Predicting Unsolvable Deals in the Birds of a Feather Solitaire Game

Richard Hoshino and Maximilian Kahn

Quest University Canada, Squamish, British Columbia, Canada

Abstract

In this paper, we analyze Birds of a Feather (BoaF), a solitaire game played with 16 cards. While the large majority of deals are solvable, the set of unsolvable deals share certain characteristics that can be determined from the adjacency matrix of the corresponding "compatibility graph". We create a binary decision tree based on just three variables to predict whether a given deal is solvable. Our predictive model, tested on 30,000 random deals, correctly classifies over 99.9% of our data.

Introduction

In the 2019 EAAI Undergraduate Research Challenge, participants were invited to analyze Birds of a Feather (BoaF), a perfect-information one-player card game (Neller 2016). Like other *open solitaire* games, the 16 cards in each BoaF game are dealt face-up. Thus, each initial configuration can be classified as either solvable or unsolvable.

The 15-puzzle is a famous open solitaire game, consisting of fifteen tiles numbered 1 to 15 randomly arranged in a 4×4 frame, with one tile missing. The object of the game is to use the empty space to slide the tiles, and create a configuration where the fifteen tiles are placed in order. A simple parity argument (Johnson and Story 1879) shows that half of the starting positions are unsolvable. Knowing the set of solvable starting positions allows game designers to avoid presenting users with an impossible puzzle.

Conversely, FreeCell is an open solitaire card game where 52 cards are dealt randomly, without any consideration of whether a given deal is solvable. Although nearly every FreeCell deal is solvable, one out of every 78,000 deals is not (Keller 2018). Despite much effort, no Microsoft FreeCell player could solve the complete set of 32,000 deals, and it was only through an extensive crowdsourcing effort that it was shown that one of the deals is indeed unsolvable.

In this paper, we analyze the recently invented BoaF game and describe our efforts of creating a model to quickly and accurately predict whether a deal is solvable. After presenting the rules of the game, we describe our three-variable binary decision tree model, show that our model makes the correct classification in over 99.9% of random deals in our testing set, and conclude with ideas for future research.

Rules of the Game

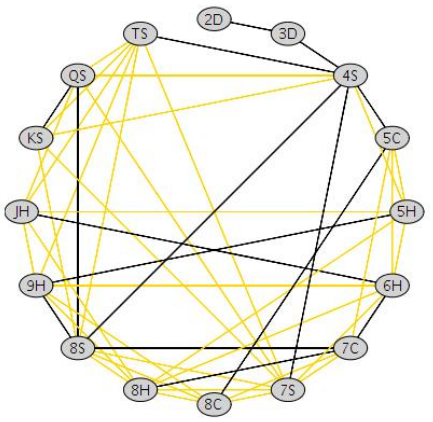
BoaF is played with a standard 52-card deck, with 4 suits (C, H, S, D) and 13 ranks of each suit (from A to K). Below is an example of an initial configuration, where 4 rows of 4 cards are dealt face-up, representing 16 one-card "stacks".

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

Any stack of cards can be picked up and placed on top of another stack in the same row or column, as long as the top card of each stack is either the same suit or their ranks differ by at most one. For example, 8C can be placed on top of 7S.

For any deal, a game is solvable if and only if there exists a sequence of 15 moves that creates a single stack of 16 cards. Our deal above is solvable, as one possible 15-move solution 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. This results in a single stack with 4S as the top card.

Given a deal, define its *compatibility graph* G as follows: each of the 16 cards is a vertex, and two vertices are adjacent if and only if their suits match or their ranks differ by at most one. By definition, the 15-move solution to any deal must be a *spanning tree* of its compatibility graph G, as we see from the dark edges of the compatibility graph below.



Predictive Model

For any BoaF deal, we can create its compatibility graph G. We define the following three variables.

1. $nw_1(G)$, the number of pairs (i, j) with $1 \leq i < j \leq 16$ for which there is no walk of length 1 from i to j in G.
2. $nw_2(G)$, the number of pairs (i, j) with $1 \leq i < j \leq 16$ for which there is no walk of length 2 from i to j in G.
3. $st(G)$, the number of spanning trees in G.

Let A be the 16×16 *adjacency matrix* of graph G, where $A_{i,j} = 1$ if vertices i and j are adjacent, and $A_{i,j} = 0$ otherwise. By definition, $nw_1(G)$ is the number of zeros above the upper diagonal of A and $nw_2(G)$ is the number of zeros above the upper diagonal of A^2. Given the adjacency matrix A, we can apply Kirchhoff's Matrix-Tree Theorem (Harris, Hirst, and Mossinghoff 2008) to calculate $st(G)$.

If $nw_1(G)$ and $nw_2(G)$ are large, then there are fewer connections between cards, implying fewer possible moves. In this light, we surmise that large values of $nw_1(G)$ and $nw_2(G)$ are indicators for a deal's unsolvability.

If G is not connected, then a spanning tree cannot exist, thus ensuring that any sequence of moves will result in a minimum of two leftover stacks. Thus, $st(G) = 0$ is a sufficient condition for a deal to be unsolvable. (But as we'll see, $st(G) = 0$ is not a necessary condition for unsolvability.)

We create our predictive model on 100,000 random 4×4 deals, representing seeds 1 to 100,000 of the open-source FreeCell shuffler (Mol 2018). The first 70,000 deals represent the training set and the final 30,000 deals represent the testing set. For each of these deals, we apply an independent codebase (Neller 2018) that uses a depth-first search algorithm to determine whether the deal is unsolvable.

In the training set, just 143 of the 70,000 deals are unsolvable, with 135 having $st(G) = 0$ and 8 having $st(G) > 0$. Of the 8 unsolvable deals with a connected compatibility graph G, the values of $nw_1(G)$ range from 76 to 82 and the values of $nw_2(G)$ range from 37 to 56. These numbers are significantly higher than the averages of these values for our training set: 71.7 for $nw_1(G)$ and 10.7 for $nw_2(G)$.

Our predictive model is a binary decision tree, where each split maximizes the information gain of the training set. The resulting model can be expressed as a six-line program.

```
if st(G)==0: return "UNSOLVABLE"
else:
  if nw2(G)<74: return "SOLVABLE"
  else:
      if nw1(G)>75: return "UNSOLVABLE"
      else: return "SOLVABLE"
```

On the 70,000 deal training set, this model correctly classifies 143 out of the 143 unsolvable deals and 69767 out of the 69857 solvable deals, for an overall accuracy of 99.87%.

Applying this model to the 30,000 deal testing set, we discover that this decision tree correctly classifies 47 out of the 47 unsolvable deals and 29925 out of the 29953 solvable deals, for an overall accuracy of 99.91%. On this testing set, our "unsolvability prediction algorithm" has a precision of $\frac{47}{47+28} = 62.7\%$ and a recall of $\frac{47}{47+0} = 100\%$.

Conclusion

Our predictive model can be viewed as a sieve that quickly identifies deals that may be unsolvable, so that a BoaF-playing app only outputs deals that have a solution. High recall is more important than high precision, since it is better to incorrectly classify deals as unsolvable (and miss out on a few deals) than incorrectly classify deals as solvable (and give the player an impossible deal).

Unlike the 15-game tiling puzzle, there does not exist a simple test for unsolvability, especially as it is NP-complete to determine whether an arbitrary $N \times N$ BoaF deal is solvable (Hoshino and Notarangelo 2019).

Though our model's recall rate is excellent, it is not perfect. To illustrate, consider the following deal where the 16 cards form a mutually orthogonal Latin square of order 4.

2C	4S	6H	8D
6D	8H	2S	4C
8S	6C	4D	2H
4H	2D	8C	6S

Since no pair of cards in the same row or column can be matched, this deal is unsolvable. However, our algorithm makes an incorrect classification as $st(G) > 0$ (because the graph is connected) and $nw_2(G) = 0$ (because there are two different walks of length 2 between each pair of vertices).

There are several directions for future research. First, we can test our model on more data, especially as the 10^5 compatibility graphs we analyzed are just a fraction of the $\binom{52}{16} \sim 10^{13}$ possible combinations. Also, we can analyze which edges in the compatibility graph also match by position, i.e., share the same row or column. To do this, we need to consider a *stackability* graph. Finally, we'd like to add more variables beyond the three considered in this paper.

The student author has created a repository of all the Python code used in this paper. The code can be found at https://github.com/Starfunk/birds-of-a-feather.

References

Harris, J. M.; Hirst, J. L.; and Mossinghoff, M. J. 2008. *Combinatorics and Graph Theory*. Springer.

Hoshino, R., and Notarangelo, M. 2019. Computational Intractability and Solvability for the Birds of a Feather Game. In *EAAI-19: Proceedings of the Ninth Symposium on Educational Advances in Artificial Intelligence*.

Johnson, W. W., and Story, W. E. 1879. Notes on the 15 puzzle. *Americal Journal of Mathematics* 2(4):397–404.

Keller, M. 2018. Freecell: Frequently Asked Questions. `http://www.solitairelaboratory.com`.

Mol, M. 2018. Rosetta Code: Deal cards for Freecell. `https://rosettacode.org/wiki/Deal_cards_for_FreeCell#Java`.

Neller, T. W. 2016. AI education: Birds of a Feather. *AI Matters* 2(4):7–8.

Neller, T. W. 2018. Birds of a Feather Solitaire Card Game. `http://cs.gettysburg.edu/~tneller/puzzles/boaf/index.html`.