Computational Intractability and Solvability for the Birds of a Feather Game

Richard Hoshino, Max Notarangelo

Quest University Canada
Squamish, British Columbia, Canada

Abstract

In this paper, we analyze Birds of a Feather (BoaF), a perfect-information one-player card game that is the subject of the 2019 EAAI Undergraduate Research Challenge. We prove that the generalized $N \times N$ BoaF game is NP-complete, and then explore the one million deals in the 4×4 BoaF testbed. We present several graph-theoretic algorithms to prove that 1880 of these million deals are unsolvable, and conclude the paper with two search algorithms that efficiently show that all of the remaining 998,120 deals are in fact solvable.

Games are the ideal platform for Artificial Intelligence research, as games are accessible to a general audience while providing highly constrained contexts that enable computer scientists to determine optimal decision-making strategies. A recent textbook (Yannakakis and Togelius 2018) explores the applications of games to AI, including procedural content generation and computational creativity.

There is a large body of work on one-player combinatorial games, also known as *puzzles* (Demaine 2018). Popular examples of puzzles include Tetris, Sudoku, and Minesweeper, which are among a collection of puzzles that are known to be NP-complete (Kendall, Parkes, and Spoerer 2008).

An active research area is the analysis of *open solitaire card games*, due to its application to classical planning problems (Ghallab, Nau, and Traverso 2004). In these perfect-information puzzles, all the cards are dealt face-up. By eliminating the element of chance, we can determine the effect of any sequence of moves, and generate powerful algorithms for efficiently solving deals without exhaustive search.

FreeCell is an example of an open solitaire NP-complete game, where there exist sophisticated methods to solve random deals. One team of researchers has examined 10^8 FreeCell deals, showing that only 1282 are unsolvable, corresponding to one loss every 78,000 deals (Keller 2018).

The 2019 EAAI Undergraduate Research Challenge was to analyze Birds of a Feather (BoaF), a recently invented open solitaire card game (Neller 2016). After defining this game, we prove that the $N \times N$ version of BoaF is computationally intractable, i.e., NP-complete. We then present various algorithms for determining the solvability of a deal, and fully analyze the million deals in the 4×4 BoaF testbed.

Definition and Rules

Birds of a Feather is played with a standard 52-card deck, with 4 suits and 13 ranks of each suit from Ace to King. The deck is shuffled and 16 cards are dealt, forming 4 rows and 4 columns. The remaining cards are not used.

Each card initially consists of a 1-card "stack." 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. A *move* denotes the process of joining two stacks.

Two cards in the same row or column are defined as a *match by position*, and two cards that share a suit, rank, or are of adjacent rank, are a *match by card*. In any legal move, we must have a match by position and by card. In this game, Aces and Kings are not considered adjacent, so the Ace of Diamonds (AD) cannot be placed on top of the King of Clubs (KC), even if they share the same row or column.

For any initial state (i.e., a deal), the game is won if we can determine a sequence of 15 moves that creates a single stack of 16 cards. We say that such a deal is *solvable*. If no such sequence exists, then we say that the deal is *unsolvable*.

Consider the random deal given in Table 1.

JD	2D	9H	JC
5D	7H	6C	5H
KD	KC	9S	5S
AD	QC	KH	3H

Table 1: The initial board state of a BoaF deal

For instance, 7H can be moved on top of 6C, since $|6-7| = 1$. This move results in Table 2, with 6C being "eliminated" from the board.

JD	2D	9H	JC
5D		7H	5H
KD	KC	9S	5S
AD	QC	KH	3H

Table 2: The board state after the move 7H $\rightarrow$ 6C

We now make the following move sequence: 9H $\rightarrow$ 9S, 5H $\rightarrow$ 5S, 7H $\rightarrow$ 9H, KH $\rightarrow$ 7H, KH $\rightarrow$ 5H, KH $\rightarrow$ 3H, QC $\rightarrow$ KH, QC $\rightarrow$ JC. This results in Table 3.

JD	2D		QC
5D			
KD	KC		
AD			

Table 3: The board state after nine moves

Note that KD can be stacked on each of the remaining cards. So we use KD to solve this deal, via a sequence of moves analogous to how a rook moves in chess. Specifically, the sequence KD $\rightarrow$ KC, KD $\rightarrow$ 2D, KD $\rightarrow$ QC, KD $\rightarrow$ JD, KD $\rightarrow$ 5D, KD $\rightarrow$ AD results in the end state with a single stack in the bottom-left corner, with KD as the top card.

Naturally, there exist deals that are unsolvable. For an extreme case, consider Table 4, where the 16 cards form a mutually orthogonal Latin square of order 4. Since no pair of cards can be matched, the best we can do is leave 16 stacks of a single card (colloquially referred to as "odd birds"), with no two birds able to "flock together."

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

Table 4: A BoaF deal that is unsolvable

A brute-force approach determines whether a deal is solvable. There are at most $16 \cdot 6$ ways to make the first move, since there are 16 options to choose the card that moves, and 6 possible locations for that card (3 in the same row and 3 in the same column). There are at most $15 \cdot 6$ choices for the second move, $14 \cdot 6$ for the third move, and so on. Since the number of steps is at most $16! \cdot 6^{16}$, the 4×4 BoaF game can be solved in constant time.

Of course, there are much better ways to verify the solvability of a deal, and we will present several heuristics later in this paper. These algorithms efficiently find a sequence of moves to complete a solvable deal, and quickly confirm that an unsolvable deal has no single-stack solution.

Let us define the $N \times N$ BoaF game, where the N^2 cards are randomly dealt from a deck of size $SR \geq N^2$, where there are S suits and R ranks of each suit. If there exists a sequence of $N^2 - 1$ moves that turns an initial deal into a single N^2-card stack, then we say that the deal is solvable.

We denote each card as r_s, where r is the rank and s is the suit. For the purposes of the analysis that follows, we will assume that $1 \leq r \leq L$ and $1 \leq s \leq L$, where L is some sufficiently large number for which $L \geq N$. Two stacks with top cards r_s and $r^*_{s^*}$ can be joined if and only if $|r - r^*| \leq 1$ or $s = s^*$, and the two cards match by position.

Given the success of our heuristics for $N = 4$, we might conjecture that there is a simple polynomial-time algorithm to verify whether a given $N \times N$ BoaF deal is solvable. We show that this is not the case, by proving that this decision problem is NP-complete. To do this, we obtain a reduction from 3-SAT, the well-known NP-complete problem on Boolean satisfiability (Garey and Johnson 1979).

Proof that BoaF is NP-complete

Let $S = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be the conjunction of m clauses with three literals on the variables $u_1, u_2, \ldots, u_v$. From S we will define BoaF(S), a deal with $N = 12m + 2v + 1$ rows and $N = 12m + 2v + 1$ columns. We will prove that S is satisfiable if and only if BoaF(S) is solvable, thus establishing the desired reduction from 3-SAT.

For example, let $S = C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_7$, where $C_1 = (u_1 \vee \overline{u_2} \vee u_4)$, $C_2 = (u_2 \vee \overline{u_3} \vee u_4)$, $C_3 = (\overline{u_1} \vee u_3 \vee u_4)$, $C_4 = (u_1 \vee \overline{u_2} \vee \overline{u_4})$, $C_5 = (u_2 \vee \overline{u_3} \vee \overline{u_4})$, $C_6 = (\overline{u_1} \vee u_3 \vee \overline{u_4})$, and $C_7 = (u_1 \vee u_2 \vee u_3)$. By definition, S is an instance of 3-SAT.

We can show that $(u_1, u_2, u_3, u_4) = (T, T, T, F)$ is a truth assignment that satisfies the Boolean formula S, i.e., each clause C_j evaluates to TRUE for all $1 \leq j \leq 7$. It is straightforward to verify that (T, T, T, F) and (T, T, T, T) are the only 4-tuples that satisfy S.

Thus, if we construct the 8-clause formula $S^* = S \wedge C_8$, where $C_8 = (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$, then we immediately see that S^* is not satisfiable.

The construction of BoaF(S) involves the concatenation of m different "blocks" with 12 rows and $12m + 2v + 1$ columns, with one block corresponding to each clause, followed by $2v + 1$ additional rows concatenated at the bottom. In the example above, S has $m = 7$ clauses and $v = 4$ variables. By the definition of 3-SAT, note that $v \leq 3m$.

The first $2v$ columns correspond to the $2v$ possible literals in S, and will be denoted by the labels $u_1, \overline{u_1}, \ldots, u_v, \overline{u_v}$. The next $6m$ columns will be used by the m blocks, with each clause having six unique columns. These columns will be labeled $C^1_j, C^2_j, \ldots C^6_j$, for each $1 \leq j \leq m$. The final $(12m + 2v + 1) - 2v - 6m = 6m + 1 > 0$ columns will not be labeled, since they are not important in our construction.

Table 5 illustrates the block for $j = 1$, corresponding to $C_1 = (u_1 \vee \overline{u_2} \vee u_4)$. We omit any columns with no entries.

u_1	$\overline{u_1}$	u_2	$\overline{u_2}$	u_4	$\overline{u_4}$	C^1_1	C^2_1	C^3_1	C^4_1	C^5_1	C^6_1
2_1	4_4					1_{71}	5_{71}				
		2_5	4_2					1_{72}	5_{72}		
				2_3	4_6					1_{74}	5_{74}
7_1			L_2								
7_2			3_5								
7_3			3_8								
			8_1	3_6							
			8_2	L_3							
			8_3	3_9							
3_4				9_1							
3_7				9_2							
L_1				9_3							

Table 5: The block corresponding to $C_1 = (u_1 \vee \overline{u_2} \vee u_4)$

Each block will have this form, where the cards in the j^{th} block will use the ranks from $10j - 9$ to $10j - 1$ and the suits from $10j - 9$ to $10j$, in addition to three cards with rank L and six cards with suits ranging from $10m + 1$ to $10m + v$.

Since $C_1 = (u_1 \vee \overline{u_2} \vee u_4)$, the 18 cards with ranks in the set $\{10j - 7, 10j - 3, 10j - 2, 10j - 1, L\}$ appear in the three columns corresponding to these literals.

In the first three rows, the cards in the first six columns alternate between $10j-8$ and $10j-6$. The suits of these six cards will all be different, chosen from the set $\{10j-9, 10j-8, 10j-7\}$ if that column's literal is in clause C_j, and from the set $\{10j-6, 10j-5, 10j-4\}$ if it is not.

The cards in the final six columns alternate between $10j-9$ and $10j-5$. Each pair of cards will be assigned the suits $10m+p$, $10m+q$, $10m+r$, where $\{p,q,r\}$ are the indices of the literals in C_j. We will assume that $p < q < r$, since we can disregard tautological clauses such as $C_j = (u_p \vee u_q \vee \overline{u_q})$ from our Boolean formula S.

Table 6 illustrates the block for $j = 6$, corresponding to $C_6 = (\overline{u_1} \vee u_3 \vee \overline{u_4})$. We see that this is similar to Table 5, except that the suits and ranks have been incremented by 50, and the 18 cards in the bottom nine rows have been placed directly below the columns labeled $\overline{u_1}$, u_3, and $\overline{u_4}$.

u_1	$\overline{u_1}$	u_3	$\overline{u_3}$	u_4	$\overline{u_4}$	C_6^1	C_6^2	C_6^3	C_6^4	C_6^5	C_6^6
52_{54}	54_{51}					51_{71}	55_{71}				
		52_{52}	54_{55}					51_{73}	55_{73}		
				52_{56}	54_{53}					51_{74}	55_{74}
	57_{51}	L_{52}									
	57_{52}	53_{55}									
	57_{53}	53_{58}									
		58_{51}			53_{56}						
		58_{52}			L_{53}						
		58_{53}			53_{59}						
	53_{54}				59_{51}						
	53_{57}				59_{52}						
	L_{51}				59_{53}						

Table 6: The block corresponding to $C_6 = (\overline{u_1} \vee u_3 \vee \overline{u_4})$

Each block consists of 12 rows and $12m+2v+1$ columns. Only 30 entries are filled above. Each of the remaining $12(12m+2v+1) - 30$ entries is assigned a card with suit $10j$, where the rank is some number larger than $10m$. Since L is a sufficiently large number, we can ensure that each card is unique. Finally, we'll ensure one of these cards is L_{10j}.

By this construction, we see that all cards with suits from $10j-9$ to $10j$ belong to block j. Furthermore, all cards with ranks from $10j-9$ to $10j-1$ belong to block j. Thus, if all m blocks are concatenated together to form our $N \times N$ deal BoaF(S), then we must ensure that each of these cards is stacked up (i.e., eliminated) strictly *within* each block.

For example, in Table 6, there are 21 cards of the form r_s, where $51 \le r, s \le 59$. If our BoaF(S) deal is solvable, then we must find a way to match up these 21 cards within this block so that our deal can become a single stack.

The bottom nine rows in a block cannot be fully cleared by itself, since the six cards with rank $10j-7$ can, at best, form three stacks of two cards. In order for us to combine these cards into a single stack, we require the help of a card appearing in the first three rows, whose column corresponds to a literal in C_j. This card serves as an *activator* to stack up the 18 cards in the bottom nine rows.

In Table 6, the only three possible activator cards are $\{54_{51}, 52_{52}, 54_{53}\}$, namely the cards with suits $10j-9$, $10j-8$, and $10j-7$. Any of these activators enables us to move the cards in the bottom nine rows to a single stack.

Lemma 1 *Let S be an instance of 3-SAT with m clauses, and consider the $12 \times (12m+2v+1)$ block corresponding to the j^{th} clause of S, whose literals have indices p, q, r. There exists a sequence of moves that leave just four top cards, with one card having rank L in the activator column, and the other three having suits $10m+p$, $10m+q$, $10m+r$, all in columns whose literals are not the activator.*

For readability, we demonstrate the proof through our example in Table 6 with $j = 6$ and $m = 7$, from which it will be clear that this sequence of moves exists for all cases.

In Table 6, if 54_{51} is the activator, then the following sequence of moves stacks up the cards in the bottom nine rows: $57_{51} \to 57_{52}$, $57_{51} \to 57_{53}$, $53_{54} \to 53_{57}$, $53_{55} \to 53_{58}$, $58_{52} \to L_{52}$, $58_{52} \to 58_{53}$, $58_{51} \to 58_{52}$, $53_{56} \to 53_{59}$, $59_{53} \to L_{53}$, $59_{53} \to 59_{52}$, $59_{51} \to 59_{53}$, $54_{51} \to 57_{51}$, $54_{51} \to 53_{55}$, $54_{51} \to 58_{51}$, $54_{51} \to 53_{56}$, $54_{51} \to 59_{51}$, $54_{51} \to 53_{54}$, $L_{51} \to 54_{51}$. This leaves Table 7, where the final card L_{51} appears in the $\overline{u_1}$ column, the same column as our eliminated activator.

u_1	$\overline{u_1}$	u_3	$\overline{u_3}$	u_4	$\overline{u_4}$	C_6^1	C_6^2	C_6^3	C_6^4	C_6^5	C_6^6
52_{54}						51_{71}	55_{71}				
		52_{52}	54_{55}					51_{73}	55_{73}		
				52_{56}	54_{53}					51_{74}	55_{74}
	L_{51}										

Table 7: Partially clearing the block corresponding to C_6

By symmetry, the other two activators create the same result: if 52_{52} is our activator, there exists a sequence of moves that makes L_{52} the final card in the u_3 column, and if 54_{53} is our activator, there exists a sequence of moves that makes L_{53} the final card in the $\overline{u_4}$ column.

We note that more than one card can be the activator, but only one is necessary to combine the bottom 18 cards in each block to leave one stack whose top card has rank L.

There are still 5 cards of the form r_s, where $10j-9 \le r, s \le 10j-1$. To ensure that our BoaF(S) deal is solvable, we must stack up these cards as well. To do that, we use the six cards whose suits are larger than $10m$. Since the C_j^k columns are only used in the j^{th} block, we must move these six cards over to the columns corresponding to our literals.

In Table 7 above, we must leave a card with suit 71 in the u_1 column, which is the column *opposite* our activator. We can do this via the sequence $51_{71} \to 55_{71}$, $51_{71} \to 52_{54}$.

Since 52_{52} was not the activator, we can leave a card with suit 73 in either the u_3 or the $\overline{u_3}$ column. We apply the moves $51_{73} \to 52_{52}$ and $55_{73} \to 54_{55}$, and then decide how we wish to combine these two stacks into one. The same is true for the other non-activator, where we can leave a card with suit 74 in either the u_4 or the $\overline{u_4}$ column.

Table 8 gives one possible result of this process, the first three rows of the cleared block of $C_6 = (\overline{u_1} \vee u_3 \vee \overline{u_4})$.

u_1	$\overline{u_1}$	u_3	$\overline{u_3}$	u_4	$\overline{u_4}$	C_6^1	C_6^2	C_6^3	C_6^4	C_6^5	C_6^6
51_{71}											
		55_{73}									
					51_{74}						

Table 8: First three rows of the cleared block of C_6

Finally, there are $12(12m + 2v + 1) - 30$ cards in this block that have not been considered in our analysis. Each of these cards has suit $10j$, with one card being L_{10j}. We use L_{10j} to stack up all of the cards with suit $10j$, ending in the same row or column as the L card in our activator column (L_{51} in our example), so that the final card is L_{10j}. ■

In our earlier example with $S = C_1 \wedge C_2 \wedge C_3 \wedge C_4 \wedge C_5 \wedge C_6 \wedge C_7$, we saw that $(u_1, u_2, u_3, u_4) = (T, T, T, F)$ is a satisfying truth assignment, so that at least one literal in each clause evaluates to TRUE. We pick any such literal to be the column of the activator card, for each of our 7 blocks.

For example, in the clause $C_4 = (u_1 \vee \overline{u_2} \vee \overline{u_4})$, either the u_1 column or the $\overline{u_4}$ column can hold the activator. By our earlier analysis, this forces a card with suit 71 into the $\overline{u_1}$ column or a card with suit 74 into the u_4 column.

By Lemma 1, we can leave just four cards in each of the seven blocks of BoaF(S), where the seven rank L cards are in our activator columns $u_1, u_2, u_3, \overline{u_4}$, and the twenty-one cards with suits $71, 72, 73, 74$ are in our non-activator columns $\overline{u_1}, \overline{u_2}, \overline{u_3}, u_4$.

In general, the satisfying truth assignment of S corresponds exactly to our v activator columns, and the v literals that evaluate to FALSE are the v columns containing the $3m$ cards whose suits range from $10m + 1$ to $10m + v$.

Now consider the Boolean formula $S^* = S \wedge C_8$, where $C_8 = (\overline{u_1} \vee \overline{u_2} \vee \overline{u_3})$. If we were to construct BoaF(S^*), then we must have one of these three FALSE literals being the activator for block 8, forcing some suit $10m + k$ (with $m = 7$ and $1 \leq k \leq 3$) to appear in *both* columns u_k and $\overline{u_k}$, in two different rows. Note that this "duplication" result *must* occur whenever the Boolean formula is not satisfiable.

More formally, we have established the following result.

Lemma 2 *Let S be an instance of 3-SAT with m clauses, and consider the first $12m$ rows in our deal BoaF(S). S is satisfiable if and only if, for all $1 \leq k \leq v$, there exists a sequence of moves that leave all the cards with suit $10m + k$ in either the u_k column or the $\overline{u_k}$ column, but not both.*

Given S, we have described how to concatenate the m blocks corresponding to each clause, to form a BoaF deal with $12m$ rows and $12m + 2v + 1$ columns. Table 9 explains how to construct the final $2v + 1$ rows of BoaF(S), where the final column f is a previously unused column.

Let $\{a_1, \ldots, a_v, b_1, \ldots, b_v, c_1, \ldots, c_v, d_1, \ldots, d_v\}$ be a set of distinct suits, so that none of these suits appears on any other card in BoaF(S). Let X be a rank with $10m < X < L$, which does not form a match with any other card in BoaF(S). Since L is a sufficiently large number, the rank X, as well as the $4v$ distinct suits, are guaranteed to exist.

u_1	$\overline{u_1}$	u_2	$\overline{u_2}$	$\ldots$	$\ldots$	u_v	$\overline{u_v}$	f
X_{c_1}	X_{d_1}							X_{10m+1}
X_{b_1}	X_{a_1}							
		X_{c_2}	X_{d_2}					X_{10m+2}
		X_{b_2}	X_{a_2}					
				$\ldots$	$\ldots$			
				$\ldots$	$\ldots$			
						X_{c_v}	X_{d_v}	X_{10m+v}
						X_{b_v}	X_{a_v}	
L_{a_1}	L_{b_1}	L_{a_2}	L_{b_2}	$\ldots$	$\ldots$	L_{a_v}	L_{b_v}	

Table 9: The final $2v + 1$ rows of our BoaF(S) deal

The blank entries in Table 9 are filled with an arbitrary card with rank L, as are the entries in the $(12m + 2v + 1) - (2v + 1) = 12m$ columns that do not appear in this table.

We are now ready to prove the main theorem in this paper.

Theorem 1 *BoaF is NP-complete.*

Proof Let S be an instance of 3-SAT with m clauses, and consider the $(12m+2v+1) \times (12m+2v+1)$ deal BoaF(S). By Lemma 2, if S is satisfiable, then for each $1 \leq k \leq v$, there exists a sequence of moves so that all the cards with suit $10m + k$ appears in either the u_k or the $\overline{u_k}$ column.

If these cards are in the u_k column, the move sequence $X_{d_k} \rightarrow X_{c_k}$, $X_{10m+k} \rightarrow X_{d_k}$ enables X_{10m+k} to stack up all these cards with suit $10m + k$. Then the sequence $X_{10m+k} \rightarrow X_{b_k}$, $X_{a_k} \rightarrow X_{10m+k}$, $L_{a_k} \rightarrow X_{a_k}$ eliminates all the cards from columns u_k and $\overline{u_k}$ except for some cards all with rank L. By symmetry, we get the same result if all the cards with suit $10m + k$ appear in the $\overline{u_k}$ column.

Therefore, if S is satisfiable, then the deal BoaF(S) has the property that there exists a sequence of moves that eliminates all of the cards, except for some cards whose rank is L. We can then move all these rank L cards to the bottom $2v + 1$ rows of our deal, stacking these cards on top of our "blank" entries, whose cards all have rank L. And then we can take any one rank L card to match up all of the other cards, leaving a single stack with $(12m + 2v + 1)^2$ cards.

If S is not satisfiable, then we claim that BoaF(S) is not solvable. To see why, recall that we must have an activator in each of the m blocks in order for us to eliminate the cards within these blocks. Because S is not satisfiable, there must exist some index k, with $1 \leq k \leq v$, for which card P_{10m+k} exists in column u_k and another card Q_{10m+k} exists in column $\overline{u_k}$, in a different row.

Suppose there exists a sequence of moves that stacks up both P_{10m+k} and Q_{10m+k}. Then it is straightforward to see that any such sequence, which *must* involve X_{10m+k} moving to both columns u_k and $\overline{u_k}$, leaves a card with rank X in one of these columns that cannot be matched to any other card. Thus, BoaF(S) cannot be reduced to a single stack.

We have therefore shown that S is satisfiable if and only if BoaF(S) is solvable. This establishes the desired reduction from 3-SAT, proving the NP-hardness of BoaF.

Finally, we note that BoaF is clearly in NP, since we can verify the solution to any solvable deal in polynomial time, specifically in $(12m+2v+1)^2 - 1 \leq (18m+1)^2 - 1$ steps. Therefore, we conclude that BoaF is NP-complete. ■

Conditions for Unsolvability

Having proven that the $N \times N$ BoaF game is NP-complete, we strongly suspect that there is no simple characterization or classification of all solvable $N \times N$ deals. Nonetheless, we can develop several sufficient conditions to verify that a given $N \times N$ deal is unsolvable, without having to resort to an exhaustive search. Each of these graph-theoretic criteria can be checked in polynomial time.

Given a deal, we construct the *match-by-card graph* G, in which the vertex set $V(G)$ is the set of N^2 cards, and an edge connects two vertices if and only if the corresponding cards match by card (but not necessarily by position). This graph G is similar, but not identical to, the dependency graph used to analyze FreeCell (Paul and Helmert 2016).

The BoaF testbed consists of one million random 4×4 deals, representing seeds 0 to 999,999 of the open-source FreeCell shuffler (Mol 2018).

Table 10 provides one of those deals, which we will show is unsolvable. The corresponding graph G is presented below, in Figure 1.

2H	3D	KD	3H
4D	AH	TS	6D
3C	4H	KC	9S
KH	AC	6C	2C

Table 10: Deal #1264, an unsolvable deal

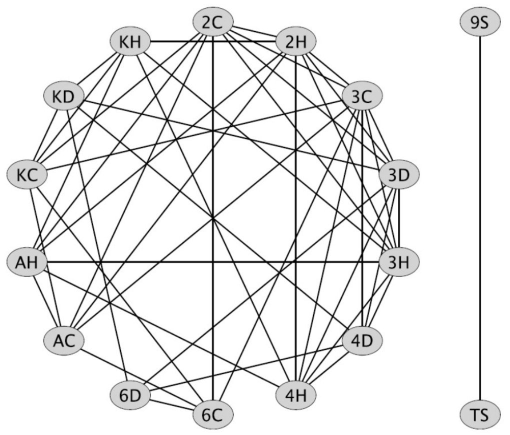

Figure 1: The match-by-card graph of Deal #1264

For a given $N \times N$ deal, the graph G can be determined in $O(N^4)$ time, since there are $\binom{N^2}{2} = \frac{N^2(N^2-1)}{2}$ pairs of vertices, and we can check in constant time whether each pair of cards shares a suit, rank, or are of adjacent rank.

In constructing our graph G, we create an $N^2 \times N^2$ adjacency matrix M that enables us to keep track of $E(G)$, the edge set of G, where $M_{i,j} = 1$ if and only if vertices i and j are connected by an edge. The sum of the v^{th} row of M gives us $\deg(v)$, the degree of each vertex $v \in V(G)$.

We now present our four unsolvability conditions, starting with one that has been previously published.

Odd Bird condition: (Neller 2018) A deal is unsolvable if the graph G has some vertex v with degree 0.

Let v be a vertex with degree 0. Then the card corresponding to this vertex cannot be placed on top of another stack, nor can any card be stacked on top of this card. As a result, it is impossible to find a sequence of moves that leaves a single stack of N^2 cards.

Clearly, we can test this condition in polynomial time, since we have the values of $\deg(v)$ from our adjacency matrix M. Out of the million deals from our testbed, 1484 are unsolvable due to the Odd Bird condition.

Multiple Flocks condition: A deal is unsolvable if the graph G is disconnected.

This is a generalization of the first condition. If G is disconnected, then there exists a set of vertices U, with $0 < |U| < N^2$, for which no edge in $E(G)$ connects a vertex in U to a vertex in $V(G) - U$. In other words, the stack(s) formed by the cards of U cannot be matched to any of the remaining cards in the deal, which implies that it is impossible to end the game with a single stack of N^2 cards.

As an example, the graph in Figure 1 satisfies the Multiple Flocks condition, since the set $U = \{$9S, TS$\}$ is disconnected from the rest of the graph, forcing any move sequence to conclude with a minimum of two stacks.

We can test this condition in polynomial time as follows: start with any vertex v and use the adjacency matrix M to generate the breadth-first search tree with root v, one level at a time. When we cannot add any more vertices, we simply count the number of vertices in the breadth-first search tree. G is connected if and only if this number is N^2.

Out of the million deals from our testbed, 287 are unsolvable due to the Multiple Flocks condition, with $|U| \geq 2$. (The case $|U| = 1$ is equivalent to the Odd Bird condition.)

These two conditions are sufficient for unsolvability, but they are not necessary conditions. There could be deals where G is connected, but certain moves that must be made will cause G to become disconnected. This often occurs when there is a *cut edge*, i.e., an edge uw for which G is connected but the removal of edge uw leaves G disconnected. This observation inspires the following.

Cut Edge condition: A deal is unsolvable if all of the following hold: G is a connected graph for which there exists a cut edge uw, the 2-edge connected component U that contains u is not collapsible, and there is no position on the board for w (excluding its original position) where the reduced board consisting of the cards $U \cup w$ is solvable.

If G is a connected graph having a cut edge $uw \in E(G)$, then the vertex set $V(G)$ can be partitioned into two sets U and W, with $U \cap W = \emptyset$, $u \in U$, and $w \in W$.

If the deal is solvable, then any move sequence that reduces the N^2 cards to a single stack must contain either $u \to w$ or $w \to u$. However, we cannot make either move if this results in a disconnected graph G.

Since uw is a cut edge, either the cards of U must be collapsible to a single stack with u as the top card, or the cards in W must be collapsible to a single stack with w as the top card.

Without loss of generality, assume that $|U| \leq |W|$. We check to see if the cards in U can be moved to a single stack with u on top. If no such sequence exists, then solvability implies that the cards in W can be moved to a single stack with top card w, where the position of this single stack is different from the original location of card w.

So given a deal of N^2 cards, we just need to consider the $N^2 - |U| - 1$ positions where this card w can end up, and consider $U \cup w$, the subset of $|U| + 1$ cards. If none of these $N^2 - |U| - 1$ mini-deals can result in a single stack, our $N \times N$ deal must be unsolvable.

This technique is extremely powerful and fast when $|U|$ is small. To illustrate the Cut Edge condition, consider the following deal from our testbed, whose match-by-card graph is presented in Figure 2.

QD	4C	4H	JS
TS	9D	3C	2C
QH	KS	9S	JD
AC	6D	QS	7D

Table 11: Deal #221,602, an unsolvable deal

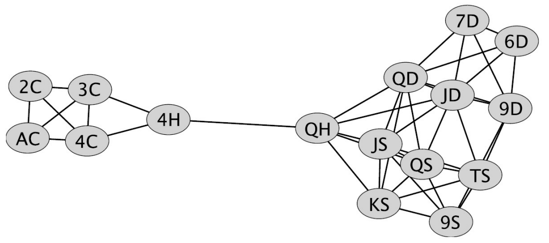

Figure 2: The match-by-card graph of the deal in Table 11.

We see that uw is a cut edge of G, where $u = 4$H and $w =$ QH. We readily see that the five-card mini-deal with $U = \{$AC, 2C, 3C, 4C, 4H$\}$ cannot be reduced to a single stack with 4H on top. Thus, we consider the 10 possible positions of the card QH in the table below (indicated by *), and consider whether the six-card mini-deal $U \cup w$ is solvable in any of these ten cases.

*	4C	4H	*
*	*	3C	2C
	*	*	*
AC	*	*	*

Table 12: Reducing the analysis to ten easy subproblems

We can quickly verify that none of these ten cases results in a single stack, which implies that this deal is unsolvable.

Each of the criteria in the Cut Edge condition can be verified in polynomial time, provided the cardinality of U is bounded by a constant. We have coded an algorithm that considers all cases where G is a connected graph with a cut edge uw, for which $|U| \leq 6$.

Out of the million deals from our testbed, 49 are unsolvable due to the Cut Edge condition.

Lollipop Stick condition: A deal is unsolvable if the graph G contains three vertices $\{x, u, w\}$ that form a "lollipop stick," having a specific property described below.

Consider the following deal from our testbed.

5S	7D	QS	2C
4C	AD	8D	5C
QH	7C	TC	3C
8C	2D	9C	3D

Table 13: Deal #360,528, an unsolvable deal

The corresponding graph is presented in Figure 3. We see that G is in the shape of a lollipop, with the cards 5S, QS, QH forming the lollipop stick.

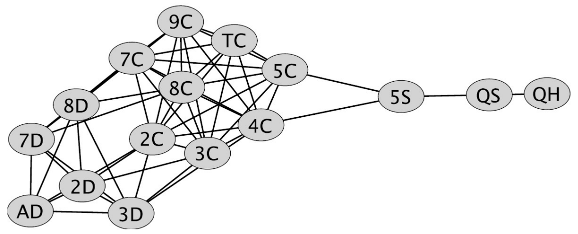

Figure 3: The match-by-card graph of the deal in Table 13.

Let $x =$ QH, $u =$ QS, $w =$ 5S. Since uw is a cut edge, we let $U = \{x, u\}$ and $W = V(G) - U$. As x and u do not match by position, U is not collapsible. Consequently, the deal is only solvable if W can be collapsed into a single stack with w as the top card. Since x and w do not match by card, the final two moves must be $u \rightarrow w$, followed by either $x \rightarrow u$ or $u \rightarrow x$.

These moves are only possible if the cards in W can be collapsed, where the top card w ends up in a location that matches by position with both x and u. In this deal, w must move from the top-left corner to the cell in Row 3, Column 3, while only moving on top of cards in W. This requires w to move at least three times. However, w matches with only two cards in W, which implies that this deal is unsolvable.

In general, a deal is unsolvable if all of the following criteria are met:

1. Three cards, x, u, w, form a lollipop stick with $\deg(x) = 1$ and $\deg(u) = 2$, with both xu and uw being cut edges.
2. x and u do not match by position.
3. w matches by card with fewer cards in $W = V(G) - \{x, u\}$ than the number of moves it must make such that it both matches by position with x and u, and is the top card of a single stack of all the cards in W.

The Lollipop Stick condition can be verified in polynomial time, since we can determine in $O(N^6)$ steps all the ways that three vertices $\{x, u, w\}$ from $V(G)$ can form a lollipop stick, from which the remaining two criteria can be checked in either constant or linear time.

Out of the million deals from our testbed, 8 are unsolvable due to the Lollipop Stick condition.

Solving with Search

In the previous section, we presented four unsolvability conditions that proved that $1484 + 287 + 49 + 8 = 1828$ of the million deals from the 4×4 BoaF testbed are unsolvable. In this section, we efficiently find the solutions to 998,120 deals, and then show that the remaining 52 deals are unsolvable.

We use two different search algorithms to find solutions to the deals in our testbed: a heuristic depth-first search and a standard depth-first search. There is a tradeoff between computation time per step and number of steps: the heuristic algorithm takes fewer steps to find a solution, but each step takes longer to run. For this reason, we present both search algorithms.

To do this, we introduce the *match-by-position* graph, defined similarly to the match-by-card graph, except that two vertices are connected by an edge if and only if their corresponding cards match by position.

Heuristic depth-first search is a search algorithm that combines elements of depth-first and best-first search (Poole and Mackworth 2017). The initial search node is the initial state of the deal; this node starts in an open list. Next, the board state that results after each legal move from the initial state is added to the open list, and the node that was evaluated is added to a closed set; this expansion is one *step*. The open list is a priority queue ordered by a heuristic function of the board state $h(x)$. As the name implies, search is depth-first, so only the leaf nodes with greatest depth are possible candidates for expansion.

We use the heuristic function

$$h(x) = a(x) + b(x) + c(x) - p_1(x) - p_2(x)$$

where $a(x)$, $b(x)$, and $c(x)$ are the heuristic components, and $p_1(x)$ and $p_2(x)$ are the penalty functions for the board state x being unsolvable. The heuristic components are defined as follows:

- $a(x)$ is the number of pairs that match by card;
- $b(x)$ is the number of pairs that match by position;
- $c(x)$ is the number of legal moves.

The two penalty functions are

$$p_1(x) = \begin{cases} 1000 & \text{if there are multiple flocks} \\ 0 & \text{otherwise} \end{cases}$$

$$p_2(x) = \begin{cases} 1000 & \begin{array}{l}\text{if the match-by-position} \\ \text{graph is disconnected}\end{array} \\ 0 & \text{otherwise} \end{cases}$$

If either of the penalty functions is triggered, the board must be unsolvable, so search nodes x with $h(x) < 0$ are never expanded.

Our standard depth-first search algorithm is almost exactly the same as classical depth-first search. The only difference is that search nodes are only expanded if they are not provably unsolvable using the penalty functions; i.e., if $p_1(x) + p_2(x) = 0$.

The two search algorithms were implemented in Java, on top of an existing codebase (Neller 2018). We tested both of the search algorithms on a personal computer with a 1.4GHz Intel Core i5 processor and 4GB RAM.

First, we tested the heuristic depth-first search algorithm on all of the 998,172 deals that were not proven to be unsolvable. Due to the long runtime per step of the heuristic search algorithm, we gave up if it did not find a solution in 100 steps. Next, we tested the standard depth-first search on all of the remaining undetermined deals. It gave up after 5.5 million steps due to memory constraints.

The heuristic algorithm solved 57% of deals (569,449) in 15 steps, the minimum possible. It solved 90% of deals (898,503) in fewer than 100 steps.

The standard depth-first search algorithm was tested on the remaining 99,669 deals. It solved 99,617 deals, in a median of 4,055 steps. It proved 50 of the remaining 52 deals unsolvable via exhaustive search. The authors have manually shown that the remaining two deals (#618,979 and #687,168) are unsolvable using a graph-theoretic analysis.

Conclusion

In this paper, we have shown that the $N \times N$ Birds of a Feather game is NP-complete, demonstrating the computational intractability of this recently invented puzzle. We also developed several graph-theoretic tests for unsolvability, as well as two depth-first search algorithms, and applied them to analyze the million deals in the 4×4 BoaF testbed. We successfully found solutions to every solvable deal, and demonstrated that the remaining 1880 deals were indeed unsolvable.

Our general approach may be useful for combinatorial games beyond Birds of a Feather. The combined approach of quickly ruling out most unsolvable options, using a heuristic search to find solutions to "easy" problems quickly, and then using an uninformed search to either find solutions to the "hard" problems or exhaustively prove them unsolvable might be a good approach to other NP-complete classical planning problems.

There are several directions for future work. There are undoubtedly other graph-theoretic unsolvability conditions that can be checked in polynomial time, especially tests that also include the match-by-position graph in their analysis. In addition, whenever there is a cut vertex in the match-by-card graph of a deal, a more restricted search could be performed even if it is not possible to directly confirm the deal's unsolvability. Of course, more deals could be tested to see if the performance of these algorithms generalizes. Finally, these algorithms could be tested on larger boards beyond the case $N = 4$, as well as $N \times M$ boards with $N \neq M$.

References

Demaine, E. 2018. Erik Demaine's Combinatorial Games Page. http://erikdemaine.org/games.

Garey, M., and Johnson, D. 1979. *Computers and Intractability: A guide to the theory of NP-completeness*. New York: W.H. Freeman.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

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

Kendall, G.; Parkes, A.; and Spoerer, K. 2008. A survey of NP-complete puzzles. *International Computer Games Association Journal* 31:13–34.

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.

Paul, G., and Helmert, M. 2016. Optimal solitaire game solutions using A* search and deadlock analysis. In *Proceedings of the 9th Annual Symposium on Combinatorial Search (SoCS 2016)*, 135–136.

Poole, D. L., and Mackworth, A. K. 2017. *Artificial intelligence: foundations of computational agents*. Cambridge University Press.

Yannakakis, G., and Togelius, J. 2018. *Artificial Intelligence and Games*. Springer. http://gameaibook.org.