Riley Siebel
2/4/11
Senior Thesis
Professor Schapire

Playing Clue: February Paper

Clue is a classic mystery game originally published by Waddington's as "Cluedo" in the UK in 1949. Clue can be played by 3-6 players, and involves these players moving about a board in an attempt to solve a mystery.

There is one card representing every player, one card representing every room on the board, and one card representing each of six weapons, for a total of twenty-one cards. One of each of these cards is hidden away, and constitute the solution to the game. The rest of the cards are dealt to the players who attempt to determine which cards were originally hidden away. This is done by means of "suggestions:" when a player enters a room he/she may make a suggestion, and propose a room, suspect, and weapon as the solution. Each player in turn must attempt to "refute" this proposition by showing the suggester that one of the proposed solution cards is possessed by the refuter. This continues until a player attempts to solve the puzzle. If they are right they win, otherwise play continues without the player.

Clue is an instance of the "treasure hunt problem," in which the player gains information at targets along the path to the goal. AI researchers have long been interested in games like these because they offer a stripped-down, less complicated reality in which to operate. Several games have been solved, to some extent.[1]

Playing Clue is an attempt to apply the tools of artificial intelligence to Clue. In particular it incorporates aspects of propositional logic, Markov decision processes, and information theory. The goal is to achieve an algorithm that plays the game Clue as close to perfectly as possible.

The game of Clue is not entirely unexplored. Indeed two Duke researchers, Chenghui Cai and Silvia Ferrari, have presented an algorithm that attempts to achieve the same goal using slightly

[1] Tic-tac-toe, for example, has been totally solved; the perfect game has been completely described. On the other hand, the game of chess is not "solved," but computers are now as good if not better than almost every human ever.

different methods[2]. Their approach is based on Markov decision processes; it utilizes these techniques to find a policy that optimizes the expected sum of a discounted reward along a path to determine where to move during a game of Clue. The difficulty (and the reason their paper needs such a long title) is that this reward is difficult to calculate, and so they use Q-Learning to approximate the value. Approximating the reward is a fairly good solution but it loses some information. This project takes a different approach; it attempts to directly calculate the rewards using information theory when possible, and only approximates the value when the reward is impractical to calculate directly.

One of the key components of this project is a logical agent. This agent needs to be able to perceive the relevant aspects of the game, maintain a representation of the game, and reason about the game as much as possible to provide the AI with as much information as possible. As a resource in creating this logical agent I was able to find a project published by *Machine Learning Experiences in Artificial Intelligence* that aims to teach propositional logic through the framework of Clue, and thus provides a framework to build a propositional logic reasoner for the game.[3] The project describes the general idea of what needs to be done, but leaves all of the implementations up to the student.

This was a good starting point, and indeed I have been able to structure much of my project as additions to this base architecture. The logical agent is more or less completed; the AI incorporates a "clue reasoner" that is able to reason about the locations of cards based on the information available up to that point, and knows the answer to the puzzle as soon as it is possible to know it.

The AI uses this reasoner to determine what room to go to, and which suggestion to make. It is also used to infer the solution to the puzzle. This reasoner is the core component of the AI player. In order to keep its knowledge base updated, the AI notifies its logical agent whenever the AI player perceives or performs any action. These include suggestions made by the AI, suggestions made by other players and all the other actions with cards that can be performed in Clue.

The logical agent is programmed to perceive the obvious things, but it might be possible to increase the scope of knowledge that the reasoner deals with. The reasoner could know, for example,

[2] C. Cai, and S. Ferrari, "A Q-Learning Approach to Developing an Automated Neural Computer Player for the Board Game of Clue." *fred.mems.duke.edu/LISCpapers/IJCNN08_**Qlearning**.pdf*

[3] Todd W. Neller, Zdravko Markov, and Ingrid Russell, "Clue deduction: an Introduction to Satisfiability Reasoning." Aug 10, 2005. http://uhaweb.hartford.edu/compsci/ccli/clue.htm

about the location of the other players, and the path they are taking through the game. If the logical agent is provided with a sufficient logical grammar to describe these attributes it might even be able to make inferences about their intentions that could be of use. For example the AI has the opportunity to call a player to the other side of the board by suggesting him as a solution.[4] Another possibility is to use the information already in use to reason about what the *other* players know about the AI's cards, or the other players, instead of just what cards they have. This would be useful in deciding what suggestions to make, as well as how to refute suggestions when the player has a choice. This could be taken to even further, reasoning about what the other players know you know about them etc.

One major use of this reasoner is in calculating the reward function. The AI player plans its moves by assigning rewards to the rooms, then propagating those rewards scaled by a discount factor across the board, in an adaptation of "value iteration".[5] The reward of a room is the maximum expected information gain of the possible suggestions that can be made in that room. In other words, it is the maximum amount the AI expects to learn about the solution in a given room, assuming they ask the best question. The "information gain" is defined as the mutual information of the suggestion and the solution. To calculate this value the AI needs to know the probabilities of every possible permutation of the cards, *given what the logical agent knows and can infer*.

As there are $6! \times 6! \times 9! \approx 2 \times 10^{11}$ permutations, calculating these probabilities is intractable with a brute force approach. This is why Cai & Ferrari used Q-learning to learn them. This is the last major stumbling block left in my project, but I believe that it is possible to make this a tractable problem. The mimimum number of cards that can be dealt to a player is 3, when all the players are playing. As soon as a player sees three cards, the number of possible permutations drops by a factor of 100 to 1000, depending on which kinds of cards the player sees. The graph of f(x!, y!, z!) is very hard to represent, but it very quickly approaches zero as x, y and z get smaller.

However it is true that in the early stages of the game it will take an impossibly long time for the AI to calculate these probabilities by brute force. In these situations I believe the best solution is to use a sampling algorithm to get a good approximation of the probabilities.

[4] One detail of the game that I glossed over is that when a suggestion is made, the player who is named as the suspect is transported to the room where the suggestion is made. This can be used to take players away from where they are trying to go on the board.

[5] Stuart Russell and Peter Norvig, "Artificial Intelligence: A Modern Approach," 652.

This combines the advantage of using approximations when appropriate with the precision of calculating the actual values when appropriate. This threshold is of course a parameter and I hope to do some experimentation to determine the best value.

Another possibility is that there is a solution in combinatorics. I am exploring the possibility that the satisfying deals can be counted in a time sensitive way using similar techniques to counting the number of full houses that can be dealt. This is something I am more recently exploring and is less certain. I hope to ask Bernard Chazelle for a little advice on approaching the problem from this perspective.

In order for this AI to have a game of Clue to play, I have created an interface and a simulation of a game of Clue that allows 3-6 human or AI players to participate on one computer. The simulation includes representations of the cards in the hands of the players and supports suggestions and everything else one would expect. The program is structured in such a way that it is fairly easy to add support for other AI's. I hope to take advantage of this by adapting Cai and Ferrari's program to play against mine, and see which comes out on top.

With 7 weeks left until the thesis outline is due when I plan to abandon the code and dedicate myself to Microsoft Word, I feel like I am in a good position to complete this project. Leaving myself another 2 full weeks to complete the design of the reward algorithm, I have 6 weeks remaining to implement the rest of the AI in code, work out the final bugs, and hopefully make my simulation a little bit more aesthetically pleasing. In this time I should also be able to start on the body of the report, including documentation of the code and descriptions of the algorithms. I also hope to come up with a reasonable representation of the f(x!, y!, z!) graph, maybe using color as the fourth axis. Then I'm right on schedule to finish out with 3 weeks for completing the report, then another week and a half to get finished for the presentation, and 2 weeks to prepare for the poster session.

Date	Task
Feb 18	Finish reward algorithm
Mar 18	Finish Code
Mar 25	*Thesis Outline
April 15	*Final Report
April 20	Finish Presentation Materials
April 25	*Presentation
May 5	Finish Poster
May 10	*Poster Session