Heuristic game playing machines for GOPS

By

Shawn Hunneyman

Abstract

The report focuses on the program that I created in Lisp to use heuristic strategies to play Goofspiel. The program is broken down into seven parts and features some demos to the progress of the program as I created it.

Introduction

Goofspiel or game of pure strategy is a card game that features two players. This game is interesting in that the game allows players to think of many different heuristic strategies to play against an opponent. This property of the game also allows for a different approach to heuristic strategies, namely programs that can play these strategies that only a machine can use. The main strategy that the program I implemented was to use probabilistic decision-making to play against other machines.

First context is needed to further explain the progress of the program that was created. A description of the game Goofspiel and some historical notes regarding the game and heuristics are needed. A program description to give the reader an operational understanding of what the program is supposed to accomplish and why. Demonstrates to show the program is working as intended and then finally a conclusion.

Background

Goofspiel-

Goofspiel was invented by Merrill Flood which can be seen from his own words in an interview with Albert Tucker. Flood said, “Do you remember I invented the game "Goofspiel", which was popular for a while?” (“Merrill Flood”)

Sheldon Ross introduces two things namely an alternative name for Goofspiel and the matching strategy. The alternative name given by Ross is from, “The game of pure strategy, sometimes called Goofspiel or Gops” (1971). Another thing that Sheldon Ross shows is that the matching strategy is best against a random player. The statistics gathered later in the demo section confirm the work done by Ross. Ross makes the claim in, “In Section 2, we consider this game under the assumption that Player II discards his cards in a completely random manner. Given this information, we show that the best thing for Player I to do is to always match the upturned spade” (1971).

Heuristics-

What are heuristics? Mohamad Hjeij and Arnis Vilks give a good definition of heuristics in the manner that I use them throughout the report. Mohamad Hjeij and Arnis Vilks say, “However, one feature does distinguish heuristics from certain other, typically more elaborate procedures: heuristics are problem-solving methods that do not guarantee an optimal solution.”(2023) This description is fitting because the heuristic strategies I had in mind were not to be the optimal solution to playing a game of Goofspiel.

Probabilistic approach-

The work Rhoads and Bartholdi did was my inspiration for using a table of probabilities to pick a card. They claim, “It is not hard to see that one should not choose a deterministic strategy. In fact, every deterministic strategy **A** can be defeated as follows. Use strategy **A** to find the card that my opponent is going to play. If my opponent is going to play a king, play the ace. Otherwise play the card that is one higher than my opponent’s choice. This counterstrategy will win every round except one resulting in a trouncing. Instead the strategy should have some random variations where one plays particular cards with some probability.” (2012) was pretty strong to convince me that working with a table should be a good choice for creating heuristic strategies.

Program Description

The program is constructed of seven main parts with each part solving a smaller problem that is needed for the program to function.

1. Creating and displaying a standard deck of playing cards. Creating and displaying a standard deck of playing cards is the most important part of playing the game while the display portion is to inform the user of what is happening and to see if the cards were created correctly.

2. Dealing a hand, shuffle the prize suite, remove the discard suite. Dealing a hand to the two players will allow the players to have a medium in which to interact with. Shuffling the prize suite and removing the discard suite are setup methods to allow the game to function correctly.

3. Game rules and a random machine. Now that the shuffling has been implemented and the discard suite is removed the game rules can be implemented to allow for legal moves, scoring of rounds, and an end of game sequence to show who won and lost. The random machine is created to pick a random card in their hand. An informal setup was created here to allow two machines to play against each other to test the various methods.

4. Heuristic machine: matching strategy. The heuristic machine that was created plays the strategy of matching the prize card with the value of the card in the machine’s hand. Ex. If the prize card was the four of diamonds, then the machine would play the four of hearts.

5. Game interface. The game interface allows two machines to play against one another or a human player to play against machine. This was the formal creation of interface to allow machines to play against each other or a human player to play against a machine.

6. Heuristic machine V Heuristic machine. This part focused on the matching strategy heuristic against a new heuristic. This new heuristic machine used the infrastructure to implement probabilistic choices. The initial choices were a random player with a slight adjustment to the higher value prize cards choices to select the higher value cards.

7. Statistics. The collection of statistics of the game’s outcome including the score, wins, loses, draws, hands that won, hands that lost, hands that draw and all the prize cards won.

Demos

Fig. 1

```
>>> display hand1...
--- *hand1* = NIL
>>> display hand2...
--- *hand2* = NIL
>>> display prize suite...
--- *prize-suite* = NIL
>>> display discard suite...
--- *discard-suite* = NIL
>>> dealing cards to hands...
>>> display hand1...
--- *hand1* =
((ACE . HEART) (2 . HEART) (3 . HEART) (4 . HEART) (5 . HEART) (6 . HEART) (7 . HEART)
 (8 . HEART) (9 . HEART) (10 . HEART) (JACK . HEART)
 (QUEEN . HEART) (KING . HEART))
```

Figure 1 is a snippet of output from the lisp implementation of the program. Showing empty hands and suites afterward the deck is created and *hand1* is dealt the heart suit from the deck. The figure captures parts 1 and 2 of the program description. This allowed me to start the layout of the game and start putting some structure in place to play the game.

Fig. 2

```
>>>--------- Round:  13 ---------<<<
--- Prize Card = (8 . CLUB)

--- PLayer 1 plays ---

--- Card  = (KING . SPADE)

--- PLayer 2 plays ---

--- Card  = (7 . DIAMOND)
--- Player 1 won ---
```

Figure 2 displays the output after the game rules were implemented and at this point an informal interface was created to allow random machines to play against one another. The prize card is shown first and then the two machines pick a random card from their hand.

Fig. 3

```
( set-card-list '1 '( 0 20 30 40 50 60 70 75 80 85 90 95 100 ) )
( set-card-list '2 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '3 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '4 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '5 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '6 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '7 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '8 '( 5 10 15 20 25 30 35 40 50 60 70 80 100 ) )
( set-card-list '9 '( 2 4 6 8 20 45 55 65 80 85 115 120 125 ) )
( set-card-list '10 '( 2 4 6 8 10 12 20 30 50 65 95 115 125 ) )
( set-card-list '11 '( 2 4 6 8 10 12 20 30 50 65 80 100 120 ) )
( set-card-list '12 '( 2 4 6 8 10 12 20 30 50 65 80 95 120 ) )
( set-card-list '13 '( 2 4 6 8 10 12 20 30 50 65 80 95 110 ) )
```

Figure 3 shows the result of implementing a table to use in the second version of the heuristic strategy. The values set make the heuristic still a random player with slight adjustments to the probability of the ace and nine through king. The horizontal rows are the card that appears, and the next thirteen numbers are the weights for each card to be played. A random number is chosen between 1 and 130. If the prize card was a six and the random number was 23 then the range is between 20 and 25 so the card selected would be on the left side corresponding to the 20. This is the fourth number from the set of thirteen so the card played would be the four.

Fig. 4

```
M1 wins: 968
M1 losses: 27
M1 draws: 5
```

Figure 4 is a portion of the stats collected in this case showing machine M1's wins, loses, and draws against M2. The majority random machine whereas M1 used the matching strategy to win the majority of the thousand games played.

Reflections and Conclusions

The program created using lisp gave me the opportunity to implement some heuristics to play the game of pure strategy and see the outcomes. I would've liked to have created more heuristics and see them perform against other heuristics, but time did not permit this. My initial plan was to create sets of heuristics that would have been broken down into three categories. The first category was to be strategies that focused only on the machines own hand. The second category was to implement a memory-like system to allow strategies based on the other players played cards. The third category would have been other strategies that I could think of. The plan changed as I did more research and found that using a table for probabilistic heuristics would allow me to generate a large set of strategies quickly but the work to create the methods to use the table took more time to implement than I was expecting. So, I was only able to create one strategy that was a slightly modified random player. This was to test if the table worked and to catch errors before I could tune it to a particular strategy that I wanted. I did accomplish my goal of creating heuristic machines to play the game and collect some statistics on those outcomes.

Bibliography

Sheldon M. Ross. “Goofspiel: The Game of Pure Strategy.” *Journal of Applied Probability*, Vol 8, No. 3, 1971, p. 621 – 625

Rhoads, Glenn and Bartholdi, Laurent. “Computer Solution to the Game of Pure Strategy.” *Games*, vol. 3, No. 4, 2012, p. 150 – 156

Hjeij, Mohamad and Vilks, Arnis. “A brief history of heuristics: how did research on heuristics evolve?” *Humanities & social sciences communications*, 2023, Vol.10, No. 1, 2023, p.64

Albert Tucker, “Merrill Flood.” *wayback machine,* web.archive.org/web/20150310083408/http://www.princeton.edu/~mudd/finding_aids/mathoral/pmc11.htm. Accessed 12 May 2023.