For our Bayesian Network, we used the bayespy module.

Our first step was to create the game. This included constructing classes for the cards, deck, players, boards, and game play. There was no open source code for this, so we had to build the game from the ground up. This portion of the project was more time consuming than we had anticipated. The first hurdle we encountered was the amount and complexity of actions the game required. It turned out to be far more than we had initially expected. The second hurdle was the language barrier. We had both worked with python before, but not to this level of complexity. As a result many minor errors and incorrect assumptions were made, which took time to track down and correct. Once these had been dealt with, we were faced with the challenge of constructing a Bayesian Network for the AI.

Our first step was to physically draw the model of the Bayesian Network we wished to create. It took about an hour of discussion before we had incorporated all of the factors that go into decision making during a turn. The model turned out to be very large and complicated.



We then broke the diagram down into levels of implementation to make the task manageable



For our basic level, we only incorporated the effect of the role of our character and the information of who had tried to shoot at who last to determine who the player should shoot. Unfortunately, during implementation we found that we could not find enough information about how to use bayespy that was relevant to our application. We consulted numerous academic papers and websites, but couldn’t find an application of bayespy that was similar to our needs. Due to this and the approaching deadline for the project, we chose to implement an AB-Pruning algorithm instead. We chose this algorithm because it allowed us to define a complex heuristic that could reflect the myriad of influences that make a turn a good choice for a player, while also taking into account the known and unknown roles of other players.

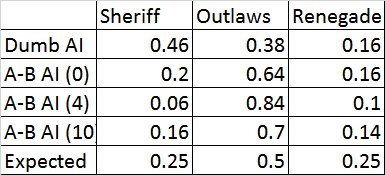
We first created a SimpleAI, which played as many cards as it could out of its hand on itself and the person to its right. When given options about selecting a card from a general store, or discarding at the end of a turn, the SimpleAI picked the first option presented. This gave us a base line from which to judge how well our AB-Prune AI played since the SimpleAI does not take its role or other player’s roles into account.

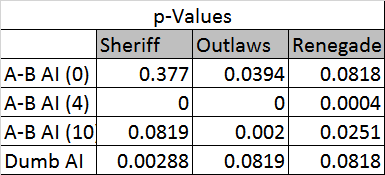
We then began creating our AB-Prune AI. We determined that we would create separate AB-Pruning algorithms for each of the possible number of players in the game. This allowed us to fine tune the scoring algorithm for the logic inherent in a game with the specified number of players and account for the different branching factors. A good example of this is the difference between a four player game and a five player game. In a four player game, the Sheriff can shoot anyone with impunity. The only other roles in the game are those the Sheriff wants to eliminate in order to win the game. With a five person game, the sheriff now has to worry about shooting the Deputy. The Deputy is motivated to eliminate the same people as the Sheriff, because if the Sheriff wins, they do as well. If the Sheriff ends up eliminating the Deputy, they have decrease the likelihood of their winning, since they have removed a player acting as a friend.

Our scoring algorithm takes into account several factors. The first is if our player is alive. This is crucial, because the best chance of winning is if you are alive. It is possible, in the case of the Outlaws, for all of the Outlaws to be dead and still have them win. This is an unlikely outcome because the number of starting players would have to be five or greater, and the Sheriff would have to be eliminated before the Renegade had killed the Deputy(s). If you are still alive, you get +100 point, while your death earns -1014 points. The second factor is dependent on the player’s role, as well as the number of dead players and their roles. This allows for the influence of the player’s role to be felt. For instance, if the player’s role was Renegade and the board showed that other Sheriff had died while an Outlaw was still alive, this would not be considered a good turn. But if all the Outlaws had died, the Renegade would view this as a very good turn, since it would mean he had won. If the players your role wants eliminated are dead, you receive positive points. Otherwise you receive no points. The third thing we took into account was if you had won, which provides an overwhelming amount of positive points. This makes the choice guaranteed to be the preferred choice. The fourth thing we took into account was the number people in range of your gun. For every person in range you receive +50 points. This determines your ability to eliminate the people you need to in order to win. For every good status card on your board (Mustang, Barrel, and Scope) you receive +20 points. For every bad status card (Jail and Dynamite) you receive -50 points. The good status cards increase your chances of not losing health, while bad status cards are likely to cost you health or a turn. If your health is at maximum, you receive +100 points, otherwise, you receive +15 points for every health point you have. For every health point your opponents have, you lose 10 points. This gives an indication of how close you are to winning when no deaths have occurred. Finally, the Sheriff loses 100 points for every person that can shoot him, while other players lose 25 points per person that can shoot them. This is to account for the fact that the Sheriff has many players trying to eliminate him, while others likely only have one.

To test the successfulness of our AB-Prune AI, initially ran the tests on a game of four players. We ran 50 games where all the AIs were AB-Prune with 0 ply, 50 games where the AIs were AB-Prune with 4 ply, and 50 games where the AB-Prune with 10 ply, and 50 games where all the AIs were SimpleAI. For each of these sets of 50 tests, we recorded the number of wins for each type of role. For the set of half and half, we also recorded the roles each AI played. We repeated this process for five, six, and seven players.

Results:

Our findings were quite fascinating. We saw a marked difference between the abilities of the dumb AI and the Alpha-Beta method. The SimpleAI favored the Sheriff more than any of the other types, while all of the Alpha-Beta methods favored the Outlaws to differing degrees.

Since we had no actual predictive model of our system, we decided to base our predictive model on pure chance: in a game of 4 people, we would expect 25% of the games to go to the Sheriff, 50% to the Outlaws, and the final 25% to the Renegade. Based on our personal experience, we did not expect this to actually be terribly reasonable, since our personal experience has suggested that Sheriffs are less likely to be successful and the Renegade even less likely to be successful, given the difficult nature of her position.

We did a simple analysis of our results of four samples, given our short time. We performed a simple one-sample z-test for each proportion whereby. In this test, we tested the null hypothesis where our p’s were equal to the expected values. Our alternate hypothesis was two-sided. In all of our cases, we saw a significant difference from the null at a 5% level of significance for at least one of the values. It is unclear whether these differences are because our predictive model is bad or because we have not tried an appropriate ply.

Of greatest interest is the changes from ply to ply. The favoring of the Outlaws varied fairly substantially based on the ply for the Alpha-Beta testing. Unfortunately, we didn’t have the time to do substantial experiments on this. Ideally, we would have liked to do 50-100 samples at each ply from 0 to 50 and see what that graph looks like. Perhaps the values converge on a set of proportions? It would be fascinating to find out.

In the end, our results make us think that perhaps the best model for gameplay could vary based on the role of the user. The SimpleAI may make a good model for the Sheriff, while the Alpha-Beta makes a better model for the Outlaws. It would be interesting to see if a model could be created to model a Renegade.

Future Work:

Future work we would like to see done, is the incorporation of the character abilities into the game and have them also factor in to the AB-Prune AI. Some of the effects the characters would not pertain to the operation of the AI algorithm, such as drawing extra cards at the start of a turn or having an automatic barrel. Those cards that would have an effect would be abilities such as, gun is always volcanic, two misses to stop a shot, and draw from the hand of the player who shot you. These would change the dynamic of who you want to shoot, given an option, as well as drastically increasing the complexity of the game.

Bibliography

Nicholson, Ann E., Kevin B. Korb, and Darren Boulton. "USING BAYESIAN DECISION NETWORKS TO PLAY TEXAS HOLD’EM POKER." *ICGA Journal* (2006): 1-13. Web. <http://www.csse.monash.edu.au/bai/poker/2006paper.pdf>.

"Constructing the Model." *Constructing the Model — BayesPy V0.4.1 Documentation*. Bayespy.org. Web. 08 Mar. 2016.

"Bayes Networks in Games? - Artificial Intelligence." *GameDev.net*. Web. 08 Mar. 2016.

Krieg, Mark L. "A Tutorial on Bayesian Belief Networks." *Surveillance Systems Division, Electronics and Surveillance Research Laboratory* (2001): 1-64. *Department of Defense, Australia*. Web. 8 Mar. 2016. <http://dspace.dsto.defence.gov.au/dspace/bitstream/1947/3537/1/DSTO-TN-0403.pdf>.