A Heuristic Search Algorithm for Finding Unsolvable Deals in the Birds of a Feather Solitaire Game

Maximilian Kahn
Quest University Canada
3200 University Boulevard, Squamish
British Columbia, Canada

Abstract

In this paper a heuristic search algorithm is presented that is able to quickly group deals into the categories of solvable or unsolvable. A deal is provably unsolvable if the adjacency matrix of the deal contains no spanning trees. The heuristic search algorithm first checks to see if the deal contains spanning trees, if it does, a heuristic check is done on the deal which determines if it is likely to be unsolvable or not.

Introduction

Birds of a Feather (BoaF) is a solitaire game invented by Todd Neller, Professor of Computer Science at Gettysburg College (Neller, 2016). BoaF provides an excellent opportunity for undergraduate research in the field of AI and games, as BoaF is simple enough to immediately begin investigating using programmatic or mathematical techniques.

In this paper the 4×4 BoaF setup is investigated, where 4 rows of 4 French-playing cards are dealt. (While BoaF has been restricted to a 4×4 grid in this paper, BoaF can be played on any $N \times M$ grid where $N > 0$ and $M > 0$.) Each card can be thought of as being a single-layer stack. In order to win, there must only be one stack of cards remaining.

The rules of BoaF dictate that a card can only be stacked on top of another card if they are in the same row or column. Furthermore, the chosen card must be compatible with the target card; two cards are compatible if either of the following are true: the pair of cards share the same suit - or, the cards are adjacent or equal in rank. For example, The queen of diamonds (QD) and the jack of clubs (JC) are compatible because JC is adjacent in rank to QD. Likewise, the ten of diamonds (TD) and the ace of diamonds are compatible because both share the same suit. In BoaF, aces are low and so are only adjacent to twos - not kings. An example of a BoaF deal is illustrated in table 1.

A solution to this deal is 3D-2D 4S-7S 5C-8C 4S-5C 4S-3D QS-KS 8S-QS 4S-TS 7C-8H 6H-JH 9H-5H 8S-9H 7C-6H 8S-7C 4S-8S. In other words, move the three of diamonds onto the two of diamonds, then move the four of spades onto the seven of spades, etc.

Once the chosen card has been moved on top of the target card, the target card in effect has been *consumed* by the chosen card. The final move, 4S-8S, will result in one stack remaining on the grid, namely, 4S.

8C	8H	8S	7S
6H	JH	5H	9H
5C	7C	KS	4S
2D	10S	QS	3D

Table 1: A BoaF deal

8C	8H	8S	7S
6H	JH	5H	9H
5C	7C	KS	4S
3D	10S	QS	

Table 2: 3D-2D

Research Motivation

As a result of BoaF being newly-created, many aspects of the game are currently unknown. The goal of this research was to develop an algorithm that can quickly identify unsolvable deals. In order to create a fast search algorithm, two design requirements were adhered to. First, the search algorithm should not use a brute-force method to identify unsolvable deals, instead, the algorithm should be able to determine whether a deal is unsolvable from the deal's initial state. Second, the properties of the initial state of each deal should be investigated through matrix operations (e.g. determinant, eigenvalues, RREF, etc.).

The first design requirement is an ambitious goal as it is not clear whether there is a way of discerning between all unsolvable deals and solvable deals simply from the initial state of the deal. Indeed, this may be deemed one of the holy grails of BoaF research as such a result implies a deep understanding of what makes a deal generally unsolvable. The second design requirement seems to be the next logical step from the first requirement as matrix operations allow for a relatively simple way of categorizing matrices (or in our case, the adjacency matrices of BoaF deals).

Definitions

Given that BoaF is a relatively new game, the landscape is ripe for the introduction of new words to describe BoaF re-

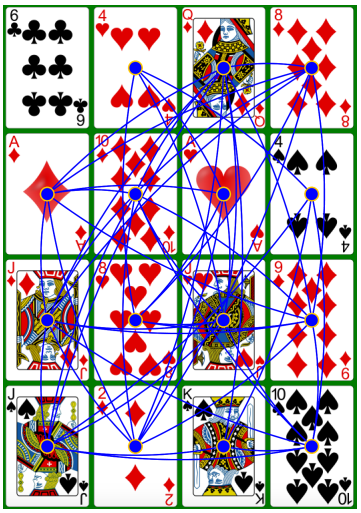

Figure 1: A deal containing an odd bird.

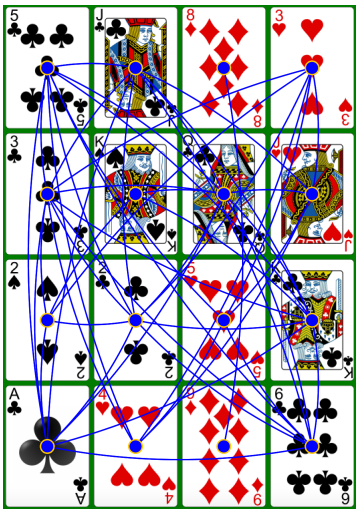

Figure 2: A deal containing a separated flock.

lated phenomena. First, a *deal* is defined to be a 4×4 grid of french-playing cards. Based on the rules of BoaF, a deal may be solvable or unsolvable. There are three kinds of unsolvable deals. The first is the *odd bird unsolvable deal* (OBUD) (Neller, 2018a.). An odd bird is a card which has no compatible pairs on the grid. Even if all the other cards can be moved such that they reduce to one stack, the odd bird will still remain, resulting in a minimum of two stacks for such a deal; thus, deals containing odd birds are unsolvable. An example of an OBUD is given in figure 1. The blue lines indicate compatibility - note how the six of clubs is not connected to any other card, it is an odd bird.

It is also possible to have a *separated flock unsolvable deal* (SFUD) (Neller, 2018b.). A separated flock is when a group of cards are compatible with each other but not with the rest of the grid. Because a separated flock can at best be reduced to one card, deals containing separated flocks can at best be reduced to two cards and are therefore unsolvable. The eight and nine of diamonds in figure 2 are only connected to each other, this is an example of a SFUD.

There is a simple way of determining whether a deal is an OBUD or SFUD. Both kinds of deals when thought of as graphs (each card being a vertex and the compatibility between cards being the edges) are the only deals that do not contain any spanning trees. Given a graph G, G is said to contain a spanning tree if there exists a subgraph T of G in which any two vertices are connected by exactly one edge, every vertex is reachable, and the minimum number of edges have been used to connect the vertices in the graph (Newman, 2018, p.121). In figure 1, it is clear that the deal contains no spanning trees because the 6C is not reachable from any of the other cards. Likewise, the deal in figure 2 contains no spanning trees because neither the 8D nor 9D is reachable from the other cards in the grid.

Unfortunately, there does exist another class of unsolvable BoaF deals that do contain spanning trees. Let us refer to deals of this kind as *full-flock unsolvable deals* (FFUD) since the graphs of such deals are *connected* but are nevertheless unsolvable.

Seed Number	Pair-Count	Determinant	Zeros
1163	38	5	88
6727	40	28	104
12226	41	32	76
24555	41	0	74
25482	40	80	94
Average FFUD	40	29	87
Average Deal	48	682	21

Table 3: Shown are all FFUDs below deal 30,000. Pair-count records the number of compatible pairs in the deal. The determinant column records the determinant of the deal's adjacency matrix. The zeros column records the number of zeros in the squared adjacency matrix. The average FFUD's properties are the mean value from the five FFUDs; the average deal's properties are the mean value from all 30,000 deals

Finding Full-Flock Unsolvable Deals

Using Neller's depth-first-search algorithm (Neller, 2018c.) to find unsolvable deals, the first 30,000 deals were classified as either solvable or unsolvable.

Five FFUDs were found in the first 30,000 deals, they are listed in table 3. Indeed, these FFUDs possess properties that are notably different from the average deal.

That the FFUDs in table 3 contain low pair-counts is not surprising; we would expect that fewer connections between cards would lead to a higher chance that a deal would be unsolvable since fewer connections means fewer opportunities to make moves. However, using this as the only heuristic is risky as there is nothing to suggest FFUDs cannot exist with higher pair-counts.

The determinant at first glance appears to be a heuristic that could do well to identify FFUDs. However, caution must be exercised here to avoid overfitting the heuristic to the training data. Excluding deal 25482, it could be hypothesized that FFUDs generally possess determinants less than say, 50, and yet deal 25482 shows that FFUDs can possess determinants much higher than 50.

Squaring the 16×16 adjacency matrix A of deal D results in a new 16×16 matrix where ${A^2}_{ij}$ represents the number of walks of length two from vertex i to vertex j (Newman, 2018, p.131-p.132); a walk being a sequence of vertices in a graph where each pair of vertices is connected by an edge.

If ${A^2}_{ij} = 0$, there are no walks of length two (henceforth known as 2-walks) between vertices i and j. Having many vertices that do not share 2-walks is another measure of graph connectedness (i.e. of a poorly connected graph) and is an indicator of a possible FFUD as can been seen from table 3.

While FFUDs tend to have a low pair-count, low determinant, and high number of zeros in the squared adjacency matrix (these three properties together shall be referred to henceforth as the *3-heuristic*), there are solvable deals that share these same properties. So to be clear, a deal displaying the 3-heuristic does not guarantee the deal is an FFUD.

A search algorithm using the 3-heuristic as a search criteria has two conflicting problems to balance. If the numbers in the 3-heuristic are too narrow (such as discarding any deal with a pair-count less than 42) then there is a risk that the search will miss FFUDs that possess atypical properties. However, if the search criteria is too broad then the search will be inefficient and too many false-positive deals (a false-positive deal being a deal that satisfies the search criteria but is solvable) will be grouped with the genuine FFUDs.

Informed by table 3, the 3-heuristic is defined as follows:

1. $abs(det(A)) \leq 100$
2. $pair - count < 46$
3. $zeros(A^2) > 70$

Where A is the adjacency matrix of deal D. The absolute value of the determinant is taken because some determinants turn out to be negative. The zeros function counts the number of zeros in the given matrix. The values above were chosen to allow for some leeway given that there may be FFUDs with atypical properties but also keeping in mind that for these three heuristics picking values that are too relaxed will result in more false-positives.

The Search Algorithm

The heuristic search algorithm for finding unsolvable deals can be simplified to these 6 steps given deal *D*:

1. Construct the adjacency matrix *A* for deal *D*.
2. Find the number of spanning trees in *A* using Kirchoff's matrix-tree theorem (Skiena, 1990, p.235).
3. If the number of spanning trees is 0, then *D* is either an OBUD or SFUD. Store this result. Return to step 1 for deal *D + 1*. If *D* possesses a nonzero number of spanning trees, then continue.
4. If $abs(det(A)) \leq 100$, continue. Otherwise, return to step 1 for deal *D + 1*.
5. If $pair - count < 46$, continue. Otherwise, return to step 1 for deal *D + 1*.
6. If $zeros(A^2) > 70$, store *D* as a possible FFUD and then return to step 1 for deal *D + 1*. Otherwise, return to step 1 for deal *D + 1*.

Results

Using the above 3-heuristic search algorithm, the first 100,000 deals were analyzed. Of the 100,000 deals, 181 deals were either classified as OBUD or SFUD, and 156 deals were classified as possible FFUDs. Of the 156 possible FFUDs, 147 turned out to be false-positives; the algorithm successfully categorized the 5 FFUDs from table 3 as well as 4 new FFUDs, specifically deals: 38711, 45088, 59481, and 93196. Neller's depth-first search algorithm was used to verify no FFUDs were missed in this search.

Conclusion

In this paper, the 3-heuristic search algorithm is presented which is able to categorize deals as solvable or unsolvable. This algorithm was tested on 100,000 deals (seeds 0 - 100,000) and was able to successfully categorize all 9 FFUDs while only finding 147 false-positives.

The 3-heuristic search algorithm can be viewed as a sieve. The strength of the algorithm lies in the fact that after running this preliminary search, most of the unsolvable deals have already been found (OBUDs and SFUDs) and the rest are in the smaller group of possible FFUDs. Instead of having to verify the solvability of 100,000 deals, only 156 deals need to be verified.

There are a variety of future research questions related to the work done in this paper. First, while the proposed algorithm seemed to work well for the 100,000 deals tested, there is little formal understanding of why the 3-heuristic method chosen by the author works well to identify FFUDs. The author encourages others to continue the work presented here and hopes a greater understanding of BoaF can be achieved through the combined effort of the AI research community.

For those interested in testing the 3-heuristic search algorithm, the author has written up the program in Python to be used with Todd Neller's Python distribution of BoaF; the code can be found at: https://github.com/Starfunk/birds-of-a-feather.

References

Neller, T. W. 2018. "Birds of a Feather" Solitaire Card Game. [Online; accessed 20-August-2018].

Neller, T. W. 2016. AI Education: Birds of a Feather in AI MATTERS. Paper presented in AI Matters "AI Education Matters" Column: Vol. 2, Issue 4, Summer 2016.

Newman, M. 2018. Networks - An Introduction. Oxford University Press.

Skiena, S. 1990. Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Reading, MA: Addison-Wesley.

Acknowledgements

I would like to thank Todd Neller for creating this fantastic opportunity for undergraduate research. I would also like to thank Richard Hoshino for his continued support and for introducing me to the BoaF challenge during AAAI 2018.