Complexity of Scorpion Solitaire and applications to Klondike[⋆]
Extended Abstract

Francesco Arena and Miriam Di Ianni[1]

Dipartimento di Ingegneria dell'Impresa "Mario Lucertini",
University of Rome Tor Vergata
francesco.arena.160@gmail.com and miriam.di.ianni@uniroma2.it

Abstract. Scorpion is a puzzle game that falls under the category of Solitaires. The problem of determining whether a starting configuration of a Solitaire game (generalised to contain an arbitrary number of cards) can lead to a winning configuration has been studied from the perspective of Computational Complexity for some of the most popular Solitaires, namely Free Cell [1], Klondike [2] and Spider [3], all resulting in hardness proofs. Scorpion, though, has a unique twist compared to aforementioned ones: each card can be moved to cover only one other card. This property narrows the set of possible moves and makes a worthwhile issue investigating whether the same problem proved NP-complete for Free Cell, Klondike and Spider falls in P as far as Scorpion is considered. In this paper we prove that the problem of deciding whether an n-card s-suits initial configuration of Scorpion allows for a winning sequence of moves is NP-complete for any $s \geq 1$, and it remains so also when the initial configuration contains no face down (locked) cards. We then negatively answer a question posed in [2] by proving that it is NP-complete to decide whether an n-card one red suit-one black suit configuration of the Klondike Solitaire allows for a winning sequence of moves.

Keywords: Card Solitaires · Computational Complexity.

1 Introduction

Solitaire games are one-player games, or puzzles, which employ a deck of cards and a tableau of possible positions for the cards. The goal of the game is to achieve a winning configuration of the cards on the tableau by applying legitimate moves to an initial configuration. The possible decks, tableaux, moves, initial and winning configurations are determined by the rules of the particular type of Solitaire one intends to play. According to [5], solitaire games came into existence when fortune-telling with cards gained popularity in the eighteenth century. Many variations of solitaire exist today, among which we remind Klondike, Free Cell, Spider, and Scorpion.

[⋆] Copyright © 2020 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

In this paper, we consider the Scorpion Solitaire that we now introduce in its traditionally played version. The deck is composed of 52 cards, partitioned into 4 suits of 13 cards each ($\heartsuit$, $\diamondsuit$, $\clubsuit$ or $\spadesuit$), ranging from *rank* 1 (*Ace*) to 13 (*King*). The setting consists of a *tableau* and a *stock*, with the tableau, in turn, consisting of 7 columns each of which may contain a stack of cards, and the stock being a special reserve set of cards. At the start of the game an initial layout is supplied, with 3 face down cards placed in the stock and 7 stacks of cards dealt in the tableau: the first 4 stacks contain 2 face down cards at the bottom and 5 face up cards on top, while the last 3 stacks consist of 7 face up cards (see Figure 1). The goal of the game is to build 4 stacks of cards on the tableau,

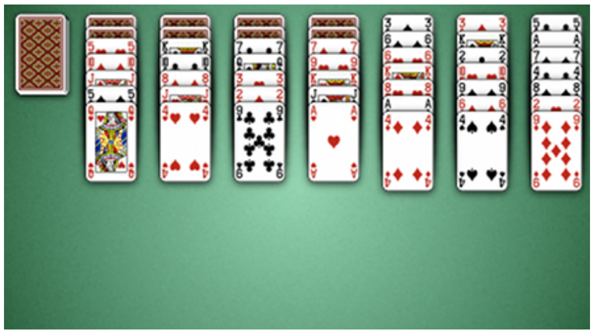

Fig. 1. Example of an initial Scorpion layout.

each of them containing 13 cards of the same suit arranged in descending order, with the King (rank 13) at the bottom and the Ace (rank 1) on top of each stack. The goal is pursued by performing a sequence of moves: a face up card x is allowed to be moved from the stack containing it to cover card y only if y is the top card of a different stack , $suit(x) = suit(y)$ and $rank(y) = rank(x) + 1$. Hence, any non-King card x is allowed to perform one single move within the game, and has no choice about which move to perform. When a card x is moved, all the cards covering x are moved together as a unit, preserving their relative order (see Figure 2). A face up King and all the cards covering it can be moved to any empty column slot on the tableau, which appears after a card located at the bottom of its own stack is moved. A face down card is turned up when it becomes the top card of its stack. At any point in the game, the player can deal the 3 cards in the stock on top of the stacks located in the 3 leftmost column slots on the tableau.

We study the computational complexity of deciding, given an initial layout, if a sequence of moves exists achieving the goal. Needless to say, we shall consider generalized layouts, that is, layouts dealing with a deck of s suits with n cards each, with a tableau composed of k columns of arbitrary size in the initial arrangement and a stock with d cards in the initial arrangement (with $d \leq k$).

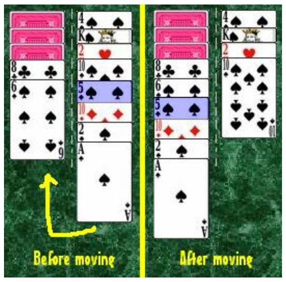

Fig. 2. Example of move of a non-King card.

Differently from the original game, we deal with a *complete information game*, that is, we have no face down cards. Instead, we may have face up *locked cards*: a locked card cannot be moved until the card covering it is not moved; in particular, all cards in the stock will be considered locked. Our choice of dealing with complete information games is shared with the literature on this topic [1–4].

The study of Solitaire card games from a computational complexity perspective is not a novel topic, where (as we remarked a few lines above), whenever we talk about a complexity result for a Solitaire game, we implicitly refer to generalized versions employing decks of arbitrarily large size. The first major result about computational complexity in Solitaire games comes from a paper about Automated Planning domains used in AI planning competitions [1]. Among several results, the paper proves that planning a winning strategy for a Free Cell game is a hard problem; more precisely, the author shows, by a reduction from 3SAT, that it is NP-Complete to decide whether a winning strategy for Free Cell exists. The NP-Completeness proof in [3] for Spider also follows from a reduction from 3SAT.
The hardness proofs for the Scorpion Solitaire we present in this paper are also reductions from 3SAT. However, they require a substantially different approach to model the non deterministic nature of the choice of a truth assignment to the variables in a boolean formula: indeed, while a non deterministic behaviour is intrinsic in the Free Cell (or Spider) playing rules, in that each (let's say, black) card may be covered by two (let's say, red) cards (for instance card $(3, \clubsuit)$ may be covered by both $(2, \heartsuit)$ and $(2, \diamondsuit)$), the Scorpion playing rules state that any (non-ace) card may be covered by exactly one other card (that is, card $(3, \clubsuit)$ may be covered only by $(2, \clubsuit)$).
In [2] Klondike, one of the most popular Solitaires games (partly because of its presence in Microsoft Windows pre-installed games), gets thoroughly analysed from a Computational Complexity perspective. The problem of solving the classical 4-suits version of the game is proved to be NP-Complete, again by a reduction from 3SAT (notice that Klondike rules share with Free Cell the "multiple coverings" feature). The paper then turns to analyse the complexity of many restricted versions of Klondike, achieving membership/hardness proofs to/for a

variety of complexity classes. In particular, the one-black one-red suits version, is considered in [2] and proved to be in NP and at least NL-hard. The presence of just one red suit and just one black suit weakens the, let's say, intrinsic non deterministic behaviour of the Solitaire playing rules: as noticed by the same auhors in [2], the "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?". Actually, In this paper, exploiting its similarities with Scorpion (in its forbidding the "multiple coverings" feature), we prove that such one-black one-red suits version of Klondike is NP-complete, so strengthening the result in [2] and negatively answering the question posed therein.

After having provided the needed formal definitions in Section 2, in Section 3 we prove the NP-completeness of deciding if a winning strategy for a 4-suits Scorpion game exists even when the initial setting has an empty stock. In Section 3, we also state the stronger result according to which the problem remains NP-complete for 1-suit games and initial setting not containing locked cards. Then, in Section 4, we prove the NP-completeness of the one-black one-red suits version of Klondike. Finally, in Section 5 we draw some conclusions and discuss some open problems.

2 Preliminaries

Let $n \in \mathbb{N}$ and $s \in \mathbb{N}$; a deck of cards is a set $D = [n] \times [s]$[1]. A *Scorpion layout* $L = \{S, C_1, \dots, C_k\}$ for a deck of cards D is an arrangement of the cards in D into k *columns* $C_1, \dots, C_k$ and a *stock* S. The stock is a sequence $\langle s_1, s_2, \dots, s_d\rangle$ of $d \leq k$ cards. Each column consists of a pair of (possibly empty) stacks: the *locked* cards stack and the *unlocked* cards stack. Let $\langle u_1, u_2, \dots u_h\rangle$ be the (non-empty) unlocked cards stack in some column: for any $j \in 2, \dots, h-1$, the sequence of cards $\langle u_j, \dots u_h\rangle$ is called a *sub-column* starting at u_j, card u_1 is the *base* of the column, card u_h is the *top* of the column and, for $i = 2, \dots, h$, card u_j *covers* card u_{j-1}.

A *Scorpion game* is a sequence $\langle L^0, L^1, \dots, L^n\rangle$ of Scorpion layouts such that, for any $t = 1, \dots, n$, the *transition* from L^t to L^{t+1} occurs according to one of the (mutually exclusive) rules described in the following.

1) L^t has a non-empty stock and L^{t+1} is derived from L^t by moving *all* cards in the stock: for $h = 1, \dots, d$, the hth card in the stock in L^t becomes the (unlocked) top card of column C_h in L^{t+1} and the stock of L^{t+1} is empty.
2) In L^t card (i, j) is the top card of column ℓ, $(i-1, j)$ is unlocked and contained in column $\ell' \neq \ell$, and L^{t+1} is derived from L^t by moving card $(i-1, j)$ to cover card (i, j): the whole sub-column starting at $(i-1, j)$ is moved to column ℓ so that the top card of column ℓ' in L^t is the top card of column of column ℓ in L^{t+1}. If $(i-1, j)$ covers card (i', j') in L^t, then (i', j') is the top card of column ℓ' in L^{t+1} while, if $(i-1, j)$ is the base card of column ℓ' in L^t, column ℓ' is empty in L^{t+1}.

[1] For $m \in \mathbb{N}$, we shortly denote as $[m]$ the set $\{1, 2, \dots, m\}$.

3) In L^t column ℓ is empty and an unlocked card (n, j) is contained in column $\ell' \neq \ell$, and L^{t+1} is derived from L^t by moving to column ℓ the subcolumn starting at (n, j): in L^{t+1} the base of column ℓ is (n, j), the top card of column ℓ is the top card of column ℓ' in L^t, and similarly as in the previous case as to the top of column ℓ'.

A *winning Scorpion layout* is a Scorpion layout L such that in L exactly s columns are non empty, the base of each of them is a (different) card (n, j), with $j \in [s]$, and, for any $i \in [n-1]$ and $j \in [s]$, card (i, j) covers card $(i+1, j)$.

A *winning Scorpion game starting at* L^0 is a Scorpion game $\langle L^0, L^1, \dots, L^p \rangle$ such that L^p is a winning Scorpion layout.

Given a deck D of cards and a Scorpion layout L^0, the Scorpion Winning Game problem (in short, SWG) consists in deciding if there exists a winning Scorpion game starting at L^0.

3 Computational Complexity of Scorpion

Our first step to the study of the computational complexity of SWG is the proof of its membership to NP. To this aim, we claim that, for any Scorpion layout L^0, if there exists a winning Scorpion game starting at L^0 then there also exists a winning Scorpion game $\langle L^0, \overline{L}^1, \dots, \overline{L}^p \rangle$ with $p \leq ns + 1 + 2|S|$. Indeed, since any card $(i, j) \in D$ such that $i < n$ may be moved at most once (that is, once it covers card $(i+1, j)$ it can be no longer moved), then a winning Scorpion game consists of at most one transition occurring according to rule 1), at most $(n-1)s$ transitions occurring according to rule 2), at most s transitions occurring according to rule 3) each of which places a different card (n, j) as the base of a column (for any $j \in [s]$), and a set of transitions still occurring according to rule 3) each of which move some card (n, j) from the base of a column to another. At most $2|S|$ moves in the last set are meaningful as to compose a winning game: they are those pairs of moves such that one move frees one of the first $|S|$ columns and the other move possibly shifts to it another column having as its base some (n, j) and as its top an appropriate card to be covered by a card in the stock.
Hence, a certificate for an instance L^0 of SWG is a sequence $L^1, \dots, L^p$ of $p \leq ns + 1 + 2|S|$ Scorpion layouts. Since verifying if $\langle L^0, L^1, \dots, L^p \rangle$ is actually a Scorpion winning game requires polynomial time, this proves that SWG is in NP.

We now turn to proving the NP-completeness of SWG.

Theorem 1. *SWG is NP-Complete even when restricted to set of instances such that D is a deck on four suits and the stock in the initial layout is empty.*

Proof. To prove the assertion, we provide a reduction from 3SAT. Let $\langle X, f \rangle$ be an instance of 3SAT, where $X = \{x_1, \dots, x_n\}$ is a set of boolean variables and $f = \{c_1, \dots, c_m\}$ is a set of clauses over X each containing exactly three literals (each literal, in turn, being a variable in X or its negation). We write $l_{i,j}$ for the

ith literal in c_j ($i = 1, 2, 3$), and, with a slight abuse of notation, we write $x_i \in c_j$ ($\neg x_i \in c_j$) if x_i (respectively, $\neg x_i$) is a literal in c_j. Without loss of generality, in what follows we shall assume that $m \geq n$.

The description of the instance $\langle D, L_f^0 \rangle$ of SWG corresponding to an instance $\langle X, f \rangle$ of 3SAT is rather intricate and the reader may refer to Fig. 3 to get some intuition about how the gadgets are structured.

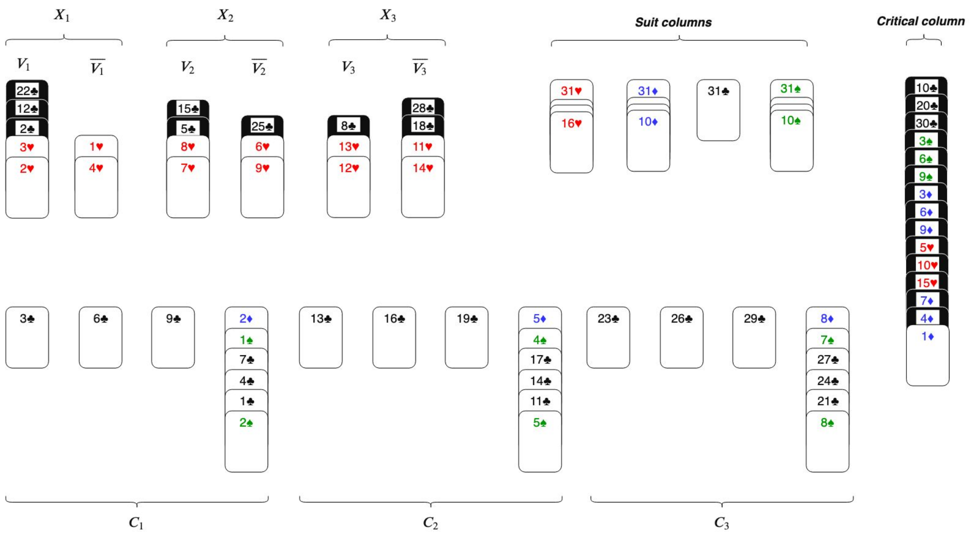

Fig. 3. Example of the instance of SWG corresponding to the 3SAT formula $f = \{c_1, c_2, c_3\}$ with $c_1 = \{x_3, x_2, x_1\}$, $c_2 = \{\neg x_3, x_2, x_1\}$, $c_3 = \{\neg x_3, \neg x_2, x_1\}$, over the set $X = \{x_1, x_2, x_3\}$.

The deck of cards D consists of 4 suits of $n_f = 10m + 1$ cards each. In order to make the proof closer to the language of card players, in what follows we shall denote the four suits as $\heartsuit$, $\clubsuit$, $\diamondsuit$ and $\spadesuit$ (instead of $1, 2, 3, 4$ as defined in Section 2).

For any $1 \leq i \leq n$, we define the *base variable value of* x_i as $\alpha_i = 5(i - 1)$ and, for any $1 \leq j \leq m$, we define the *base clause value of* c_j as $\beta_j = 3(j - 1)$ and the *base clubs clause value of* c_j as $\gamma_j = 10(j - 1)$.

The columns of L_f^0 are partitioned into three sets: a set of *variable gadgets* columns, a set of *clause gadgets* columns, and a set consisting of one *critical columns* and four *suit columns.*

Variable gadgets columns. Each variable x_i is associated to a pair of columns denoted as v_i and $\bar{v}_i$:

1. column v_i contains the unlocked cards $(\alpha_i + 3, \heartsuit)$ and $(\alpha_i + 2, \heartsuit)$, $(\alpha_i + 2, \heartsuit)$ being the top of column v_j and covering card $(\alpha_i + 3, \heartsuit)$;

2. column $\bar{v}_i$ contains the unlocked cards $(\alpha_i+1,\heartsuit)$ and $(\alpha_i+4,\heartsuit)$, $(\alpha_i+4,\heartsuit)$ being the top of column v_j and covering card $(\alpha_i+1,\heartsuit)$.

Clause gadgets columns. Each clause c_j is associated to the four columns u_{j_1}, u_{j_2}, u_{j_3} and u_{j_4}:

1. u_{j_1}, u_{j_2} and u_{j_3} contain a single unlocked card each (being their, respective, top card): u_{j_1} contains $(\gamma_j+3,\clubsuit)$, u_{j_2} contains $(\gamma_j+6,\clubsuit)$ and u_{j_3} contains $(\gamma_j+9,\clubsuit)$;
2. finally, u_{j_4} contains the following stack of unlocked cards:

$$\langle(\beta_j+2,\diamondsuit),(\beta_j+1,\spadesuit),(\gamma_j+7,\clubsuit),(\gamma_j+4,\clubsuit),(\gamma_j+1,\clubsuit),(\beta_j+2,\spadesuit)\rangle,$$

with $(\beta_j+2,\spadesuit)$ at its top.

We can now model a variable (or its negation) being contained in a clause. Starting with $j=1$, and up to $j=m$, we repeat the following steps: for any $1\le i\le n$,

- if $l_{1,j}=x_i$ then we add card $(\gamma_j+8,\clubsuit)$ as a locked card at the bottom of column v_i, while if $l_{1,j}=\neg x_i$ then we add card $(\gamma_j+8,\clubsuit)$ as a locked card at the bottom of column $\bar{v}_i$;
- if $l_{2,j}=x_i$ then we add card $(\gamma_j+5,\clubsuit)$ as a locked card at the bottom of column v_i, while if $l_{2,j}=\neg x_i$ then we add card $(\gamma_j+5,\clubsuit)$ as a locked card at the bottom of column $\bar{v}_i$;
- if $l_{3,j}=x_i$ then we add card $(\gamma_j+2,\clubsuit)$ as a locked card at the bottom of column v_i, while if $l_{3,j}=\neg x_i$ then we add card $(\gamma_j+2,\clubsuit)$ as a locked card at the bottom of column $\bar{v}_i$.

As an example, if $c_2=\{\neg x_2,x_3,\neg x_8\}$ then we add $(18,\clubsuit)$ to the bottom of column $\bar{v}_2$, $(15,\clubsuit)$ to the bottom of column v_3 and $(12,\clubsuit)$ to the bottom of column $\bar{v}_8$, all of them locked.

Critical column: the *critical column* consists of the single unlocked card $(\beta_m+1,\diamondsuit)$, which is both base and top of the critical column and covers the following stack of locked cards:

$$\begin{aligned}\langle\ &(10,\clubsuit),(20,\clubsuit),\ldots,(10m,\clubsuit),\\ &(3,\spadesuit),(6,\spadesuit),\ldots,(3m,\spadesuit),\\ &(3,\diamondsuit),(6,\diamondsuit),\ldots,(3(m-1),\diamondsuit),(3m,\diamondsuit),\\ &(5,\heartsuit),(10,\heartsuit),\ldots,(5n,\heartsuit),\\ &(1,\diamondsuit)=(\beta_1+1,\diamondsuit),(4,\diamondsuit)=(\beta_2+1,\diamondsuit),\ldots,(\beta_{m-1}+1,\diamondsuit)\ \rangle.\end{aligned}$$

As to the *suit columns*, that is, the *hearts column*, the *spades column*, the *diamonds column* and the *clubs column*, all the cards they contain are unlocked and

- the *hearts column* consists of the stack

$$\langle(n_f,\heartsuit),(n_f-1,\heartsuit),\ldots,(5n+1,\heartsuit)\rangle$$

with $(5n+1,\heartsuit)$ as its top;

– the remaining three suit columns have a similar structure, that is,

spades column:	$\langle (n,_f \spadesuit), (n_f - 1, \spadesuit), \ldots, (3m+1, \spadesuit) \rangle$,
diamonds column:	$\langle (n_f, \diamondsuit, (n_f - 1, \diamondsuit), \ldots, (3m+1\diamondsuit) \rangle$,
clubs column:	$\langle (10m+1, \clubsuit) \rangle$.

The description of L_f^0 is now complete. Needless to say, deriving L_f^0 from $\langle X, f \rangle$ requires polynomial time.

It remains to show that there exists a winning Scorpion game starting at L_f^0 if and only if there exists a truth assignment for X satisfying all clauses in f.

Assume there is a truth assignment $a : X \to \{\texttt{true}, \texttt{false}\}$ satisfying f. We now derive from a a winning Scorpion game starting at L_f^0; for the sake of readability, in what follows we shall describe the sequence of card moves yielding the Scorpion layouts by which the winning game is composed.

For any $1 \le i \le n$, the following steps are repeated.

– If $a(x_i) = \texttt{true}$, the sub-column $\langle (\alpha_i + 3, \heartsuit), (\alpha_i + 2, \heartsuit) \rangle$ is moved from column v_i to column $\bar{v_i}$ so that card $(\alpha_i + 3, \heartsuit)$ covers card $(\alpha_i + 4, \heartsuit)$. In this way, cards in the locked stack of column v_i get unlocked one after the other and can be moved to become the top cards of some clause gadget column: if, for some $1 \le j \le m$, card $(\gamma_j + 2, \clubsuit)$ (or $(\gamma_j + 5, \clubsuit)$ or $(\gamma_j + 8, \clubsuit)$) has become the top card of column v_i then it gets unlocked and it is moved to cover card $(\gamma_j + 3, \clubsuit)$ in column u_{j_1} (respectively, card $(\gamma_j + 6, \clubsuit)$ in column u_{j_2} or card $(\gamma_j + 9, \clubsuit)$ in column u_{j_3}). This makes another card in the locked stack unlocked, and so on.
– If $a(x_i) = \texttt{false}$, the sub-column $\langle (\alpha_i + 1, \heartsuit), (\alpha_i + 4, \heartsuit) \rangle$ is moved from column $\bar{v_i}$ to column v_i so that card $(\alpha_i + 1, \heartsuit)$ covers $(\alpha_i + 2, \heartsuit)$. In this way, just as in the previous case, cards in the locked stack of column $\bar{v_i}$ get unlocked one after the other and are moved to become top cards of some clause gadget column.

Since a satisfies all clauses in f then, for each $1 \le j \le m$, at least one column out of $u_{j_1}, u_{j_2}, u_{j_3}$ has its top card changed into, respectively, $(\gamma_j + 2, \clubsuit)$, $(\gamma_j + 5, \clubsuit)$ and $(\gamma_j + 8, \clubsuit)$; we can therefore move at least one of the three cards $(\gamma_j + 7, \clubsuit)$, $(\gamma_j + 4, \clubsuit)$, or $(\gamma_j + 1, \clubsuit)$ from u_{j_4} to cover the new top card in u_{j_1}, or u_{j_2} or u_{j_3}. After this, the top card of either u_{j_1}, or u_{j_2} or u_{j_3} has become $(\beta_j + 2, \spadesuit)$: we can then move $(\beta_j + 1, \spadesuit)$ from u_{j_4} to cover $(\beta_j + 2, \spadesuit)$. By doing so, card $(\beta_j + 2, \diamondsuit)$ has become the top card of column u_{j_4}.

We have, thus, derived a Scorpion Layout such that, for any $1 \le j \le m$, card $(\beta_j + 2, \diamondsuit)$ is the top card of column u_{j_4} (and, actually, its only card). At this point, we move $(\beta_m + 1, \diamondsuit)$ from the critical column to cover $(\beta_m + 2, \diamondsuit)$ on top of column u_{m_4}, so unlocking $(\beta_{m-1} + 1, \diamondsuit)$ in the critical column. This last card, in turn, can be moved to cover card $(\beta_{m-1} + 2, \diamondsuit)$ on top of column $u_{(m-1)_4}$, and so on. At the end of this procedure, card $(5n, \heartsuit)$ has been unlocked and has become the top card of the critical column: we move it to cover the top card in the hearts column, then we cover it by moving card $(5n - 1, \heartsuit) = (\alpha_n + 4, \heartsuit)$ from column v_n or $\bar{v_n}$. Continuing, we move all the $\heartsuit$ cards onto the hearts pile.

Now, card $(3m, \diamondsuit)$ is the top card of the critical column and, hence, unlocked. It can, thus, be moved on top of the diamonds column and, then, be covered by card $(3m-1, \diamondsuit)$ that is, by the sub-column $\langle(3m-1, \diamondsuit), (3m-2, \diamondsuit)\rangle$, that moves from column u_{m_4}, followed by $(3m-2, \diamondsuit)$ from the critical column. Proceeding this way, we bring all the $\diamondsuit$ cards in the diamonds column.
Similarly, all $\spadesuit$ cards are moved to the spades column and all $\clubsuit$ cards are moved to the clubs column.
We have thus derived from a a winning Scorpion game starting at L_f^0.

Conversely, assume there exists a winning Scorpion game $\langle L_f^0, L^1, \ldots, L^M\rangle$. This means that there exists $1 \leq k \leq M$ such that the critical column in L^k does not contain any $\diamondsuit$. In L_f^0, the critical column contains a single unlocked card, that is, $(\beta_m + 1, \diamondsuit)$, and such card has to be moved in order to unlock the card it covers ($(\beta_{m-1} + 1, \diamondsuit)$). Before card $(\beta_m + 1, \diamondsuit)$ can be moved, card $(\beta_m + 1, \spadesuit)$ (that, in L_f^0, covers card $(\beta_m + 2, \diamondsuit)$ in column u_{m_4}) has to be moved and this is possible only after $(\beta_m + 2, \spadesuit)$ has been moved to a different column. This is possible either by direcly moving $(\beta_m + 2, \spadesuit)$ to cover $(\beta_m + 3, \spadesuit)$ or by moving any of the $\clubsuit$ cards lying in u_{m_4} beneath $(\beta_m + 2, \spadesuit)$ (so moving the whole sub-column). The former case cannot happen, since $(\beta_m + 3, \spadesuit) = (3m, \spadesuit)$ is locked in the critical column and can become unlocked only after all $\diamondsuit$ cards above it are moved. Hence, in order to move card $(\beta_m + 1, \diamondsuit)$, some card out of $(\gamma_m + 7, \clubsuit)$, $(\gamma_m + 4, \clubsuit)$, $(\gamma_m + 1, \clubsuit)$ (the cards lying on u_{m_4} beneath $(\beta_m + 2, \spadesuit)$) are to be moved first. All such $\clubsuit$ cards are locked in some variable gadget column and they become unlocked only after the $\heartsuit$ segment at its top is moved, that is,

a) after the sub-column $\langle(\alpha_i + 3, \heartsuit), (\alpha_i + 2, \heartsuit)\rangle$ is moved to cover card $(\alpha_i + 4, \heartsuit)$ (in column $\bar{v}_i$), for some v_i containing $(\gamma_m + 7, \clubsuit)$, or $(\gamma_m + 4, \clubsuit)$, or $(\gamma_m + 1, \clubsuit)$,
b) or after the sub-column $\langle(\alpha_i + 1, \heartsuit), (\alpha_i + 4, \heartsuit)\rangle$ is moved to cover card $(\alpha_i + 2, \heartsuit)$ (in column v_i), for some $\bar{v}_i$ containing $(\gamma_m + 7, \clubsuit)$, or $(\gamma_m + 4, \clubsuit)$, or $(\gamma_m + 1, \clubsuit)$.

Summarizing, card $(\beta_m + 1, \diamondsuit)$ is moved only after $(\alpha_i + 3, \heartsuit)$ has been moved to cover $(\alpha_i + 4, \heartsuit)$ or $(\alpha_i + 1, \heartsuit)$ has been moved to cover $(\alpha_i + 2, \heartsuit)$. Notice that these last two moves are mutually exclusive, that is, the occurrence of one of them forbids the occurrence of the other; hence, we may set

$$a(x_i) = \begin{cases} \texttt{true} & \text{if case a) occurs, that is,} \\ & (\alpha_i + 3, \heartsuit) \text{ is moved to cover } (\alpha_i + 4, \heartsuit), \\ \texttt{false} & \text{if case b) occurs, that is,} \\ & (\alpha_i + 1, \heartsuit) \text{ is moved to cover } (\alpha_i + 2, \heartsuit). \end{cases}$$

Since, by construction, case a) occurs when $x_i \in c_m$ and case b) occurs when $\neg x_i \in c_m$, the truth assignment above to variable x_i satisfies clause c_m.
The above reasoning can be repeated for card $(\beta_{m-1} + 1, \diamondsuit)$ and, hence, clause c_{m-1}, and so on down to card $(\beta_1 + 1, \diamondsuit)$ and, hence, clause c_1. This proves that the so built truth assignment a satisfies all clauses in f.

It turns out that SWG is NP-complete even for sets of instances satisfying stronger restrictions than those considered in Theorem 1, that is, when the game is played with a single suit deck of cards ($s = 1$), and the initial layout has an empty stock and no locked cards. Since the proof of this case is much more intricate and long of that of Theorem 1, for the sake of simplicity and due to space constraints, we have shown the proof for the case of a traditional 4 suits deck of cards and an initial layout containing locked cards. The proof of the following, stronger, result is then deferred to the full version of the paper.

Theorem 2. *SWG is NP-Complete even when restricted to set of instances such that D is a deck on just one suit, and the initial layout contains an empty stock and no locked cards.*

We explicitly observe that, by simple generalization, from Theorem 2 it follows that SWG is NP-complete for any (fixed) number of suits.

4 Klondike

Klondike is played with a 4 suits deck of cards and its setting consists of a *stock*, a *waste*, 4 *foundations* (one foundation per suit) and a *tableau* (see Figure 4).
The foundations are initially empty, but, as the game progresses, the foundations will be built up by suit. The tableau consists of seven stacks of cards: at the beginning, all cards in each stack are face down except for the top card which is left face up. Goal of a Klondike game is to place all the cards into the foundations: when a foundation is empty, as soon as an Ace appears as the top card of any stack in the tableau, it can be removed from the tableau to become the top card of that foundation; when the top card of a foundation is card i of suit x, with $1 \leq i < 13$ and $x \in \{\heartsuit, \clubsuit, \diamondsuit, \spadesuit\}$, as soon as card $i+1$ of suit x appears as the top card of any stack in the tableau, it can be removed from the tableau to become the top card of that foundation. Hence, the last card moved to a foundation is a King.
The tableau is used to organize cards until it becomes possible to place them in the foundations. Cards can be moved among the stacks in the tableau by alternating color: if the top card of a stack in the tableau is card i of a red suit ($\heartsuit$ or $\diamondsuit$) and card $i-1$ of a black suit ($\clubsuit$ or $\spadesuit$) is a face up card in a different stack, then card $i-1$ can be moved to cover card i (and similarly with opposite colors). Similarly to Scorpion Solitaire, whenever a card is moved from a source stack to a destination stack in the tableau all (face up) cards on the source stack lying above the moved card must be moved as well, in their order, to the destination stack. Only a king can be placed into an empty stack. If a face down card is on the top of a stack, it becomes face up.
The stock is a stack-like set of face down cards: when the player clicks on the stock, 3 cards are dealt face up to the waste. Only the top most card in the waste is playable (once it has been played, the card immediately below it becomes playable, and so on): more precisely, it can be moved to cover a face up card in some stack in the tableau in decreasing order and alternating color (that is, a

red suit card i may cover a black suit card $i+1$ and a black suit card i may cover a red suit card $i+1$).

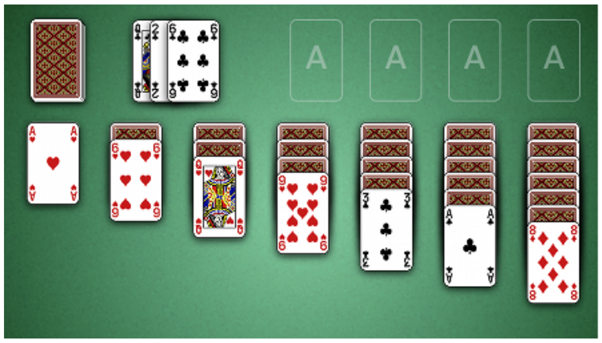

Fig. 4. Example of a Klondike layout.

We again consider complete information games (in [2]'s terminology *thoughtful solitaire*), that is, all cards will be face up, although some of them will be *locked* till the cards covering them are removed. Needless to say, our decks contain arbitrary amounts of cards, and the tableau arbitrary amount of stacks. Furthermore, even though it isn't expressly mentioned in [2], we are assuming that the starting configuration allows for stacks in the tableau with possibly more than one face up card on their top.

Among the other results, [2] proves it is NL-hard to decide if there exists a winning Klondike game starting at a given layout also when the deck of cards is restricted to have only one black suit and one red suit, the initial layout has empty stock and waste and *generalized kings* (namely, maximum rank cards) are not allowed to be moved to an empty stack in the tableau. In [2], such restricted version of the Klondike related decision problem is named FLAT-SOLIT(1,1). We have strengthened the result in [2] as summarized in the next theorem.

Theorem 3. *FLAT-SOLIT(1,1) is NP-Complete.*

Due to space limitations, the proof of Theorem 3 will be provided in the full paper. For the time being, we want to notice the similarities occurring between SWG restricted to 2 suits decks and empty stock and FLAT-SOLIT(1,1): indeed, in both problems, for every card c (other than maximum rank cards) there exists a unique card c' such that c may be moved to cover c' in the tableau and, conversely, for every card c (other than aces) there exists a unique card c' such that c' may be moved to cover c in the tableau. However, there is also a noticeable difference between the two games: according to the Klondike rules, foundations are filled by cards of the same suit, in decreasing order, and by moving one card after the other. No similar structure exists in the Scorpion Solitaire, and this is the reason why it does not seem so easy to directly design a reduction from

SWG to FLAT-SOLIT(1,1). and we have, instead, exploited again a reduction from 3SAT to prove Theorem 3. However, the Klondike layout corresponding to an instance of 3SAT we construct in the proof of Theorem 3 is directly derived from the Scorpion layout associated to the same instance built in Theorem 2.

5 Conclusions and open problems

In this paper we have studied the computational complexity of deciding if a winning strategy exists for a Scorpion Solitaire game and we have proved that the problem is NP-complete even for one suit games and no face down (locked) cards. We have also negatively answered a question posed in [2] by proving that it is NP-complete to decide whether a one red suit-one black suit configuration of the Klondike Solitaire allows for a winning sequence of moves.

A Scorpion game depends on several parameters: number of suits, of cards per suit and of locked cards, stock size, number of columns. It seems quite clear after Theorem 2 that the complexity of the problem is not influenced by number of suits, number of locked cards and stock size.
Instead, the role played as to the complexity of the problem by the number of columns and by the relation between number of cards and number of columns looks intriguing. In fact, the problem is trivially in P when the number of column is one or is equal to the (total) number of cards. And it seems reasonable to conjecture that the problem remains in P when the number of columns is "reasonably" larger than half of the (total) number of cards. On the other hand, the Scorpion layout corresponding to an instance $\langle X, f \rangle$ of 3SAT derived in the proof of Theorem 2 is built over a deck of $8n + 24m$ cards and consists of $2n + 4m + 1$ columns (where $n = |X|$ and m is the numbe of clauses in f): that is, the problem is NP-complete when the number of columns is much smaller than half of the number of cards. But what about an even smaller number of columns? Actually, it seems quite a hard issue to understand what happens, for instance, for a constant number of columns; even for just two columns. We think investigating such topic could be interesting, eventually also in the parameterized complexity setting.

References

1. Malte Helmert. *Complexity results for standard benchmark domains in planning.* Artificial Intelligence. 143 (2), 219-262, 2003.
2. Luc Longpréa e Pierre McKenzie, *The complexity of Solitaire.* Theoretical Computer Science, 410 (50), 5252-5260, 2009.
3. Jesse Stern, *Spider is NP-Complete*, arXiv:1110.1052v1 [cs.CC], 2011.
4. X. Yan, P. Diaconis, P. Rusmevichientong, B. Van Roy, *Solitaire: Man Versus Machine*, in: Proc. Advances in Neural Information Processing Systems (NIPS), 17, 1553-1560, 2004.
5. D. Parlett. *A History of Card Games.* Oxford University Press, 1991.