Rummy 500 With Symbolic AI Opponent

(Draft 4)

Matt Grzenda

Abstract

This paper serves to document the construction and performance of a Rummy 500 game program with a symbolic AI opponent.

Contents

1. **Introduction**
2. **Background**
3. **Approach # talk about different approaches to encoding heuristics (other programs) situation action rules, etc. generic heuristic architectures (if -> then), self organizing rule bases**
4. **Knowledge Representations # Do for next week, how to encode a rule, how to encode the rule base, how you examine the rules, two levels of abstraction (pseudocode and prolog).**
5. **Game Playing Framework**
6. **English explanations of Decisions**
7. **Statistics and Assessment**
8. **Possible Extensions and Elaborations**
9. **Conclusion**
10. **References**

1 Introduction

There are multiple decisions to be made in Rummy 500. From where to draw a card, to which melds to play on the board, which card to discard at the end of a turn, and more. Since humans normally play this game, there is a set of heuristics that define a strategy to play the game. It is also possible to play this game randomly, where decisions about which melds to play. The program progressed from a random Rummy player, to a rule based symbolic AI player in the following fashion:

1. Representing the deck and discard pile
2. Representing the players
3. …

2 Background

The Construction of a Card Game

All card games were created by humans and all card games are played by humans. In these card games, humans develop heuristics to play the game in a way which maximizes their payment. The payment is defined as something of value in a game that an individual would want to possess. In card games, payments may include points, cards, the ability to lose cards, and, in the case of Poker, money (von Neuman and Morganstern, 58).

Considering how humans play games, why not construct a card game centered around the use of pre-programmed heuristics to play the game? Unless the implementation of the game happens to be able to learn new heuristics, it will only perform as well as the programmer who created it. However, depending on the heuristics that were encoded, the machine could still be a formidable opponent. The reasoning behind using a heuristic architecture in this implementation of Rummy 500 is for a simple, natural design of a machine opponent, while still being able to create a worthy opponent.

Classic Card Playing Programs

Game playing programs are created for a variety of reasons. Whether it is to see if it is possible for a machine to beat a human (Mitchell, 156), or because elements of the game can be abstracted to other parts of society (Billings, Papp, Schaeffer, Szafron, 1), people have been fascinated with creating AI games for decades. This of course extends to card games, to the point where many card playing applications are now available to buy on any App Store.

GIB was a bridge playing program created in the late 1990s. Unlike the other bridge playing programs before it, such as Bridge Baron or Paradise, GIB does not use any human methodology to play bridge (Ginsberg, 584). Instead GIB uses a brute force search to determine the best move in a given situation. For selecting cards for example GIB uses the Monte Carlo Card Selection Algorithm. This algorithm constructs a deal of the bridge game thus far, based on what is known and which cards are in play. Any unknown cards are randomly dealt out if necessary. Every possible move is calculated and then compared to find the best move on this particular deal. This move is then returned (Ginsberg, 585). After the performance of the machine was published in a bridge magazine, GIB was invited to compete in the world bridge championships in France, where it finished in 12th place out of a possible 34th against human opponents (Ginsberg 586).

The significance of GIB lies in its achievements in performance and its radically different architecture. Each bridge playing program before GIB used a heuristic architecture to play bridge while GIB used a mini-max style tree generation algorithm to calculate how to respond to a certain scenario. Unless there is a significant jump in research for heuristic based bridge programs, GIB has set a new standard of card game architectures.

Another influential card game program was a Texas Hold'em playing program called Lokibot. It was created as a way to research how AI should respond to imperfect knowledge,

multiple competing agents, risk management, deception, and unreliable information (Billings, Papp, Schaeffer, Szafron, 1). Other programs before it had created simpler versions of poker to simply have a game that was playable, but the developers of Lokibot wanted the real-world parts of the game to be included (Billings, Papp, Schaeffer, Szafron, 2). In order to account for this, Lokibot evaluates its hand every time a card is turned over during gameplay. To evaluate its hand, an enumeration technique is used which calculates how good its hand is when compared to every other possible hand. A percentile is calculated and based on the hand strength and the program decides whether to check, bet or fold. The enumeration is also weighted to account for different opponents' playing strategies and also because not all hands are equally likely (Billings, Papp, Schaeffer, Szafron, 7). To test out the overall performance of Lokibot, it played itself. The self play was tournament style and each "opponent" was a different variation of itself with varying skill level. Overall, the variation with the most strategies encoded to decide on moves won the most (Billings, Papp, Schaeffer, Szafron, 12)

Lokibot's significance is its approach to hard problems to solve in AI. By taking real world behaviours that are hard to define in a general context and applying them to a well defined space such as Texas Hold'em, these problems can be examined more closely. Under this new light, dealing with imperfect knowledge, unreliable information, multiple competing agents and more is just a probability computation. Each one of these real world behaviours also exists outside of Poker in the world of business and even warfare. What was learned by creating this Texas Hold'em game could be the basis for stock trading, weather and political forecasting, and business transactions applications of tomorrow (Billings, Papp, Schaeffer, Szafron, 2).

The Heuristic Architecture

According to Herbert A. Simon and Allen Newell, "A process that may solve a given problem, but offers no guarantees of doing so, is called a heuristic for that problem" (1957). Based on this definition, humans tend to be heuristic problem solvers. When a human encounters a complex problem, rarely is their first thought the most optimal solution to the problem. More often than not, their first solution is lacking in some way. However, this makes heuristics far from being useless. Humans tend to think of simple solutions, compared to the exhaustive alternatives. For example, search algorithms for data structures tend to be heuristic based since the exhaustive options are more computationally expensive (Kokash, 3).

When a program is said to be based on rules or conditions, this program can be said to be based on heuristics since the programmer created a process which may solve a given problem. The Cyc project, for example, was, "an attempt to model the human consensus knowledge." (Yuret, 1). Cyc used a very large knowledge base to try and accomplish this goal, with a large team hired to enter knowledge into the database. Cyc proposed to solve the problems of brittleness in programs by offering the common sense that humans had. Although Cyc did not accomplish its main goal to be able to derive a deeper meaning from a provided symbol, it opened up a treasure trove of research possibilities (Yuret, 25). Flaws in the program created a demand for new approaches to issues such as how to model symbols in other ways than deductive inference (Yuret, 16).

Another noteworthy program which is based on a heuristic architecture is called Bagger. This program is designed to find the optimal way to put all your groceries in bags (Roman, Gamble, Ball, 26). Bagger does this by grouping all of the items by weight and then packing the items into bags one at a time starting with the heaviest items and working down to the lightest weight items (Roman, Gamble, Ball, 26). The significance of using heuristics here is the amount of computation time saved compared to other methods. An exhaustive method would try every possible combination of grocery items in bags, and keep track of the results. Once all the possible combinations of groceries are found, the best option is returned. The exhaustive method is much more expensive to compute than the heuristic approach due to the excess number of steps needed to compute it.

Heuristic architectures are not the perfect solution to any problem, but they are also not the worst solution. The goal of a heuristic architecture is to write a program which solves a problem the way humans do. Heuristic approaches to problems often lead to the most optimal solution as heuristics are improved and swapped for better heuristics. If any process that may solve a given problem is a heuristic, then the all processes thought of by humans are heuristics.

Rummy 500

Rummy 500 is a classic card game played by 2 – 8 people. Each person is dealt 13 cards to start. On each person's turn, they start by picking a card from either the deck or the discard pile. When drawing from the discard pile, you are allowed to draw multiple cards. In order to legally draw from the discard pile, a person must be able to use at least one of the cards he/she draws. At this point, the person will look for a meld, or combination of cards, to play on the board to earn points. A legal meld in Rummy consists of at least three cards. The first type of move is having three cards of the same face value (i.e. three kings), or, to use poker terms, a straight flush (i.e. 2, 3, 4 of hearts). Once a player has either produced a meld, played on another opponent's meld, or realizes he/she can not make a meld, they put a card in the discard pile and their turn is over.

Rummy is split into multiple rounds. A round starts once cards are dealt and ends when a player runs out of cards. Next, points are calculated based on the melds each person played and based on the board. Cards 2 – 10 are worth 5 points; jacks, queens, and kings are worth 10 points; and Aces are worth either 5 points if played low, or 15 points if played high. A player's score is the points from their melds, minus the sum of the cards left in their hand. Once a player reaches 500 points, the game is over and that player has one the game.

3. Approach

4. Knowledge Representations

5. Game Playing Framework

6. English Explanations of Decisions

7. Statistics and Assessment

8. Possible Extensions and Elaborations

9. Conclusion

10. References

1. von Neumann and O. Morgenstern, *Theory of Games and Economic Behavior*, Princeton University Press, 1944.

2. Mitchell, Melanie. *Artificial Intelligence: a Guide for Thinking Humans*. Pelican, 2020.

3. Billings, Darse & Papp, Denis & Schaeffer, Jonathan & Szafron, Duane. (1998). Poker as a testbed for machine intelligence research. Artificial Intelligence - AI.

4. Matthew L. Ginsberg. 1999. GIB: Steps Toward an Expert-Level Bridge-Playing Program. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (*IJCAI '99*). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 584–593.

5. Kokash, Natallia. 2005. An Introduction to Heuristic Algorithms. University of Trento, Italy.

6. Yuret, Deniz. (1996). The binding roots of symbolic AI: a brief review of the Cyc project.

7. G. -. Roma, R. F. Gamble and W. E. Ball, "Formal derivation of rule-based programs," in *IEEE Transactions on Software Engineering*, vol. 19, no. 3, pp. 277-296, March 1993, doi: 10.1109/32.221138.

8. Contributors to Wikimedia projects. “500 Rum - Wikipedia.” Wikipedia, the Free EncyclopediaWikimedia Foundation, Inc., 15 Sept. 2003, https://en.wikipedia.org/wiki/500_rum.