Quantifying the Space of Hearts Variants*

Mark Goadrich[1[0000-0002-9137-7836]] and Collin Shaddox[1[0000-0002-4833-2448]]

Hendrix College, Conway AR 72032, USA
goadrich,shaddoxac@hendrix.edu

Abstract. Hearts is a card game with a rich history and many interesting variants. Why has it remained popular while undergoing significant changes? We use computational simulations of Hearts to understand the experience of players through the application of four heuristics which quantify the drama and security felt by the winning player, the ability of players to win in chaotic imperfect-information situations, and the player's ultimate interest in their decisions. We find that there is a direct relationship between the historical evolution of Hearts through ludemic change and subsequent heuristic improvements to game play.

Keywords: Card Games · General Game Playing · Heuristics.

1 Introduction

A ludeme [15] "is a fundamental unit of play," also known as a building block or mechanic by which game rules are constructed [8]. Over time, in a similar manner to living organisms, games can evolve through changes to their underlying ludemic structure [5], through rearrangement, additions, deletions, and mutations. However, the selective pressure for game evolution is found through optimizing a game to be interesting and fun for human players.

Building off recent progress in developing AI for card games [13], in this paper we seek to leverage computational techniques to understand evolutionary selection pressures on games. In particular, we compare and contrast the player experience through calculating heuristics [6] across various versions of the card game Hearts. Previously used as a testbed for AI research by Sturtevant et al. [18], Hearts is a simple popular trick avoidance game with a long history, where the player with the most points at the end of the game loses. For consistency, we employ a general game playing approach [9], applying the same AI algorithm across multiple games, to computationally understand the heuristic ramifications of each variant.

Specifically, we pose the following questions related Hearts:

1. How do changes in the rules manifest in player experience?
2. What is the relationship between ludemic space and heuristic space?

* Supported by Hendrix Odyssey Summer Research Grants

We begin with a brief explanation of Hearts and its suitability for analysis, followed by a discussion of ten diverse rule variants. We next develop four heuristic metrics to help understand the player experience of card games, and then apply these metrics to each of the variants through computational simulation in CardStock [3]. Finally, we analyze the computational player's experience for each variant, cluster the variants to better understand their differences heuristic space, and conclude with avenues for future work.

2 Hearts

Typically, a game of HEARTS is played over multiple rounds until one player accumulates 100 points. To standardize our analysis and increase the speed of our simulations, each variant of HEARTS analyzed will consist of only one round with exactly four players, with no passing of cards between players. The current canonical rules of Hearts [14] can be summarized as follows:

> A one-round game of HEARTS for four players consists of thirteen tricks. First, shuffle a standard deck of cards. Each player receives thirteen cards. For each trick, players play one card to the trick. The first player will set the lead suit for the trick, which subsequent players must follow suit if they can, otherwise they may play any card from their hand. Also, the first player is restricted to not play a card from the Hearts suit unless one has already been played. Once all cards have been played, the player who played the highest card that matches the suit of the led card will collect all the cards in the trick and become the first player for the next trick. Once all tricks have been played, players earn one point for each ♡ collected in tricks, plus 13 points if they collected the Q♠. If a player happens to collect all ♡ and the Q♠, then they will *Shoot the Moon* and instead subtract 26 points from their score. The player with the lowest point value wins the game.

Multiple ludemes make HEARTS distinctive from other trick-taking games. First, the goal is to avoid taking tricks that contain certain cards instead of accumulate them. Players must avoid the whole suit of ♡, but the Q♠ is the most critical to avoid because of its high point value. In addition, the normal restriction where players must follow the led suit in a trick is compounded with a new limit that players must *not play* ♡ until there is no other option. Finally, players have the ability to recover from initial poor play through by collecting every point and reverse their situation to a winning position.

3 Variants

David Parlett describes the history of HEARTS and a multitude of variants to the basic rules [14]. For our analysis, we examine ten specific modifications to the standard rules given in Section 2. Table 1 summarizes the specific rule changes for each variant we examined. Figure 1 organizes these variants in terms of what

Table 1: Variants of Hearts and their attributes

Variant	First	Last	♡ Broken	Moon	Deck	Points
SlobberHannes	✓	✓			32	Q♣:1
Polgnac	✓	✓			32	J♡:1, J♢:1, J♣:1, J♠:2
Pure Hearts					52	♡:1
Black Lady					52	♡:1, Q♠:13
Black Maria					52	♡:1, Q♠:13, A♠:10, K♠:7
Broken Hearts			✓		52	♡:1, Q♠:13
Hearts			✓	✓	52	♡:1, Q♠:13
Grey Lady			✓	✓	52	♡:1, Q♠:7
Omnibus Hearts			✓	✓	52	♡:1, Q♠:13 J♢:-10
Spot Hearts			✓	✓	52	♡:X, Q♠:13
Widow Hearts		✓	✓	✓	51	♡:1, Q♠:13

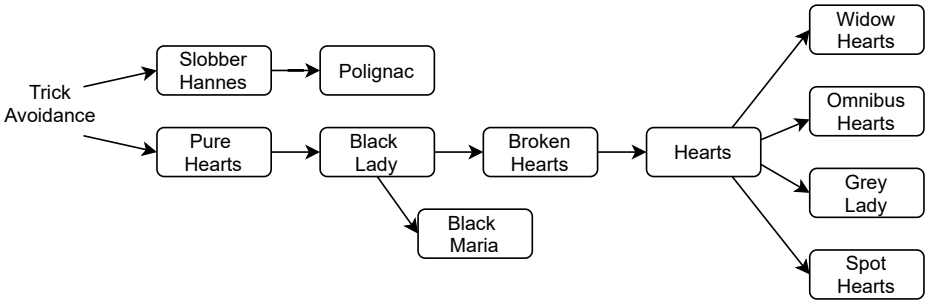


Fig. 1: Evolutionary History of Hearts Variants and Relatives.

is currently known about their historical progression through changes via edits, insertions, and deletions to the canonical Hearts rules in ludemic space [5].

To examine historical ancestors of Hearts, we start with bare-bones Pure Hearts which has only 1 point for each ♡ collected and no rules for breaking ♡ or to *Shoot the Moon*. Black Lady adds in the 13 points for collecting the Q♠, and its offshoot Black Maria adds additional 10 points for the A♠ and 7 for the K♠. We denote the breaking ♡ restriction as Broken Hearts, and when the *Shoot the Moon* scoring is added, we arrive at modern Hearts.

Many variants to modern Hearts can be created by making small mutations which introduce alternate methods of scoring points via collected cards. For instance, Grey Lady reduces the points for Q♠ to only 7 points, while Omnibus Hearts shifts in the other direction to make the J♢ worth -10 points. In Spot Hearts, each ♡ is worth its pip value rather than 1 point. Parlett states that these variants attempt to mitigate the large point value of the Q♠.

One additional variant is Widow Hearts in which the 2♠ is removed, each player is only dealt 12 cards, and the 3 leftover cards are collected at the end of

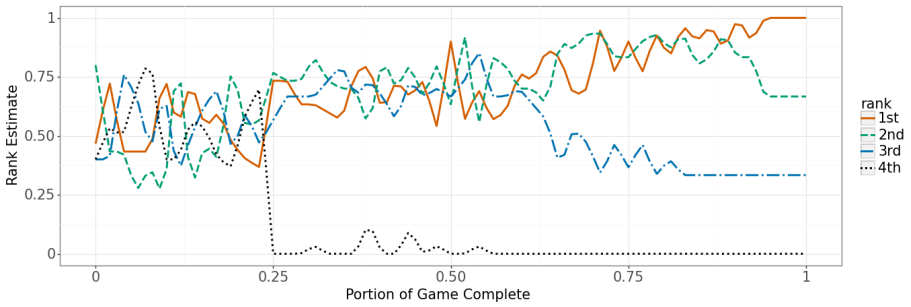


Fig. 2: Lead history for a typical game of Hearts.

the game by the player who wins the last trick. We also examine two distantly related historical cousins in this vein, SlobberHannes and Polignac. Both of these games use a smaller 32 card deck, deleting all cards with rank 6 and below, as well as adding 1 point penalties each for taking the first and last trick. SlobberHannes only assigns the Q♠ one point, while Polignac adds 2 for J♠, and 1 for each other Jack.

4 Heuristic Metrics

For each variant, we encoded the rules using the RECYCLE language and ran simulations in CardStock with a mixture of random and AI players [3]. Random players will make choices using a uniform distribution across each choice, while the AI players will use statistics gathered from random simulations for each choice to determine their best chance of winning.

Our AI players employ a Perfect Information Monte Carlo (PIMC) strategy [11]. When faced with a choice, for each potential move, an AI player creates 10 clones of the current game state, mapping identically all known information from the player's perspective (cards previously played plus cards in their own hand), and creating a random determinization of the hidden information (cards in other player's hands) [19]. In the clone, all players are assigned to make choices randomly. The clone is played out to completion, and each player is assigned a value based on their final rank, scaled so that 1^{st} place is mapped to 1, and 4^{th} place is mapped to 0. In the event of tied ranks, all tied players earn the higher rank. These ranks are accumulated and averaged across all clones, and the move where the AI player earned the highest rank is selected.

To understand of the shape and flow of player experience, we examine *lead histories*, which record the rank estimates for all players every time any AI player makes a move. A lead history will have two dimensions, the estimated player *rank*, and the number of *moves* in the game. As an example, Figure 2 shows the AI estimates of player rank in a typical 4 player game of Hearts. In the beginning of the game, most players are estimated to be in the middle with a good chance of winning. However, we can see a critical point about one quarter

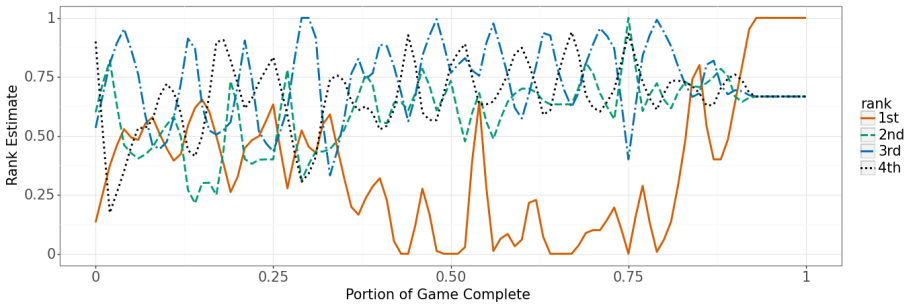


Fig. 3: Lead history demonstrating an AI player winning by *Shooting the Moon*.

of the way through the game where it is clear to everyone the losing player has lost and can never recover.

From these lead histories, we are able calculate four heuristic metrics, adapted from work by Browne and Maire [6]. Each heuristic is calibrated to a 0-1 scale, with 0 being no evidence of this quality, and 1 being high evidence.

4.1 Drama

If a player can come from behind and eventually win a game, we label this as dramatic. Figure 3 shows a dramatic run of Hearts where one player collects all the point cards to *Shoot the Moon*. We can see the winning player only solidifies their win right before the last trick of the game. We observed this behavior in 10% of our simulations.

We define *drama* as the average severity of being in a trailing position for the eventual winner. First, we define *dthresh*, a threshold for drama, set between the top ranked player and the next highest rank, so that when a player estimates their rank above the threshold, they believe it is more likely than not that they will be the winner of the game. Using *np* to denote the number of players,

$$dthresh = \frac{1 + (\frac{np-2}{np-1})}{2} \tag{1}$$

In a two-player game, the threshold will be 0.5, half-way in-between the winning and losing ranks of 1 and 0 respectively. In a four-player game, the threshold will equal $5/6$.

The full *drama* heuristic is then calculated using the winning player's path through the lead history. The number of times the leader is below the threshold is *dcount*. Each time their estimate falls below the drama threshold, the difference between the threshold and the estimate is calculated, and the sum of these differences is averaged. These differences are also scaled with the square root, so that larger differences are weighted more heavily in the final average.

$$drama = \frac{\sum_{i=1}^{dcount} \sqrt{dthresh - est_{winner,i}}}{dcount} \tag{2}$$

4.2 Security

One other calculation related to drama is the notion of the lead security of the winning player. A simple way to determine security is the percentage of the game that the winner was in the lead in the game. While drama can be impacted by just a few poor evaluations, the security heuristic is more stable. Using *dcount* from above, and dividing by the total number of *moves* in the game gives us the following equation.

$$security = 1 - \frac{dcount}{moves} \tag{3}$$

4.3 Spread

When deciding which move to make, an AI will try to determine their chances of winning for each given move. As a player looks at their possible moves in the game, many times they can identify some moves quickly as good and others as bad. Other times, it is difficult to know which move will have the best outcomes. If there is a difference in the win percentage estimates between possible moves, then this is a meaningful choice for the player: they should choose the move that gives them the best estimate. If there is no difference, then the move is meaningless.

By subtracting the minimum estimate from the maximum estimate (which will ultimately be chosen by the player) at each turn, we can calculate the spread at choice i (s_i) between these moves. If we find consistently high spread throughout the whole game, this will indicate that the game is a series of interesting decisions [1].

If we define the number of choices a player has in the game as $|choices|$ then we can determine the degree to which a player has meaningful moves by:

$$spread = \frac{\sum_{i=1}^{|choices|} s_i}{|choices|} \tag{4}$$

4.4 Order

Finally, we wish to determine how much control a player has over their own fate, or if they are at the whims of random events. When the *order* is low, this indicates the AI player has a hard time winning against random chaotic players, but when it is high, the AI player is very successful in determining their success in the game. To calculate the *order* heuristic, first, we record the win percentage ($aiwp$) of the AI player in games with one AI player and the rest Random players. The AI player is always goes first.

Next, we determine the expected win percentage (ewp) for the number of players in the game, assuming that the game is fair. For our games with 4 players, a fair game would expect the first player win 25% of the time. A perfect AI in a perfect information world should be able to win 100% of the games. This is reduced as unaccounted for chaos through hidden information is introduced.

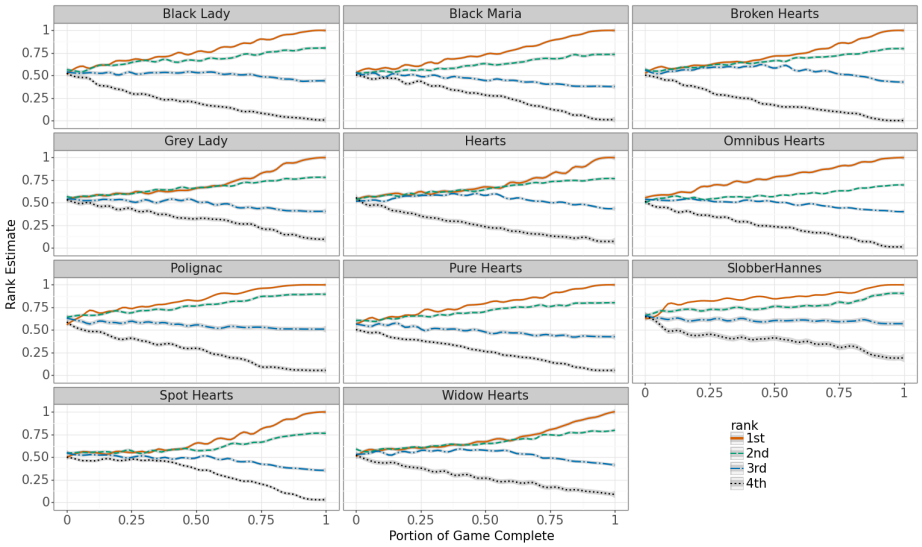


Fig. 4: Averaged lead histories for Hearts and each of the ten variants.

Therefore, we can calculate the *order* of the game by finding the ratio of the *aiwp* gain over the perfect ordered AI gain as follows:

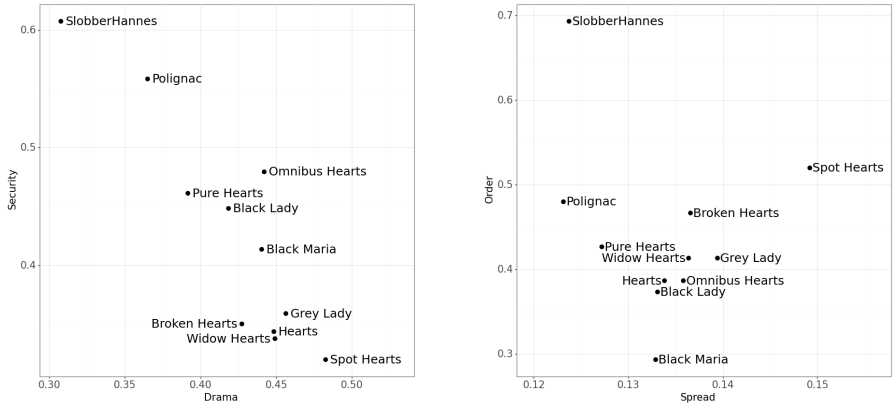
$$order = \frac{aiwp - ewp}{1 - ewp} \tag{5}$$

5 Results

To gather statistics for each variant in CardStock, we ran 100 games with one AI versus three random players, plus 100 games with all AI players. Figure 4 shows the averaged lead histories for each variant. These were calculated by averaging for each rank across the 100 simulations with all AI players.

First, we can trace the effects of historical progression, starting with Pure Hearts. The addition of the Q♠ in Black Lady has a significant impact on the fortunes of the losing player early in the game, steepening their decline. Looking next to Broken Hearts, Parlett states that the hearts-breaking restrictions "feel unnecessary", [14], however in our simulations, this variant has the clear effect of delaying the separation of the top three players until the midgame. On these graphs, there appear to be no large differences when adding in the *Shoot The Moon* rule.

When comparing the point-focused modern variants, it is clear that Spot Hearts and Grey Lady are the most effective at mitigating the Q♠, with Spot Hearts pushing any separation between player's expected ranks until the midgame. Omnibus Hearts with its reward for the J♢ gives the winning



(a) Drama & Security (b) Spread & Order

Fig. 5: Heuristic space quantification of Hearts variants.

player some early separation from the pack, while Widow Hearts delays the final rank determination until the last trick.

We also see a drastic difference in SlobberHannes and Polignac. It appears the the penalty for taking the first trick has a direct impact on the fate of the losing player. Also, due to SlobberHannes having only one penalty card, there is not much variety for all players until the last trick.

Figure 5 shows each variant plotted on each of the metric dimensions of *drama*, *security*, *spread* and *order*, with the pairing chosen to facilitate ease of visualization of the results. First, we can see evidence of evolutionary pressure on Hearts towards higher *drama* and lower *security*. SlobberHannes and Polignac, two early variants, score high on *security* and low on *drama*. The core variants of Pure Hearts and Black Lady fall in the middle of this graph, while Spot Hearts, a modern variant, has highest *drama* and least *security*.

We also find evidence of selective pressure toward higher *spread* scores and more interesting variants, where those variants with more diverse point structure such as Spot Hearts, Grey Lady, and Omnibus Hearts, tend to have larger *spread*. When comparing variants on *order*, there is a less direct connection. However, once again we see SlobberHannes is an outlier, where players have the highest chance to win against random players.

Spot Hearts appears to have many appealing qualities, however, it is not the current dominant variant. We believe this is because for humans, Hearts is meant to be a light game, and time spent calculating the score with each ♡ worth different points is too high when compared to the simple 1 point per ♡ math, and this illustrates a limitation of our computational approach.

Finally, Figure 6 shows a clustering of these variants using the four heuristic metrics described above. We normalized each metric dimension to a range of 0-1, calculated the distance matrix between all variants, and derived a hierarchical clustering using UPGMA [17]. When compared to Figure 1, the uniqueness of

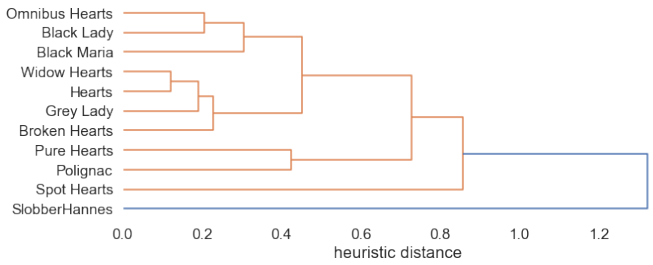


Fig. 6: UPGMA Clustering of Hearts variants in heuristic space.

SlobberHannes and Polignac is clearly present in both ludemic and heuristic space. Spot Hearts, which we noted previously was on the other end of the spectrum from these two cousins, is also found to be quite distant from the other variants, enough to provide a distinct play experience. The closest to Hearts are Widow Hearts, Grey Lady, Broken Hearts, which also matches their underlying ludemic distance.

6 Future Work

There are a number of opportunities for improvement on our current work. While the PIMC players we used can make intelligent decisions, as shown by their ability to *Shoot the Moon*, they are very simple in comparison to more advanced AI methods such as Information Set Monte Carlo Tree Search (ISMCTS) [19] or Conterfactual Regret Minimization (CFR) [4] available through OpenSpiel [10]. Also, we limited our analysis of Hearts variants to one round, however as Neller and Presser [12] demonstrate for the simple dice game Pig, optimal strategy changes when rounds are played within a full game, and we believe the same will hold for Hearts. In addition, the number of heuristics we calculated is small and only gives a window into the full player experience. A richer set of heuristics will provide a larger space for more accurate clustering.

Looking forward, the analysis presented here is easily extendable to other player counts beyond four, and could be used to determine if a game retains the same heuristic qualities under different numbers of players. Finally, we envision creating a full map of the heuristic space of card games, including related trick-taking games such as Spades [2], Doppelkopf [16], and Skat [7], as well as different genres such as shedding, fishing, or press-your-luck. We believe that this map will illuminate connections across families and assist players in finding games suited to their individual taste.

7 Acknowledgements

The authors thank Anna Holmes and Daniel Sweeney for their contributions to the CardStock project, and the insightful comments of our anonymous reviewers.

References

1. Alexander, L.: GDC 2012: Sid Meier on how to see games as sets of interesting decisions. Gamasutra. The Art & Business of Making Games **7**, 2012 (2012)
2. Baier, H., Sattaur, A., Powley, E.J., Devlin, S., Rollason, J., Cowling, P.I.: Emulating human play in a leading mobile card game. IEEE Transactions on Games **11**(4), 386–395 (2018)
3. Bell, C., Goadrich, M.: Automated playtesting with recycled cardstock. Game & Puzzle Design **2**, 71–83 (2016)
4. Brown, N., Lerer, A., Gross, S., Sandholm, T.: Deep counterfactual regret minimization. In: International conference on machine learning. pp. 793–802. PMLR (2019)
5. Browne, C.: Ai for ancient games. KI-Künstliche Intelligenz **34**(1), 89–93 (2020)
6. Browne, C., Maire, F.: Evolutionary game design. IEEE Transactions on Computational Intelligence and AI in Games **2**(1), 1–16 (2010)
7. Buro, M., Long, J.R., Furtak, T., Sturtevant, N.R.: Improving state evaluation, inference, and search in trick-based card games. In: IJCAI. pp. 1407–1413 (2009)
8. Engelstein, G., Shalev, I.: Building blocks of tabletop game design: an encyclopedia of mechanisms. CRC Press (2019)
9. Genesereth, M., Love, N., Pell, B.: General game playing: Overview of the AAAI competition. AI magazine **26**(2), 62–62 (2005)
10. Lanctot, M., Lockhart, E., Lespiau, J.B., Zambaldi, V., Upadhyay, S., Pérolat, J., Srinivasan, S., Timbers, F., Tuyls, K., Omidshafiei, S., Hennes, D., Morrill, D., Muller, P., Ewalds, T., Faulkner, R., Kramár, J., Vylder, B.D., Saeta, B., Bradbury, J., Ding, D., Borgeaud, S., Lai, M., Schrittwieser, J., Anthony, T., Hughes, E., Danihelka, I., Ryan-Davis, J.: OpenSpiel: A framework for reinforcement learning in games. CoRR **abs/1908.09453** (2019), http://arxiv.org/abs/1908.09453
11. Long, J., Sturtevant, N., Buro, M., Furtak, T.: Understanding the success of perfect information monte carlo sampling in game tree search. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 24 (2010)
12. Neller, T.W., Presser, C.G.: Optimal play of the dice game Pig. The UMAP Journal **25**(1) (2004)
13. Niklaus, J., Alberti, M., Pondenkandath, V., Ingold, R., Liwicki, M.: Survey of artificial intelligence for card games and its application to the Swiss game Jass. In: 2019 6th Swiss Conference on Data Science (SDS). pp. 25–30. IEEE (2019)
14. Parlett, D.: A history of card games. Oxford University Press, USA (1991)
15. Parlett, D.: What's a ludeme? Game & Puzzle Design **2**(2), 81 (2017)
16. Sievers, S., Helmert, M.: A doppelkopf player based on UCT. In: Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz). pp. 151–165. Springer (2015)
17. Sokal, R.R.: A statistical method for evaluating systematic relationships. Univ. Kansas, Sci. Bull. **38**, 1409–1438 (1958)
18. Sturtevant, N.R., White, A.M.: Feature construction for reinforcement learning in hearts. In: International Conference on Computers and Games. pp. 122–134. Springer (2006)
19. Whitehouse, D., Powley, E.J., Cowling, P.I.: Determinization and information set monte carlo tree search for the card game dou di zhu. In: 2011 IEEE Conference on Computational Intelligence and Games (CIG'11). pp. 87–94. IEEE (2011)

Appendix A Hearts RECYCLE Code

```
(game (setup
 (create players 4)
 (create teams (0) (1) (2) (3))
 (create deck (game vloc STOCK)
  (deck (RANK (A, K, Q, J, TEN, NINE, EIGHT, SEVEN,
               SIX, FIVE, FOUR, THREE, TWO))
        (COLOR (RED (SUIT (HEARTS, DIAMONDS)))
               (BLACK (SUIT (CLUBS, SPADES)))))))
 (do ((put points 'SCORE (
       ((SUIT (HEARTS)) 1) ((RANK (Q)) (SUIT (SPADES)) 13)))
      (shuffle (game iloc STOCK))
      (all player 'P (repeat 13 (move (top (game iloc STOCK))
                                      (top ('P iloc HAND)))))
      (set (game sto BROKEN) 0)))
 (stage player (end (all player 'P (== (size ('P iloc HAND)) 0)))
   (stage player (end (all player 'P (== (size ('P vloc TRICK)) 1)))
     (choice (
      ((and (== (size (game mem LEAD)) 0)
            (== (game sto BROKEN) 0))
       (any (filter ((current player) iloc HAND) 'NH
        (!= (cardatt SUIT 'NH) HEARTS)) 'C
        (do ((move 'C (top ((current player) vloc TRICK)))
             (remember (top ((current player) vloc TRICK))
                       (top (game mem LEAD)))))))
      ((and (== (size (game mem LEAD)) 0)
            (== (game sto BROKEN) 0)
            (== (size (filter ((current player) iloc HAND) 'NH
                       (!= (cardatt SUIT 'NH) HEARTS))) 0))
       (any ((current player) iloc HAND) 'C
        (do ((move 'C (top ((current player) vloc TRICK)))
             (remember (top ((current player) vloc TRICK))
                       (top (game mem LEAD)))))))
      ((and (== (size (game mem LEAD)) 0)
            (== (game sto BROKEN) 1))
       (any ((current player) iloc HAND) 'C
        (do ((move 'C (top ((current player) vloc TRICK)))
             (remember (top ((current player) vloc TRICK))
                       (top (game mem LEAD)))))))
      ((and (== (size (game mem LEAD)) 1)
            (== (size (filter ((current player) iloc HAND) 'H
                (== (cardatt SUIT 'H)
                    (cardatt SUIT (top (game mem LEAD))))))) 0))
       (any ((current player) iloc HAND) 'C
        (move 'C (top ((current player) vloc TRICK)))))
      (any (filter ((current player) iloc HAND) 'H
            (== (cardatt SUIT 'H)
                (cardatt SUIT (top (game mem LEAD))))) 'C
       ((== (size (game mem LEAD)) 1)
```

```
49          (move 'C (top ((current player) vloc TRICK))))))))
50       (do ((put points 'PRECEDENCE (
51             ((SUIT (cardatt SUIT (top (game mem LEAD)))) 100)
52             ((RANK (A)) 14) ((RANK (K)) 13) ((RANK (Q)) 12)
53             ((RANK (J)) 11) ((RANK (TEN)) 10) ((RANK (NINE)) 9)
54             ((RANK (EIGHT)) 8) ((RANK (SEVEN)) 7) ((RANK (SIX)) 6)
55             ((RANK (FIVE)) 5) ((RANK (FOUR)) 4) ((RANK (THREE)) 3)
56             ((RANK (TWO)) 2)))
57            (forget (top (game mem LEAD)))
58            (cycle next (owner (max (union
59             (all player 'P ('P vloc TRICK))) using 'PRECEDENCE)))
60            ((and (== (size (filter (union (all player 'P
61                   ('P vloc TRICK))) 'PH (== (cardatt SUIT 'PH) HEARTS))) 0)
62                  (== (game sto BROKEN) 0))
63             (set (game sto BROKEN) 1))
64            (all player 'P (move (top ('P vloc TRICK))
65                                 (top ((next player) vloc TRICKSWON)))))))
66   (stage player (end (all player 'P (== (size ('P vloc TRICKSWON)) 0)))
67     (do (((== (sum ((current player) vloc TRICKSWON) using 'SCORE) 26)
68           (dec ((current player) sto SCORE) 26))
69          ((!= (sum ((current player) vloc TRICKSWON) using 'SCORE) 26)
70           (inc ((current player) sto SCORE)
71                (sum ((current player) vloc TRICKSWON) using 'SCORE))))))
72  (scoring min ((current player) sto SCORE)))
```

Appendix B Hearts Encoding in RECYCLE

First, we create the players, followed by a French deck of 52 cards in the `STOCK` location. Next, we create a PointMap for scoring called '`SCORE`, where each ♡ is given a score of 1, and the Q♠ is given a score of 13. The cards in the `STOCK` are shuffled, and each player is dealt 13 cards into their `HAND`. Finally, we set an integer called `BROKEN` to track if any Hearts have been played. At the beginning of each round, this is set to 0.

The hand stage ends when all the players have no cards left in their `HAND` location. Inside this stage, we have another stage, which will end when each player has played one card to their `TRICK` location.

In Hearts, the current player has a choice between *five* distinct exclusive options. First, if they are the first player (determined by asking if there is a card in the memory location `LEAD`), and ♡ have not been broken (still has a value of 0), we create a filter that contains all cards from their `HAND` where the suit is not ♡. When processed, these cards will get the temporary variable name `'C`. The current player can play any one of these cards to their `TRICK` location. After they play their card, the game remembers it in the `LEAD` location, for everyone to reference as the trick progresses around the table.

Second, if they are the first player, and ♡ have not been broken, but they have no cards that are not ♡, then they can play any card from their `HAND` to

their TRICK location. Again, after they play their card, the game remembers it in the LEAD location.

Third, if they are the first player, but ♡ *has* been broken, then they can play any card from their HAND to their TRICK location. Again, after they play their card, the game remembers it in the LEAD location.

Fourth, if they are not the first player (determined by seeing that there is already a card in the LEAD memory location), and they cannot follow the suit of the card that was led, then they can play any card from their HAND to their TRICK location.

Finally, if they are not the first player, and they *do* have a card that can follow suit, then they can play one of these card with a matching suit to their TRICK location.

Once the inner trick stage ends, then the game determines the winner of the trick. We define another PointMap called 'PRECEDENCE to sort the cards from highest to lowest rank. In this map, we add in an extra 100 points for the suit that was led, so that this initial card and cards that follow suit will be ranked higher than other cards that did not follow suit.

With the map created, we no longer need to remember the LEAD card, so we forget it. Now, we use the 'PRECEDENCE map to determine who won, by finding the owner of the card that gets the maximum value of all cards played to TRICK locations. This player is then set to be the next player in the cycle for this round stage, and will go first next trick.

If anyone played ♡ this trick, and ♡ has not been broken, then it is now broken by setting the BROKEN variable to 1. Next, all players will move their TRICK card to the TRICKSWON location of the winning player for scoring.

Once the thirteen hands are over, it is time to determine each player's score. If a player has scored 26 points this turn, using the 'SCORE map from above, they have collected every ♡ and the Q♠, thus *Shoot the Moon*. In this case, their score will be decremented by 26 points. In all other cases of scoring less than 26, their points will be added to their SCORE. Finally, we specify that Hearts is a game where you win by having the least points.

In the code above, highlighted sections denote portions of code that are modified to create the variants studied. Changes to the Orange code in line 6 shrink the deck for Polignac and SlobberHannes. Alterations to the Grey code in lines 9 and 10 will accommodate different point structures. The Red sections in lines 14, 18-34, and 60-63 capture the idea that ♡s must be broken before being led, while the Blue sections in lines 67-69 allow for players to Shoot the Moon.