Theoretical Computer Science 410 (2009) 5252–5260

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

The complexity of Solitaire

Luc Longpré [a], Pierre McKenzie [b,*]

[a] *Computer Science, University of Texas at El Paso, United States*
[b] *DIRO, Université de Montréal, Canada*

ARTICLE INFO

Keywords:
Computational complexity
Completeness
Games
Solitaire

ABSTRACT

Klondike is the well-known 52-card Solitaire game available on almost every computer. The problem of determining whether an n-card Klondike initial configuration can lead to a win is shown NP-complete. The problem remains NP-complete when only three suits are allowed instead of the usual four. When only two suits of opposite color are available, the problem is shown NL-hard. When the only two suits have the same color, two restrictions are shown in AC^0 and in NL respectively. When a single suit is allowed, the problem drops in complexity down to $AC^0[3]$, that is, the problem is solvable by a family of constant-depth unbounded-fan-in { AND, OR, mod_3 } -circuits. Other cases are studied: for example, "no King" variant with an arbitrary number of suits of the same color and with an empty "pile" is NL-complete.

1. Introduction

Solitaire card games, called patience games outside of the United States, apparently originate from the fortune-telling circles of the eighteenth century [9]. Of the many hundred different solitaire card games in existence [8], to the best of our knowledge, only *FreeCell* [4] and *BlackHole* [5] have been studied from a complexity viewpoint. In both cases, determining whether an initial configuration can lead to a win was shown NP-complete.

Over the last two decades, a particular variation of solitaire, the Klondike version, was popularized by Microsoft Windows (Fig. 1). Earlier, Parlett [8] had described Klondike as the "most popular of all perennial favorites in the realm of patience, which is surprising since it offers the lowest success rate of any patience".

In a paper entitled *Solitaire: Man Versus Machine*, Yan, Diaconis, Rusmevichientong and Van Roy [10] report that a human expert (and distinguished combinatorialist, former president of the *American Mathematical Society*!) patiently recorded data on his playing 2000 games of *thoughtful solitaire*, that is, the intellectually more challenging Klondike in which the complete initial game configuration is revealed to the player at the start of the game. The expert averaged 20 min per game and was able to win the game 36.6% of the time. Pointing to the prohibitive difficulty of obtaining nontrivial mathematical estimates on the odds of winning this common game, Yan et al. then describe heuristics developed using the so-called rollout method, and they report a 70% win rate.

Why is it so hard to compute the odds of winning at Klondike? In part, this question prompted our investigation of the complexity of the game. As well, after using the Minesweeper game [6] for several years as a motivating example for students, we looked for different NP-complete examples based on other widely popular and deceptively simple games.

The precise rules of Klondike are described in Section 2. To make the game amenable to a computational complexity analysis, the game is generalized to allow instances of arbitrary finite size. Hence an instance of the game involves a "deck" containing the "cards"

* Corresponding author.
E-mail addresses: longpre@utep.edu (L. Longpré), mckenzie@iro.umontreal.ca (P. McKenzie).

0304-3975/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2009.08.027

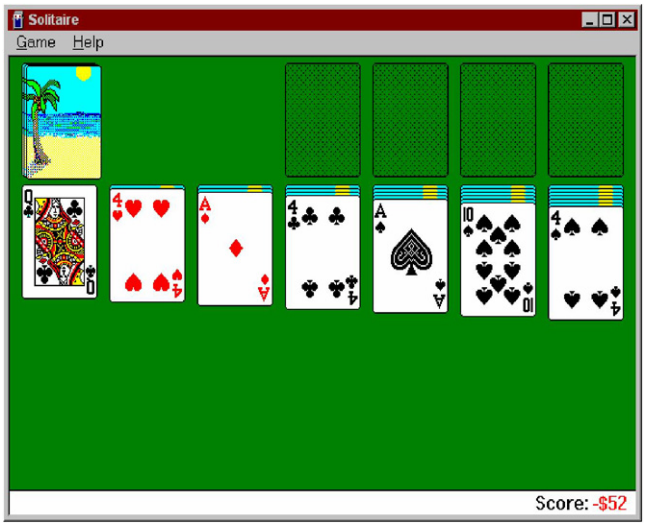

Fig. 1. Initial Klondike configuration on Microsoft Windows. In the terminology of [10], there are four empty *suit stacks*, seven *build stacks* containing 1, 2, 3, 4, 5, 6 and 7 cards respectively with only the top card facing up, and one *pile* containing the remaining cards facing down; another stack, the *talon*, will appear in the course of the game when cards from the pile are moved to the talon three by three.

$1\clubsuit, 2\clubsuit, 3\clubsuit, \ldots, n-1\clubsuit, n\clubsuit$
$1\diamondsuit, 2\diamondsuit, 3\diamondsuit, \ldots, n-1\diamondsuit, n\diamondsuit$
$1\heartsuit, 2\heartsuit, 3\heartsuit, \ldots, n-1\heartsuit, n\heartsuit$
$1\spadesuit, 2\spadesuit, 3\spadesuit, \ldots, n-1\spadesuit, n\spadesuit$

and the game configurations are generalized appropriately. The problem of interest is to determine, given an initial game configuration, whether the game can be won. We show here that this is NP-complete.

Our NP-completeness proof bears resemblance to the proof that FreeCell is NP-complete [4]. In particular, the method of nondeterministically assigning truth values to card configurations is the same. However, our strategy gets by with only three card suits as opposed to the usual four used in the current proof that FreeCell is NP-complete. Many differences between FreeCell and Klondike further arise, for instance, when we argue the NP upper bound in the presence of "backward" moves (i.e., from a suit stack to a build stack) and when we consider restricted variants of the game. We highlight the following as our main results:

1. Klondike is NP-complete and remains so with only three suits available,
2. Klondike with a black suit and a red suit is NL-hard,
3. Klondike with any fixed number b of black suits is in NL,
4. flat Klondike (that is, without a pile) with an input-dependent number of black suits and without generalized Kings (see below) is NL-complete,
5. Klondike with a single suit is in $AC^0[3]$,
6. flat Klondike with 2 black suits and without generalized Kings is in AC^0.

Section 2 contains preliminaries and a precise description of the Klondike variation studied in this paper. Section 3 proves that Klondike is NP-complete and Section 4 considers restricting the usual four-suit game. Section 5 concludes with a discussion and some open questions.

2. Preliminaries

2.1. Complexity theory

We assume familiarity with basic complexity theory, such as can be found in standard books on the subject, for example [7]. We recall the inclusion chain

$$AC^0 \subset AC^0[3] \subset AC^0[6] \subseteq L \subseteq NL = \text{co-NL} \subseteq P \subseteq NP.$$

Here, the complexity class AC^0 is the set of languages accepted by DLOGTIME-uniform unbounded-fan-in constant-depth $\{\wedge, \vee, \neg\}$-circuits of polynomial size. The larger class $AC^0[m]$ is the set of languages AC^0-Turing reducible to the MOD_m Boolean function, defined to output 1 iff m does not divide the sum of its Boolean inputs. The classes L and NL stand for deterministic and nondeterministic logarithmic space respectively. The classes P and NP are deterministic and

nondeterministic polynomial time respectively. We adopt the definitions of [3] for constant-depth circuit uniformity (see also [2]).

If not otherwise stated, the hardness results in this paper are in terms of many-one logspace reducibility.

2.2. Klondike

We allow ourselves to borrow the following excellent description of Klondike found in [10, Section 2]:

> The goal of the game is to move all cards into the suit stacks, aces first, then two's, and so on, with each suit stack evolving as an ordered increasing arrangement of cards of the same suit. On each turn, the player can move cards from one stack to another in the following manner:
>
> 1. Face-up cards of a build stack, called a card block, can be moved to the top of another build stack provided that the build stack to which the block is being moved accepts the block (see Points 6 and 7 below for the meaning of acceptance). Note that *all* face-up cards on the source stack must be moved together. After the move, these cards would then become the top cards of the stack to which they are moved, and their ordering is preserved. The card originally immediately beneath the card block, now the top card in its stack, is turned face-up. In the event that all cards in the source stack are moved, the player has an empty stack.
> 2. The top face-up card of a build stack can be moved to the top of a suit stack, provided that the suit stack accepts the card.
> 3. The top card of a suit stack can be moved to the top of a build stack, provided that the build stack accepts the card.
> 4. If the pile is not empty, a move can deal its top three cards to the talon, which maintains its cards in a first-in-last-out order. If the pile becomes empty, the player can redeal all the cards on the talon back to the pile in one card move. A redeal preserves the ordering of cards. The game allows an unlimited number of redeals.
> 5. A card on the top of the talon can be moved to the top of a build stack or a suit stack, provided that the stack to which the card is being moved accepts the card.
> 6. A build stack can only accept an incoming card block if the top card on the build stack is adjacent to and braided with the bottom card of the block. A card is adjacent to another card of rank r if it is of rank $r + 1$. A card is braided with a card of suit s if its suit is of a color different from s. Additionally, if a build stack is empty, it can only accept a card block whose bottom card is a King.
> 7. A suit stack can only accept an incoming card of its corresponding suit. If a suit stack is empty, it can only accept an Ace. If it is not empty, the incoming card must be adjacent to the current top card of the suit stack.

Yan et al. coin the name *thoughtful solitaire* for the Klondike variation in which the player sees the complete game configuration, including the ranks of all the cards facing down, throughout the course of the game. The Klondike rules are otherwise unchanged.

Since we generalize Klondike to involve a variable number of cards, we adjust the notion of a game configuration to allow an arbitrary initial number of build stacks of arbitrary size. No new build stacks can be created in the course of the game however. We do not insist that the initial build stacks contain $k, k-1, \ldots, 2, 1$ cards respectively, but this could be arranged in most cases with no loss of generality by shifting all the card values upward and then inflating the initial build stacks with low value cards that can be released right away. Note that because the creation of new build stacks that may become empty quickly would allow the movement of generalized Kings, this method does not provide a general transformation from an instance with an arbitrary set of stacks. However, since none of our lower bound proofs use properties of generalized Kings, they all hold with this additional restriction. We will occasionally consider Klondike with a number of suits other than 4. In that case, the number of suit stacks is adjusted accordingly.

Card numbers and suit numbers are represented in binary notation. We assume any reasonable encoding of cards and game configurations that allows extracting individual card information in AC^0. In particular, the pile, talon and stacks are represented in table form so that the predicate "c is the ith card in the table" is AC^0-computable.

Definition 1. Problem SOLIT(b, r):
Given: an initial b-black-suit and r-red-suit Klondike configuration involving the same number n of cards in every suit.
Determine: Whether the $(b + r)n$ cards can be placed on the $b + r$ suit stacks by applying the Klondike game rules starting from the given initial configuration.

In Section 3 we will be studying SOLITAIRE, by which we mean SOLIT$(2, 2)$. In Section 4, we will consider Klondike restrictions, such as FLAT-SOLIT(b, r), by which we mean SOLIT(b, r) with an initial configuration having an empty pile and empty talon. We define the further restriction FLAT-SOLIT$_{\text{NoKing}}(b, r)$ to mean FLAT-SOLIT(b, r) played with modified rules that forbid an empty stack from accepting a generalized King (i.e., we disallow refilling an empty build stack; this is equivalent to viewing the highest ranked cards as generalized Queens rather than generalized Kings). Using a "$*$", such as in FLAT-SOLIT$(*, 0)$, means that the number of suits corresponding to the $*$ is not fixed and depends on the input.

3. Klondike is NP-complete

Theorem 2. SOLITAIRE *is* NP*-complete.*

Proof (NP *Upper Bound*). Consider a winning N-card Klondike instance involving k build stacks. We need to argue that the length of a shortest winning sequence of moves is polynomial in N. We define four types of moves:

Type 1: moving a card out of the talon, or any move that causes a face-down card in a build stack to turn face-up. This includes any move from build stack to build stack, or moving the last face-up card from a build stack to a suit stack (except when such a move empties a build stack).
Type 2: moving a card from a build stack to a suit stack (without being of type 1).
Type 3: moving a card from a suit stack to a build stack.
Type 4: moving cards from the pile to the talon.

Let ℓ be a shortest winning sequence of moves. Such a sequence contains $N - k$ moves of type 1 since exactly k cards were visible at the outset. We claim that two successive moves of type 1 in this sequence are separated by $O(N)$ moves of type 2, type 3 or type 4. The NP upper bound follows from the claim since moves of type 1 are irreversible and no obstacle remains after the $N - k$ moves of type 1; thus ℓ is $O(N^2)$.

To see the claim, we show that if there is a winning sequence, then there is one where the moves between two successive moves of type 1 consist of a sequence of at most N moves of type 2, followed by at most N moves of type 3, followed by at most N moves of type 4. First, moves of type 2 and of type 3 do not interfere with moves of type 4, so we can delay all type 4 moves until all type 2 and type 3 moves are finished.

Now, it remains to see that type 3 moves can be assumed to occur last in an optimal sequence of moves of types 2 or 3. This will suffice since there can be no more than N consecutive type 2 moves and no more than N consecutive type 3 moves. So consider a configuration C and an optimal sequence $\mathcal{S}$ of moves of types 2 or 3 leading to a configuration C'. We prove, by induction on the number k of type 3 moves, that postponing the type 3 moves in $\mathcal{S}$ until the end still leads from C to C'. If $k = 0$ then this is vacuously true. So let $k > 0$. Let δ, the first type 3 move in $\mathcal{S}$, remove a card c from a suit stack σ_c. Let C_1 be the configuration reached from C by applying the prefix of $\mathcal{S}$ up to and including δ. Since the remainder of $\mathcal{S}$ leads optimally from C_1 to C' in $k - 1$ type 3 moves, by the induction hypothesis, these $k - 1$ type 3 moves can be postponed until after the type 2 moves. Let $\mathcal{S}'$ be the sequence, leading optimally from C to C', obtained from $\mathcal{S}$ by postponing its last $k - 1$ type 3 moves until the end. Note that no type 2 move involving σ_c can occur after δ in $\mathcal{S}'$ until c is moved back to σ_c. But moving c back to σ_c after δ would merely undo δ, contradicting the optimality of $\mathcal{S}'$. Consequently, because no type 2 move after δ in $\mathcal{S}'$ involves σ_c, δ can also be postponed until after all type 2 moves in $\mathcal{S}'$. This concludes the induction. Hence, between any two successive type 1 moves in a shortest sequence, there can be at most $3N$ moves of types 2, 3 or 4, proving the claim.

NP-*hardness*. We reduce from 3SAT. The main idea is to construct a pair of build stacks for each 3SAT formula variable. The top cards on these stacks will correspond to whether we want the variable to be true or false. Only one of these two top cards can be moved to another build stack. Moving a top card will uncover cards corresponding to the clauses that become true when the variable takes the Boolean value associated with the top card.

Let F be a 3CNF Boolean formula with n variables $v_1, \ldots, v_n$ and m clauses $c_1, \ldots, c_m$. Construct an initial configuration corresponding to this formula so that the configuration is winning if and only if the formula is satisfiable. For each variable v_i, associate cards $2i$ et $2i + 1$. For each clause c_j, associate cards $2n + 7j, \ldots, 2n + 7j + 6$.

For each variable v_i, construct 3 build stacks (called the *variable* stacks):

1. one with card $2i$♠ facing up on top and cards to be determined below,
2. one with card $2i$♣ facing up on top and cards to be determined below,
3. one with the sole card $(2i + 1)$♡.

For each clause $c_j = (l_p, l_q, l_r)$, construct 3 build stacks (called *clause* stacks) with one card facing up on top and one card facing down below:

1. one stack with $(2n + 7j + 6)$♡ on top and $(2n + 7j + 5)$♣ below,
2. one stack with $(2n + 7j + 4)$♡ on top and $(2n + 7j + 3)$♣ below,
3. one stack with $(2n + 7j + 2)$♡ on top and $(2n + 7j + 1)$♣ below.

Also, if $l_p = v_i$, put $(2n + 7j + 1)$♠ facing down in stack $2i$♠, and if $l_p = \overline{v_i}$, put $(2n + 7j + 1)$♠ facing down in stack $2i$♣. If $l_q = v_i$, put $(2n + 7j + 3)$♠ facing down in stack $2i$♠, and if $l_q = \overline{v_i}$, put $(2n + 7j + 3)$♠ facing down in stack $2i$♣. If $l_r = v_i$, put $(2n + 7j + 5)$♠ facing down in stack $2i$♠, and if $l_r = \overline{v_i}$, put $(2n + 7j + 5)$♠ facing down in stack $2i$♣. Arrange for all clause (spade, face-down) cards within any given build stack to occur in order of increasing card rank.

Finally, create the *critical* build stack, facing down, with cards $(2n + 7j)$♡ in any order for $1 \leq j \leq m$, followed by all remaining cards in increasing order, starting with aces, and followed further by 3 generalized Kings $2n + 7m + 7$♡, $2n + 7m + 7$♣ and $2n + 7m + 7$♠. Regrouping all the generalized Kings at the bottom of the critical stack serves to prevent moves to an empty build stack during the core of the simulation. The pile, talon and suit stacks are empty. For each clause j, we will refer to the card $2n + 7j$♡ as to the *critical clause-j card*.

Suppose that some assignment satisfies the formula. Here is how to win the game. In the assignment, if the variable v_i is false, put card $2i$♣ on card $(2i + 1)$♡. If v_i is true, put card $2i$♠ instead. Then, move all the cards that were below the $2i$ cards. This is possible because all these cards are spade cards numbered $2n + 7j + 1$ or $2n + 7j + 3$ or $2n + 7j + 5$, and the red cards $2n + 7j + 2$ and $2n + 7j + 4$ and $2n + 7j + 6$ all sit facing up on top of their build stacks. If the formula is satisfiable, all the clauses have at least one literal set to true, so at least one clause j card will be released in this manner for each j.
Claim: For each j, a sequence of j-clause stack moves now exists such that

1. one clause-j stack can be made to accept the critical clause-j card, and
2. after this sequence, if a situation is reached such that all the cards ranked less than $2n + 7j$ are placed on the suit stacks and a black card ranked $2n+7j$ sits on top of the critical stack, then all the cards ranked $2n+7j$, $2n+7j+1, \ldots, 2n+7j+6$ can be placed on the suit stacks.

This claim implies a win as follows. Part (1) of the claim ensures that all the critical cards can be moved from the critical stack to the clause stacks. This releases the aces and allows moving all the cards ranked less that $2n + 7$, including those that remained on the variable stacks, to the suit stacks. Part (2) of the claim together with an induction on j then yield the winning sequence of moves.

To prove the claim, fix j and let $2n + 7j + k\spadesuit$ for some $k \in \{1, 3, 5\}$ be the smallest clause-j card that got released from a variable build stack. If $k = 1$, then $2n + 7j + k\spadesuit$ was accepted by $2n + 7j + 2\heartsuit$ and the resulting stack accepts the critical clause-j card. If $k = 3$, then $2n + 7j + k\spadesuit$ was accepted by $2n + 7j + 4\heartsuit$, so the $2n + 7j + 2\heartsuit$ card can be displaced, thus uncovering $2n + 7j + 1\clubsuit$ which in turn accepts the critical clause-j card. Finally, if $k = 5$, then $2n + 7j + k\spadesuit$ was accepted by $2n+7j+6\heartsuit$; now $2n+7j+4\heartsuit$ can be displaced, followed by $2n+7j+3\clubsuit$, followed by $2n+7j+2\heartsuit$, again uncovering $2n + 7j + 1\clubsuit$ which accepts the critical clause-j card. This proves part (1) of the claim. To prove part (2) of the claim, it suffices to observe that although some $2n+7j+k\clubsuit$ card(s) remain(s) under some $2n+7j+k+1\heartsuit$ card(s) after the above sequence of moves, the resulting stack configurations do not form an obstacle when the complementary cards in all suits are available in increasing order.

Conversely, suppose that the configuration produced from the formula is winning. Then the initial sequence of a winning sequence of moves must uncover the aces. This initial sequence cannot involve backward moves (i.e., from a suit stack to a build stack). This initial sequence must then first release every critical card $2n + 7j\heartsuit$. Each of these cards must be moved to a black $(2n+7j+1)$ card. For any given j, this cannot happen unless for some i, some clause-j card is released from one (and only one, since a single card $2i + 1$ is visible) of the two v_i-variable stacks. An assignment of variables v_i based on which of the two v_i-variable stacks was first released is, by construction, a satisfying assignment to our formula. □

4. Complexity of Klondike restrictions

The proof of Theorem 2 used only {♣, ♡, ♠}. Furthermore, the initial configuration constructed had an empty pile and empty talon. Thus we have:

Theorem 3. SOLIT(2, 1) *and* FLAT-SOLIT(2, 1) *are* NP*-complete.*

Because the NP upper bound argument from Theorem 2 extends to the case in which an arbitrary number of (red and black) suits is allowed, we also have:

Theorem 4. SOLIT(∗, ∗) *and* FLAT-SOLIT(∗, ∗) *are* NP*-complete.*

Recall the "no King" game restriction, in which empty build stacks can never be filled. Because the Klondike instances constructed in the NP-hardness proof from Theorem 2 neither allow nor tolerate refilling an empty stack (except at the very end when all the cards have been released), we also have:

Theorem 5. FLAT-SOLIT$_{\text{NoKing}}(b, r)$ *is* NP*-complete for any* $b > r \geq 1$.

One might expect the remaining cases, namely the case of one black suit and one red suit, and the case in which all suits are black, to be trivial. This is not quite so. We begin with the latter.

A FLAT-SOLIT(∗, 0) instance w involves a set of nb cards $c_{1,1}, c_{1,2}, \ldots, c_{1,b}, c_{2,1}, \ldots, \ldots, c_{n,b}$ scattered within an arbitrary number of build stacks, where the card $c_{i,s}$ is the suit-s card of rank i. Since only black suits occur in w, the only actions possible are those that move a generalized King and its block to an empty build stack and those that move a card from a build stack to a suit stack. Even when all suits are black, the generalized King moves are powerful because the choice of which black King to move to an empty stack can be critical to the successful completion of the game. We do not yet fully understand the power of such moves. So we turn to FLAT-SOLIT$_{\text{NoKing}}(*, 0)$.

We say that a FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ instance w is *nontrivial* if for every s, the suit-s cards occur in increasing order in every build stack. Clearly, no win is possible from a trivial w. When w is nontrivial, we define the directed graph $H(w)$ on the set of cards of w as follows: for $1 \leq i, j \leq n$ and $1 \leq s, t \leq b$, the arc $(c_{i,s}, c_{j,t})$ exists in $H(w)$ iff

1. $s = t$ and $j = i - 1$ (call this a *horizontal* edge), or
2. $s \neq t$ and the card $c_{i,s}$ is immediately beneath $c_{j,t}$ in some build stack (call this a *vertical* edge).

Proposition 6. *Consider a nontrivial* FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ *instance* w*. A win is possible from* w *iff* $H(w)$ *is cycle-free.*

Proof. Suppose that a cycle exists in $H(w)$. By construction, such a cycle must involve at least one horizontal edge $(c_{i,s}, c_{i-1,s})$. Since $H(w)$ obviously captures implication chains of the form "a card c cannot be placed on a suit stack before all the cards reachable from c in $H(w)$ are themselves placed on a suit stack", this cycle implies that $c_{i,s}$ must be placed on a suit stack before $c_{i-1,s}$. Hence a win is impossible from w.

Conversely, suppose that H is cycle-free. Then there exists an (inverse) topological sort of $H(w)$, that is, an ordering $c^{(1)}, c^{(2)}, \ldots, c^{(bn)}$ of the nodes of $H(w)$ such that no edge $(c^{(k)}, c^{(l)})$ with $k < l$ appears in $H(w)$. We prove by induction on

k that if $c^{(1)}, c^{(2)}, \ldots, c^{(k-1)}$ are on the suit stack, then $c^{(k)}$ can be placed on a suit stack.
Basis: If $k = 1$, we know that no edge out of $c^{(1)}$ appears in $H(w)$. By construction, $c^{(1)}$ must be $c_{1,s}$ for some suit s and must appear on top of a build stack. Hence $c^{(1)}$ can form a (first) suit stack.
Inductive step: Let $k > 1$ and suppose now that $c^{(1)}, c^{(2)}, \ldots, c^{(k-1)}$ are on the suit stacks. If $c^{(k)} = c_{i,s}$ cannot be moved to a suit stack, then either $c_{i,s}$ is immediately beneath some $c_{j,t}$ not yet on the suit stacks, or $c_{i-1,s}$ is not yet on the suit stacks. In both cases, because the cards not yet on a suit stack occur as $c^{(l)}$ for some $l > k$, an edge $(c^{(k)}, c^{(l)})$ with $k < l$ must occur by construction of $H(w)$. But this contradicts the properties of the topological ordering. This completes the induction and proves that all the cards can be moved to the suit stacks. Hence a win is possible from w. □

Theorem 7. FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ *is* NL*-complete.*

Proof. NL *upper bound*. Consider a FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ instance w. We first check in AC^0 that w is nontrivial. If w is trivial then we reject immediately. Otherwise, Proposition 6 implies a co-NL = NL upper bound, because $H(w)$ is easily constructed in log space (in AC^0 in fact), and the total number bn of nodes in $H(w)$ together with the card numbers and suit numbers involved in w are $O(\log n)$-bit numbers.

NL*-hardness*. We reduce to FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ the co-NL-complete problem of determining whether no path exists from node s to node $t \neq s$ in a directed graph G without self-loops and with edge set $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$. The reduction is rendered delicate by the fact that several cards need to be assigned to each node in G: this is because a card can only occur once in a Klondike instance and furthermore, the "horizontal requirements" arising from the ranks of the cards are of course not compatible with the implicit ordering of the nodes in G. We now describe the flat Klondike instance w produced from G. It uses the cards $c_{i,u}$, $1 \leq i \leq 2n^2$, $1 \leq u \leq n$, arranged in $|E|(n-1)+2$ build stacks as follows:

1. for each edge (i, j) in E and for each k, $0 \leq k < n-1$, one stack with $c_{k(2n)+i,j}$ on top and with $c_{(k+1)(2n)+n+j,i}$ facing down below
2. one stack with $c_{2n^2,s}$ on top and with $c_{1,t}$ facing down below
3. one stack with the remaining cards facing down, in increasing order.

The reader can check that no card is mentioned twice in this (log space) construction. Furthermore, w is nontrivial, since only one build stack contains two cards of the same suit, and this stack is properly ordered. Hence the graph $H(w)$ is defined. It is a $n \times 2n$ grid with the horizontal edges forming n parallel lines running from left to right (card ranks decrease from left to right).

To see how the vertical edges operate, imagine the grid partitioned into columns $n, n-1, \ldots, 1$ of width $2n$. Each such column is further partitioned into a *target* region, of width n, and a *source* region, of width n. The vertical edges arising from $(i, j) \in E$ run from the source region in every column $k+1$ on line i to the target region in column k on line j. Observe then that the vertical $H(w)$ edges arising from E together with the horizontal $H(w)$ edges are incapable of forming a cycle in $H(w)$. The vertical edge $(c_{1,t}, c_{2n^2,s})$ is the only edge in $H(w)$ which connects a column (in fact, the rightmost entry in the source region of the nth column on line t) to a column situated to its left (in fact, to the leftmost entry in the target region of the first column on line s).

It follows that if a cycle exists in $H(w)$, then $(c_{1,t}, c_{2n^2,s})$ is part of it. Hence, if such a cycle exists, a path exists in $H(w)$ from the line s to the line t. This implies that a path existed from s to t in G.

Conversely, if a path $s = v_1, v_2, \ldots, v_m = t$ with $m \leq n-1$ exists in G, then a path can be traced from the column n on line s to the column $n-m$ on line t in $H(w)$ by appropriately combining neighbouring column traversals with horizontal displacements on the successive lines $v_1, v_2, \ldots, v_m$. A final horizontal displacement leads to $c_{1,t}$ and thus to $c_{2n^2,s}$, creating a cycle in $H(w)$.

Hence a cycle exists in $H(w)$ iff a path exists in G. By Proposition 6, a path exists in G iff no win is possible from w. This concludes the NL-hardness proof. □

We now relate the case of an arbitrary number of black suits to the case of a red suit and a black suit. We can show that when generalized King moves are disallowed, the all-blacks case reduces to the case of a red suit and a black suit.

Proposition 8. FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ AC^0*-reduces to* FLAT-SOLIT$_{\text{NoKing}}(1, 1)$.

Proof. Let a FLAT-SOLIT$_{\text{NoKing}}(*, 0)$ instance w involve the nb cards $c_{1,0}, c_{1,1}, \ldots, c_{1,b-1}, c_{2,0}, \ldots, \ldots, c_{n,b-1}$ where $c_{i,s}$ is the suit-$(s+1)$ card of rank i. The idea is to rename each $c_{i,s}$ as a ♡ card, and to use ♣ cards to restrict the release of the renamed cards in such a way as to enforce the rules that had to be followed in w when the original black cards were constrained by their respective suit stacks. Once the renamed images are released, all the auxiliary cards will be released to produce a win in the target {♣, ♡} instance.

This is done as follows. The FLAT-SOLIT$_{\text{NoKing}}(1, 1)$ instance constructed will involve $3nb+1$ club cards and $3nb+1$ heart cards. First, for $0 \leq s < b$, we rename the suit-$(s+1)$ cards in the instance w as follows:

$$\begin{aligned} c_{1,s} &\rightarrow 3ns + 3n♡ \\ c_{2,s} &\rightarrow 3ns + 3n - 3♡ \\ &\ldots \\ c_{n-1,s} &\rightarrow 3ns + 6♡ \\ c_{n,s} &\rightarrow 3ns + 3♡. \end{aligned}$$

Then, for $0 \leq s < b$, we add the n following build stacks, with the top card facing up and the bottom card (when present) facing down:

Below:		$3ns+3n-2\clubsuit$	$\cdots$	$3ns+7\clubsuit$	$3ns+4\clubsuit$
Top:	$3ns+3n+1\clubsuit$	$3ns+3n-1\clubsuit$	$\cdots$	$3ns+8\clubsuit$	$3ns+5\clubsuit$

Finally, the *critical* stack is set to the cards $3ns+2\clubsuit$, $0 \leq s < b$, in any order, followed by the remaining cards $A\clubsuit$, $A\heartsuit$, $2\heartsuit$, $3\clubsuit$, $4\heartsuit$, $5\heartsuit$, $6\clubsuit$, ..., $3nb\clubsuit$, $3nb+1\heartsuit$ in increasing order.

For any s, $0 \leq s < b$, until the $A\heartsuit$ becomes visible, no backward move is possible, and the cards $3ns+3n\heartsuit$, $3ns+3n-3\heartsuit$, ..., $3ns+6\heartsuit$ and $3ns+3\heartsuit$ can only be placed in that order on the n build stacks designed to accept them. This holds for each s independently. Only after the $3ns+3\heartsuit$ cards for $0 \leq s < b$ have found their ways to their mates $3ns+4\clubsuit$ can the critical stack be freed of the b cards $3ns+2\clubsuit$ sitting on top of it. In such an event, all the original build stacks arising from the renamed w cards are empty and only sorted build stacks remain, leading to a win. This happens iff the original w instance was winning. □

Corollary 9. FLAT-SOLIT$_{\text{NoKing}}(1,1)$ *and* SOLIT$(1,1)$ *are* NL*-hard.*

The simplest Klondike restrictions can be solved by constant-depth circuits, as the following shows:

Theorem 10. (a) *For any constant b,* FLAT-SOLIT$_{\text{NoKing}}(b,0)$ *is in* AC^0
(b) FLAT-SOLIT$(1,0)$ *is in* AC^0
(c) SOLIT$(1,0)$ *is in* $\text{AC}^0[3]$.

Proof. Part (a): By Proposition 6, we need to determine whether a cycle exists in the graph $H(w)$ of a nontrivial FLAT-SOLIT$_{\text{NoKing}}(b,0)$ instance w. We first prove it for $b=2$ and explain later how to generalize the proof.

When $b=2$, we claim that $H(w)$ has a cycle iff there exist an edge ($i\clubsuit$, $k\spadesuit$) and an edge ($j\spadesuit$, $\ell\clubsuit$) in $H(w)$ such that $i<\ell$ and $j<k$. This condition is AC^0-testable.

We now prove the claim. Call a pair of edges ($i\clubsuit$, $k\spadesuit$) and ($j\spadesuit$, $\ell\clubsuit$) in $H(w)$ a *crossing* when $i<\ell$ and $j<k$. Clearly, a crossing together with the horizontal edges in $H(w)$ form a cycle. Conversely, let G be a cycle in $H(w)$. The cycle must have cards from both suits otherwise the instance is trivial. Consider the two parallel paths, one for $\clubsuit$ and one for $\spadesuit$, running from left to right and formed by the horizontal edges in $H(w)$. Let $i\clubsuit$ and $j\spadesuit$ be the rightmost (i.e., lowest ranked) $\clubsuit$ and $\spadesuit$ cards that belong to G. The edge in G leaving from $i\clubsuit$ must lead to a $\spadesuit$, say $k\spadesuit$, otherwise the instance is trivial or i was not the rightmost. Similarly, the edge in G leaving from $j\spadesuit$ must lead to a $\clubsuit$, say $\ell\clubsuit$. It is not possible for both $i=\ell$ and $j=k$ to hold, since no configuration of the build stacks can simultaneously give rise to the edges ($i\clubsuit$, $j\spadesuit$) and ($j\spadesuit$, $i\clubsuit$). So assume with no loss of generality that $i<\ell$. Then $j<k$ otherwise the instance is trivial. So we have $i<\ell$ and $j<k$.

For the case $b>2$, we claim that $H(w)$ has a cycle iff there exists a sequence of $d \leq b$ edges $(c_{a_1,s_1}, c_{b_2,s_2})$, $(c_{a_2,s_2}, c_{b_3,s_3})$, $(c_{a_3,s_3}, c_{b_4,s_4})$, ..., $(c_{a_d,s_d}, c_{b_1,s_1})$ such that $a_i \leq b_i$ for $0 \leq i \leq d$. Clearly, these edges together with the horizontal edges in $H(w)$ creating paths from c_{b_i,s_i} to c_{a_i,s_i} form a cycle. Conversely, assuming a cycle G, consider the edges leaving from the rightmost card from G for each suit involved in G in the order they appear in G. These edges have the claimed property. This proves the claim and concludes part (a).

Part (b): Here we only have one suit, but the generalized King can refill an empty build stack. If the generalized King occurs at the bottom of a build stack, then we accept iff the instance is nontrivial. Otherwise, we accept iff two conditions hold:

- every build stack is sorted except the King's build stack which is sorted ignoring the King card, and
- if there is a card c above the King in the King's build stack, then there is another build stack all of whose cards are ranked lower than c.

Part (c): Here we further have to deal with the pile and the talon. If r is the rank of a pile card c, we will denote by $\lfloor r \rfloor$ the rank of the largest ranked card in the pile ranked less than r (if no such card exists then let $\lfloor r \rfloor = r$). Suppose first that the generalized King occurs in the build stacks. Then we accept iff the build stacks pass the test described in part b), and furthermore, for each card c in the pile, say ranked r, we have:

1. if the card c' ranked $\lfloor r \rfloor$ is above c when the pile is facing down, then one of the following conditions holds (where a *large* card is defined as a card ranked higher than r):
   - the number of large cards occurring between c' and c in the pile is congruent to 2 modulo 3, or
   - the number of large cards on top of c in the pile is congruent to 2 modulo 3, or
   - there are no large cards below c in the pile.
2. if the card c' ranked $\lfloor r \rfloor$ is below c when the pile is facing down, then one of the following conditions holds:
   - the number of large cards on top of c in the pile is congruent to 2 modulo 3, or
   - there are no large cards between c' and c on the pile.

	Flat Klondike, no King	Flat Klondike	Klondike
1 black	in AC^0	in AC^0	in AC^0[3]
b blacks	in AC^0	in NL	in NL
$*$ blacks	NL-complete	NL-hard, in NP	NL-hard, in NP
1 black, 1 red	NL-hard, in NP	NL-hard, in NP	NL-hard, in NP
2 blacks, 1 red	NP-complete	NP-complete	NP-complete
$*$ blacks, $*$ red	NP-complete	NP-complete	NP-complete

Fig. 2. Our current knowledge of the complexity of Klondike . A "b" represents any fixed number and an "$*$" represents an input-dependent number.

These tests can be performed in parallel for each card c in AC^0[3]. A case analysis and an induction show that these conditions are necessary and sufficient to make a win possible.

Now suppose that the generalized King occurs in the pile. Then we first check that the instance without the pile is nontrivial. Now let c' be the lowest ranked card sitting at the bottom of a build stack. Note that the option to move the generalized King arises from the moment that c' is displaced to its suit stack. But the King need not be moved immediately: it may be necessary to postpone moving the King until more pile cards have been displaced. This is solved by checking that $\bigvee_c \text{King}(c)$ holds, where c ranges over all pile cards ranked higher than c' and $\text{King}(c)$ stands for the set of modular conditions defined for c above, but now conceptually ranking the King between $\lfloor \text{rank}(c) \rfloor$ and $\text{rank}(c)$ in the card ordering. □

Finally we note an upper bound that applies to the all-black instances:

Proposition 11. *For any constant b,* Solit$(b, 0)$ *is in* NL.

Proof. Consider a Solit$(b, 0)$ instance w. We can build a graph of configurations for the game starting from w. Indeed, note that such a configuration can be deduced deterministically from w and the following data:

- the ranks of the highest ranked cards on top of the b suit stacks,
- the number of cards currently in the talon, and
- the set of Kings that have been moved to an empty build stack so far.

Since the number of suits is fixed, the total number of possible such data values is polynomial. Thus the configuration graph can be computed in deterministic logspace from w. Then it remains to check in NL whether some configuration in which all the Kings sit on the suit stacks is reachable from the initial configuration. □

5. Conclusion

Fig. 2 summarizes what we have learned in this work about the complexity of Klondike. Some gaps are obvious. In particular, the cases involving two suits beg for a more satisfactory characterization. The flat cases of a red suit and a black suit are especially puzzling. We strongly suspect these cases to be in P, but could they possibly be hard for P? Are they in NL? Some simple Klondike cases involve the graphs $H(w)$ built around a grid with the horizontal lines representing the suit stack constraints. Could some of these be related to the grid graph reachability problems studied in [1]?

Our Klondike definition does not allow creating new build stacks in the course of the game, but the initial number of build stacks is not bounded. Does Klondike remain NP-hard if we insist on only seven build stacks initially, as in the usual 52-card Klondike? In the flat version, the number of configurations is then bounded by a polynomial and transitions between these configurations are easy to compute, probably resulting in an NL upper bound. It would seem plausible that the general case with seven stacks be doable in P as well.

Returning to our original motivations, we note on the one hand that Klondike and its restrictions can serve to illustrate NP-completeness, but also a wealth of other complexity classes all the way down to AC^0. On the other hand, Klondike being NP-complete provides absolutely no mathematical justification that investigating the odds of winning in the case of a standard 52-card deck will be difficult. But the fact that Klondike is just another name for SAT can at least be seen as confirmation that the game does involve a good level of intricacy. Fig. 2 might suggest the following: start investigating the odds of winning in the apparently simpler game restrictions and then proceed onwards to the NP-complete cases.

Acknowledgements

We thank François Laviolette from Laval University and Philippe Beaudoin from the University of British Columbia for their help in proving the Klondike NP upper bound in Theorem 2. The second author thanks Andreas Krebs and Christoph Behle in Tübingen for helpful discussions.

The second author was supported by the Natural Sciences and Engineering Research Council of Canada and the *Fonds de recherche sur la nature et les technologies du Québec*.

References

[1] E. Allender, D. Barrington, T. Chakraborty, S. Datta, S. Roy, Grid graph reachability problems, in: Proc. 21st Annual IEEE Conference on Computational Complexity, pp. 299–313, 2006.

[2] D. Barrington, N. Immerman, H. Straubing, On uniformity within NC^1, Journal of Computer and System Sciences 41 (3) (1990) 274–306.

[3] S. Buss, S. Cook, A. Gupta, V. Ramachandran, An optimal parallel algorithm for formula evaluation, SIAM Journal on Computing 21 (1992) 755–780.
[4] Malte Helmert, Complexity results for standard benchmark domains in planning, Artificial Intelligence 143 (2) (2003) 219–262.
[5] I. Gent, C. Jefferson, I. Lynce, I. Miguel, P. Nightingale, B. Smith, A. Tarim, Search in the Patience Game "Black Hole", in: AI Communications, ISSN: 0921-7126, IOS Press, pp. 1–15.
[6] R. Kaye, Minesweeper is NP-complete, in: The Mathematical Intelligencer, vol. 22, no. 2, Springer Verlag, 2000, pp. 9–15.
[7] C. Papadimitriou, Computational Complexity, Addison-Wesley, 1994.
[8] D. Parlett, Solitaire: Aces Up and 399 Other Card Games, Pantheon, 1979.
[9] D. Parlett, A History of Card Games, Oxford University Press, 1991.
[10] X. Yan, P. Diaconis, P. Rusmevichientong, B. Van Roy, Solitaire: Man Versus Machine, in: Proc. Advances in Neural Information Processing Systems, 17, 2004, NIPS.