Uniform-Budget Solo Chess with Only Rooks or Only Knights Is Hard

Davide Bilò
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Luca Di Donato
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Luciano Gualà
Department of Enterprise Engineering, University of Rome "Tor Vergata", Italy

Stefano Leucci
Department of Information Engineering, Computer Science and Mathematics, University of L'Aquila, Italy

Abstract

We study the Solo-Chess problem which has been introduced in [Aravind et al., FUN 2022]. This is a single-player variant of chess in which the player must clear all but one piece from the board via a sequence captures while ensuring that the number of captures performed by each piece does not exceed the piece's *budget*. The time complexity of finding a winning sequence of captures has already been pinpointed for several combination of piece types and initial budgets. We contribute to a better understanding of the computational landscape of Solo-Chess by closing two problems left open in [Aravind et al., FUN 2022]. Namely, we show that Solo-Chess is hard even when all pieces are restricted to be only rooks with budget exactly 2, or only knights with budget exactly 11.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Mathematics of computing → Combinatorics

Keywords and phrases solo chess, puzzle games, board games, NP-completeness

Digital Object Identifier 10.4230/LIPIcs.FUN.2024.4

1 Introduction

Solo-Chess is a puzzle game available on `chess.com` [9]. The game is played by a single player on a 8×8 chessboard that initially contains an arrangement of chess pieces. All the pieces have the same color but they are otherwise allowed to capture each other following the standard capturing rules of chess. Each move performed by the player is required to be a capture and the goal is that of removing all but one piece from the board. Moreover, each piece has an associated budget that limits the number of captures it can make. More precisely, all the initial budgets are 2, and only pieces with positive budget are allowed to capture by spending one unit of their budget.[1]

The problem of finding a winning sequence of captures has been first studied from the computational point of view in [1], where the authors generalize the chessboard to an arbitrary size and allow each piece to have an arbitrary non-negative initial budget. More precisely, given a set of piece types $P \subseteq \{\text{♟}, \text{♜}, \text{♞}, \text{♝}, \text{♚}, \text{♛}\}$ and a collection of allowed budgets $B \subseteq \mathbb{N}$, we denote by Solo-Chess(P, B) the problem in which all pieces on the board have some type in P and an initial budget in B.[2]

[1] In the `chess.com` version of the game, there is at most one king on the chessboard and, if a king exists, it must be the last remaining piece.

[2] To lighten the notation, we sometimes write Solo-Chess(t, B) instead of Solo-Chess$(\{t\}, B)$.

Aravind et al. [1] focus on instances containing pieces of a single type, and show that Solo-Chess(♜, $\{0,1,2\}$), Solo-Chess(♝, $\{0,1,2\}$), and Solo-Chess(♛, $\{2\}$) are NP-hard, while Solo-Chess(♟, $\{0,1,2\}$) and Solo-Chess({♟, ♜, ♞, ♝, ♚, ♛}, $\{0,1\}$) can be solved in polynomial time. They also consider the Solo-Chess(♜, $\{0,1,2\}$) problem played 1D boards (i.e., boards with a single row/column) and provide a polynomial-time algorithm. Among others, [1] explicitly mentions the following two open problems:

- What is the computational complexity of Solo-Chess(♜, $\{2\}$)?
- What can be said about the complexity of Solo-Chess played by knights alone?

Our results. In this paper we answer these two questions by showing that both the above problems are NP-hard. In particular, Solo-Chess played by only knights remains NP-hard even when all knights have the same constant budget. Formally, we prove the NP-hardness of Solo-Chess(♜, $\{2\}$) and Solo-Chess(♞, $\{11\}$) by providing polynomial-time reductions from (suitable versions of) the vertex cover and Hamiltonian path problems.

An interactive demonstration of our reductions can be found at `https://www.isnphard.com/g/solo-chess/`.

Other related work. The Solo-Chess problem has also been studied in [4], where the special case in which the number of captures of each piece is unrestricted was considered, and it shown that it is polynomial-time solvable whenever $|P| = 1$, while the problem becomes NP-hard for any choice of P with $|P| \geq 2$. The authors also consider the case in which exactly one of the pieces cannot be captured (and hence it must be the last piece of the board in any solution), and all other (capturable) pieces are of the same type (which might or might not coincide with type of the uncapturable piece). Almost all possible combinations of types are shown to be either NP-complete or polynomial-time solvable.

Aravind et al. [1], also study a variant of Solo-Chess that is played on a graph, where vertices represent pieces, and edges represent the available captures. Solo-Chess is also close in spirit to other problem that require capturing pieces in order to clear a board, such as peg solitaire and its variants [11, 8, 3].

Finally, we mention that the classical 2-player chess game is known to be EXPTIME-complete or PSPACE-complete depending on whether the number of allowed moves is upper-bounded by a polynomial [5, 10].

Structure of the paper and notation. The NP-Hardness of Solo-Chess(♞, $\{11\}$) is discussed in Section 2, while Solo-Chess(♜, $\{2\}$) is considered in Section 3. Throughout the paper, we use the notation $p \to p'$ to denote a move in which piece p captures piece p'. Sometimes, it will be more convenient to refer to the squares occupied by the pieces instead. If p and p' are on squares q and q', respectively, all of the following will also denote move $p \to p'$: (i) $p \to q'$, (ii) $q \to p'$, and (iii) $q \to q'$. We also shorten a sequence of k consecutive captures of the form $p_1 \to p_2$, $p_2, \to p_3$, $\ldots$, $p_{k-1} \to p_k$ by simply writing $p_1 \to p_2 \to p_3 \to \ldots p_{k-1} \to p_k$.

2 Solo-Chess with only knights

In this section we establish the NP-hardness of Solo-Chess(♞, $\{11\}$). In order to do so, we first show that the more general Solo-Chess(♞, $\{0,2,11\}$) problem is NP-hard, and then, in Section 2.3, we argue that knights with budgets 0 and 2 can be simulated by only using knights with budget 11. In the rest of this section, we refer to a knight with budget b as a b-knight.

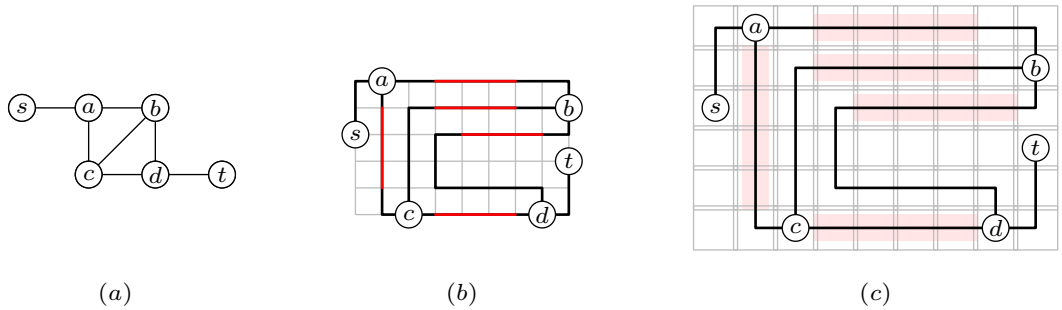


Figure 1 (a) And instance G of RHP. (b) The corresponding planar orthogonal grid drawing $\mathcal{D}$ of G, where the segments $\ell'_{u,v}$ are shown in red. (c) A sketch of the chessboard obtained from $\mathcal{D}$, where the tiles corresponding to the segments $\ell'_{u,v}$ are highlighted with a red band along the direction of the segment.

Our reduction to Solo-Chess(♞, $\{0, 2, 11\}$) is from a restriction of the Hamiltonian path problem to a suitable class of input graphs, which we name RHP. The Hamiltonian path problem is a well-known NP-hard problem that asks to decide whether a given input graph $G = (V, E)$ contains a simple path that traverses all vertices in V. In RHP we restrict ourselves to instances in which G is planar, V contains exactly two vertices s, t with degree 1, and all other vertices have degree 3 (see Figure 1 (a)). We call s the *start vertex* and t the *end vertex* of G. Clearly, G contains a Hamiltonian path iff it contains a Hamiltonian path from its start to its end vertex.

Notice that the RHP problem is NP-hard. Indeed, [7] shows that the Hamiltonian cycle problem is NP-hard when the instances are restricted to planar cubic graphs (i.e., graphs in which every vertex has degree exactly three) by providing a reduction from the 3-SAT problem. A closer inspection of such a reduction shows that the resulting graphs G' contain some special edge $e = (u, v)$ such that if G' admits a Hamiltonian path P then P must traverse e.[3] We can then obtain an instance G of RHP by deleting e from G' and replacing it with two new vertices s, t along with the edges (s, u) and (u, t).

The high-level idea of our reduction is that of embedding the graph G on a chessboard. Since all vertices in $V \setminus \{s, t\}$ have degree 3, finding a Hamiltonian path in G can be thought of as the problem of deleting exactly one edge incident to each vertex of degree 3. To achieve this, each "edge" in our reduction will be equipped with a suitable edge deletion gadget. Once the selected edges have been deleted, the rest of the board encodes the sought Hamiltonian path, but it still contains some knights that need be cleared along the Hamiltonian path. This will be done by tracing the Hamiltonian path using a knight that is initially placed in s, while capturing all remaining knights along the way. Actually, in order to keep the budgets small, this traversal will not be performed by a single knight but rather by a collection of knights that use suitable gadgets as relay stations.

We now describe the technical details of our reduction to Solo-Chess(♞, $\{0, 2, 11\}$): we start by finding a planar orthogonal grid drawing $\mathcal{D}$ of G, i.e., a mapping that associates each vertex $v \in V$ to a distinct point p_v having integer coordinates, and each edge $(u, v) \in E$ to a non self-intersecting polyline $\ell_{u,v}$ connecting p_u to p_v and consisting of the union of alternating (and non-empty) horizontal and vertical segments such that (i) all the segments' endpoints

[3] Such an edge can be found in polynomial-time. In fact, the reduction of [7] even uses a special gadget for the exact purpose of forcing an edge to be in all Hamiltonian paths of G'.

are at integer coordinates, (ii) $\ell_{u,v}$ does not contain any point p_w with $w \in V \setminus \{u, v\}$, and (iii) the polylines of different edges do not intersect (except possibly at their endpoints). Suppose w.l.o.g. that the x-coordinates (resp. y-coordinates) used by the drawing range from 1 to w (resp. from 1 to h). A drawing $\mathcal{D}$ with $w \cdot h = O(n^2)$ can be found in polynomial time w.r.t. $|V|$ (see [12] and the references in Section 5.3 of [2]). We further assume that each polyline $\ell_{u,v}$ between two distinct vertices $u, v \in V \setminus \{s, t\}$ contains at least one horizontal or vertical segment ς with a length of at least 5,[4] and we choose a contiguous portion $\ell'_{u,v}$ of ς that has length 3, starts and ends at integer coordinates, and does not include the endpoints of ς (see Figure 1 (b)).

The chessboard of our instance of Solo-Chess(♞, $\{0, 2, 11\}$) has size $(14h+1)\times(14w+1)$ and consists of $h \times w$ *tiles*, i.e., contiguous sub-chessboards of size 15×15, such that each tile corresponds to a point at integer coordinates in $\mathcal{D}$ and any two horizontally or vertically adjacent tiles share 15 squares along their common edge (see Figure 1 (c)). Each of these tiles is either *empty* or it hosts a (portion of) some *gadget*. Gadgets are arrangements of knights which span an integral number of connected tiles. We will make use of six *gadget types*, five of which span exactly one tile, while the remaining one spans 4 consecutive tiles (either horizontally or vertically). More precisely, we use a *start gadget* and an *end gadget* for the tiles corresponding to s and t, respectively; a *straight edge gadget* for each tile corresponding to a point that lies on some polyline $\ell_{u,v}$ but is neither in $\{p_u, p_v\}$ nor on $\ell'_{u,v}$ (if any); a *corner edge gadget* for each tile corresponding to an endpoint of a segment of some polyline $\ell_{u,v}$, except for the endpoints p_u, p_v of $\ell_{u,v}$ itself; and a *cubic vertex gadget* for each tile corresponding to a point p_v for $v \in V \setminus s, t$. The sixth and final gadget type is the *edge deletion gadget*. We use an edge deletion gadget for each edge $(u, v) \in E$ with $u, v \in V \setminus s, t$ and we place it on the four tiles corresponding to the points of integer coordinates in $\ell'_{u,v}$.

Our gadgets will interact with one another by sharing 0-knights placed on their perimeter. The squares hosting these knights are marked with a × symbol in our figures and will be referred to as *input/output (I/O) squares*. An I/O square q of a gadget acts as an input if some $(b+1)$-knight that is not in one of the gadget's squares captures the 0-knight originally placed on q. In this case, we say that the gadget *takes a b-knight as an input*. An I/O square q of a gadget acts as an output whenever some b-knight that is in one of the gadget's squares captures a knight on q. In such a case we say that the gadget *outputs* a $(b-1)$-knight. The gadgets are designed to ensure that, in any winning sequence of moves, no I/O square can be the target of two distinct captures, and hence it cannot act as both an input and as an output (although it might play different roles in different winning sequences).

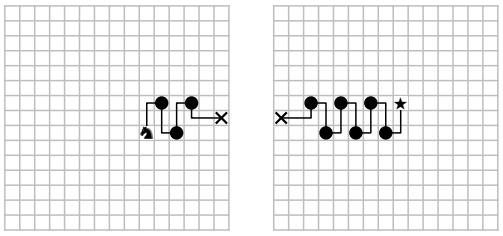
In the following, all squares marked with ●, ○, ★, or × contain 0-knights, the ones marked with ▪ contain 2-knights, and the ones marked with ♞ contain 11-knights. We say that a b-knight is *lively* if $b \geq 6$ and *lazy* if $b \leq 5$. We now discuss our gadgets.

Start and end gadgets

The start and end gadgets are shown in Figure 2 (a) and Figure 2 (b), respectively, and have a single I/O square each. The start gadget corresponds to vertex s and is meant to output a single 7-knight (and no b-knight with $b > 7$ can be output).

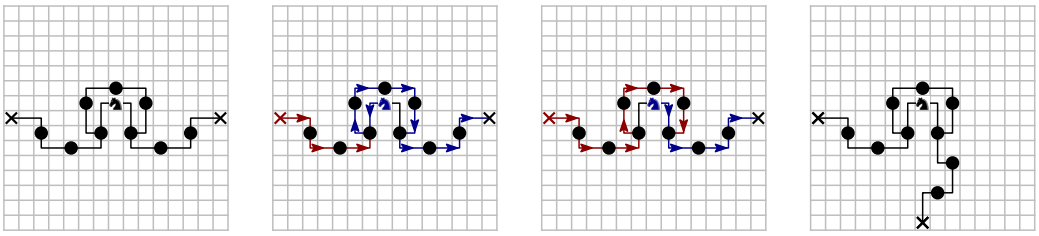
The end gadget corresponds to vertex t and is meant to be played at the end of any winning sequence. Since the knight initially at ★ has budget 0 and can only be captured from exactly one of the ● squares, any winning sequence must necessarily place the last remaining knight on ★, which we name the *goal square*. It is possible to clear all but a single night from the end gadget iff the gadgets takes a b-knight with $b \geq 7$ as input.

[4] This can always be guaranteed, e.g., by "scaling up" the drawing by a factor of 5.



(a) (b)

Figure 2 (a) The arrangement of knights in the start gadget. (b) The arrangement of knights in the end gadget.



(a) (b) (c) (d)

Figure 3 (a) The arrangement of knights in a straight edge gadget. (b) A sequence of moves that allows the gadget to output a 3-knight from the right I/O square when a 3-knight is input in the left I/O square. Red moves are played before blue moves. (c) A sequence of moves that allows the gadget to output a 7-knight from the right I/O square when a 7-knight is input in the left I/O square. (d) The arrangement of knights in a corner edge gadget.

Straight edge gadgets

Each of these gadgets corresponds to a portion of either a horizontal or a vertical segment of some polyline $\ell_{u,v}$ connecting p_u to p_v in $\mathcal{D}$, as long as such portion does not lie in $\ell'_{u,v}$ (which is handled by the edge deletion gadget). In the following we discuss how knights are arranged in the case of a horizontal segment. The vertical case is obtained by rotating the discussed gadget by 90 degrees (either clockwise or counterclockwise).

The arrangement of knights in the gadget is shown in Figure 3 (a). If a b-knight with $b \geq 3$ is input in one of the I/O squares, then it is possible to output a 3-knight from the opposite I/O square (see Figure 3 (b)). Similarly, when a 7-knight is input in one of the I/O squares, it is possible to output a 7-knight from the opposite I/O square (see Figure 3 (c)) but it is not possible to output any b-knight with $b > 7$.

Moreover, if a lazy knight is input in one of the I/O squares, there exists no winning sequence of captures that allows the gadget to output a lively knight. Finally, it is impossible for any sequence of moves to use both I/O locations as outputs, or for any winning sequence of moves to use both I/O locations as inputs (since this would isolate some knight in the gadget from the goal square).

Essentially, this gadget allows to "teleport" either a 3-knight or a 7-knight from an I/O square to the opposite one, while clearing all but the latter square. By chaining together multiple straight edge gadgets it is possible to move a 3-knight or a 7-knight across any horizontal or vertical segment of a polyline.

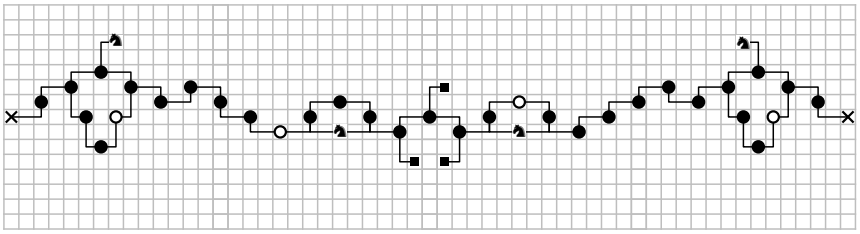


Figure 4 The arrangement of knights in an edge deletion gadget.

Corner edge gadgets

A corner edge gadget allows to connect an horizontal segment of a polyline to an adjacent vertical segment, or vice-versa, and is shown in Figure 3 (d). We only discuss one possible orientation of the gadget, as the others are obtained by a 90-, 180-, or 270-degree rotation. The gadget is almost identical to the straight edge gadget of Figure 3 (a), as the only differences are the positions of a square marked ●, and that of the rightmost I/O square which has been relocated to the bottom edge of the gadget. Neither of these changes affects the threat relationships between the pieces, hence our discussion of the straight edge gadgets also applies to corner edge gadgets.

By chaining together a combination of straight edge and corner edges gadgets it is possible to move a 3-knight or a 7-knight across any portion of a polyline.

Edge deletion gadgets

We use exactly one edge deletion gadget per edge $(u, v) \in E$ with $u, v \notin \{s, t\}$. Such a gadget spans 4 consecutive tiles either horizontally or vertically (i.e., a 15×57 or 57×15 sub-chessboard) and is placed in the tiles corresponding to the portion $\ell'_{u,v}$ of the polyline $\ell_{u,v}$.

As for the straight edge gadget, in the following we only discuss the horizontal version of the gadget, shown in Figure 4, since the vertical version can be obtained by a 90-degree rotation.

The gadget has two I/O squares on opposite sides and there are two intended ways to play the gadget, which are shown in Figure 5, and we name them *traversal mode* and *deletion mode*. In traversal mode a lively knight is input in one of the I/O squares and a 7-knight is output from the opposite I/O square. In deletion mode each of the two I/O squares outputs a 3-knight.

No winning sequence of moves can use both I/O squares as inputs (since this would isolate some knight in the gadget from the goal square). Moreover, if a lazy knight is input in one I/O square, then it is impossible for a winning sequence to output a lively knight from the opposite I/O square. To see this, let k_1, k_2, k_3, k_4 and q_1, q_2, q_3, q_4 respectively be the 11-knights and the squares marked with ○ in Figure 4, from left to right. Assume w.l.o.g.[5] that a lively knight is output by the rightmost I/O square q^*, and notice that this implies that the 0-knight on q^* is captured by k_4, which cannot clear q_4. Then, q_4 must be cleared by k_3 and, in turn, q_3 must be cleared by k_2, and q_2 must be cleared by k_1. This means the input knight k on the leftmost I/O square must clear q_1, i.e., k must have a budget of at least 6.

[5] A symmetric argument holds when the I/O square used as an output is the leftmost one (once k_i is renamed in k_{5-i} and the appropriate symmetric squares for $q_1, \ldots, q_4$ are chosen).

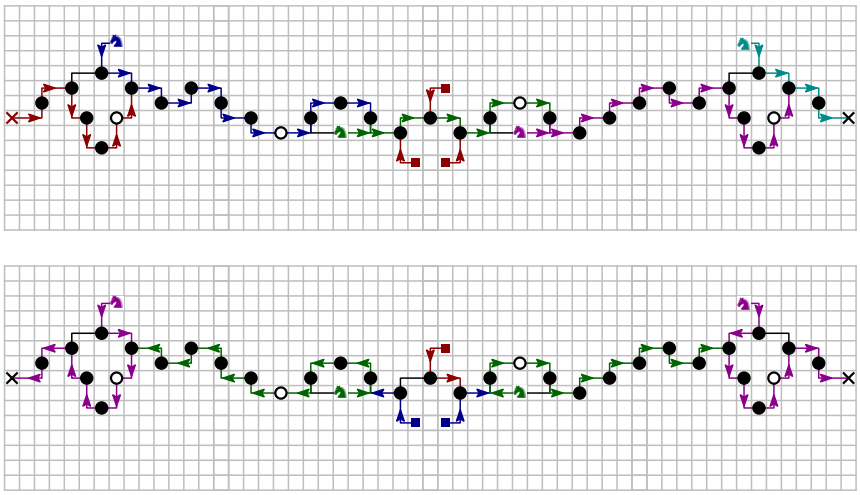


Figure 5 Top: the sequence of moves played when an edge deletion gadget is used in traversal mode in order to output a 7-knight from the right I/O square when a lively knight is input in the left I/O square. Bottom: the sequence of moves played when an edge deletion gadget is used in deletion mode in order to output a 3-knight from each of the two I/O squares. In both figures the order of the moves is: red, blue, green, purple, and teal (if any).

Cubic vertex gadgets

Each vertex $v \in V \setminus \{s, t\}$ corresponds to a tile (namely, the tile associated to coordinates p_v) which contains (a suitable rotation of) the cubic edge gadget of Figure 6 (a). This gadget has 3 I/O squares, each of which corresponds to a distinct edge incident to v.

In the intended operation of the gadget, any two I/O squares are used as inputs while the remaining one is an output. In details, if the gadget takes two knights with budgets at least 3 and 7 as input then it can be used to output a 7-knight. Figures 6 (b)–(d) show how this can be done for any combination of the intended input/outputs, up to symmetries.

No winning sequence of moves can use all the I/O squares as inputs. Moreover, if the gadget outputs a lively knight then it must take at least one lively knight as input. Indeed, the lively knight output from the gadget must necessarily be the only 11-knight k initially placed in the gadget itself (notice that any input knight placed on some I/O square must perform at least 6 captures to reach another I/O square), which implies that k cannot clear the square q marked with ○. Hence, q must be cleared by some input knight k', but this requires k' to perform at least 6 captures.

A similar reasoning shows that, in any winning sequence moves, the gadget cannot output two ore more lively knights. Indeed, for this to happen, there needs to be an output lively knight $k' \neq k$, which must necessarily be also an input knight. Then, (a) k must be the other lively output knight and, since k' cannot clear q (as this would require at least 6 captures, resulting in a budget of at most 5, (b) k must clear q. However, it is impossible for both (a) and (b) to happen.

Finally, we point out that it is possible to play the gadget in the following unintended way: whenever two knights with budget at least 3 as used as inputs, a 3-knight can be output form the remaining I/O square. However, as we argue in more details later, our edge deletion gadgets ensures that doing so always results in a losing configuration.

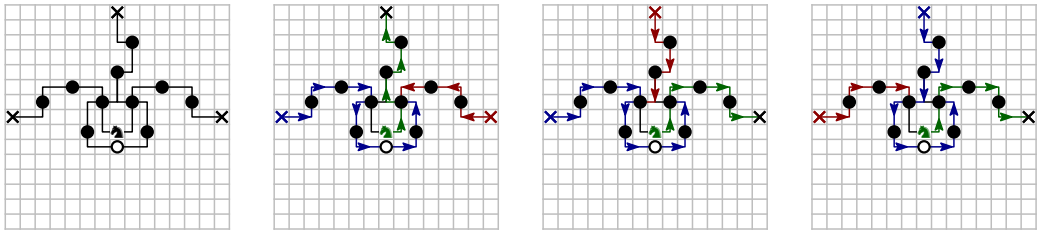


Figure 6 (a) The arrangement of knights in a cubic vertex gadget. (b)–(d) each show a sequence of moves that outputs a 7-knight from one of the I/O squares when two knights having budgets at least 3 and 7 are input in the other two I/O squares. Red moves are performed fist, followed by blue moves, and then by green moves.

2.1 One direction

Let P be a Hamiltonian path of $G = (V, E)$ that starts in s and ends in t. Let E_P be the edges in P and $\overline{E}_P = E \setminus E_P$, and recall that the edges connecting s and t to their sole neighbors must belong to E_P. The following sequence of moves wins the instance of Solo-Chess(♞, $\{0, 2, 11\}$). For each edge $(u, v) \in \overline{E}_p$ we play the edge deletion gadget $g_{u,v}$ corresponding to (u, v) in deletion mode in order to output two 3-knights: one on the u-side and one on the v-side of (u, v); then, we play all the straight edge and corner edge gadgets that connect $g_{u,v}$ to u (resp. v), in this order, which has the effect of placing a 3-knight on the input of the cubic vertex gadget corresponding to u (resp. v) while clearing all other squares of the played gadgets. After this step, the only unplayed gadgets are (i) the start and end gadgets, (ii) the cubic vertex gadgets corresponding to the vertices in V, and (iii) the straight edge, corner edge, or edge deletion gadgets corresponding to the edges in E_P. We can play all these gadgets in the same order as vertices and edges are encountered in P: we start by outputting a 7-knight from the start gadget, then we play the sequence of straight/corner edge gadgets (possibly none) until we reach the cubic vertex gadget g_u corresponding to the vertex u that follows s in P. This brings a 7-knight to one of the I/O squares of g_u, while a 3-knight was already on some other I/O square. We use the 3-knight and the 7-knight as inputs for g_u in order to output a 7-knight on the only remaining I/O square of g_u, which corresponds to the edge (u, v) following u in P. We then play the gadgets associated with (u, v) so as to place a 7-knight on an I/O square of the cubic vertex gadget g_v of v. Notice that, in addition to straight/corner edge gadgets, playing the gadgets associated with (u, v) also involve playing a single edge deletion gadget in traversal mode. We repeat this process until a 7-knight is output from the cubic vertex gadget g_z corresponding to the vertex z immediately preceding t in P. Finally, we play the straight/corner edge gadgets associated with (z, t) to place a 7-knight on the input of the end gadget which, at this point, is the only gadget containing non-empty squares. To complete the winning sequence it suffices to play the end gadget using the input 7-knight.

2.2 The other direction

Fix a winning sequence σ and consider a gadget g that is not the start gadget. In order for g to output a lively knight from some I/O square in σ, it is necessary for g to receive a lively knight as input from another I/O square. Moreover, in σ, no gadget can output more than one lively knight and the end gadget must take a lively knight as input.

We define a directed graph H_σ whose vertex set is the set of gadgets in our instance of Solo-Chess(♞, $\{0, 2, 11\}$) and such that H_σ contains a directed edge (g, g') iff there a move that causes a lively knight to be output from g and be input into g'. The previous observations imply that the out-degree of all vertices of H_σ is at most 1, that all vertices with out-degree 1 have in-degree at least 1 except for the start gadget, and that the in-degree of the end gadget is 1.

Then, H_σ must contain a path P from the start gadget to the end gadget. Moreover, P must traverse all cubic vertex gadgets. Indeed, suppose towards a contradiction that there is some cubic vertex gadget g, corresponding to a vertex $u \in V$, that is not in P.

If g does not take any lively knight as input then the only way to clear all squares of g results in g outputting a lazy knight from an I/O square associated with some edge $(u, v) \in E$, where $v \notin \{s, t\}$.[6] Then, our instance of Solo-Chess(♞, $\{0, 2, 11\}$) has an edge deletion gadget g' associated with (u, v). Since the I/O square q on "u's side" of g' cannot be used as input in traversal mode (as no lively knight can be input from q), g' must output a knight on q and this causes one or mode knights placed on the (straight or corner) edge gadgets used to encode the portion of the polyline $\ell_{u,v}$ between p_u and $\ell'_{u,v}$ to become disconnected from the goal knight, a contradiction.

Otherwise g takes a lively knight as input, which means that there must exist a path P' from the start gadget to g in H_σ. Let g' be the last vertex of P' that is also in P, and notice that g' must have out-degree at least 2 in H_σ, i.e., it must output at least two lively knights, a contradiction.

If a graph obtained from G by performing edge subdivisions[7] contains a simple path spanning V, then G contains a Hamiltonian path. Let G' be the graph obtained from G by subdividing each edge (u, v) into a path containing as many internal nodes as the number of gadgets placed in the tiles corresponding to the internal points of the polyline $\ell_{u,v}$ (that is, $|\ell_{u,v}| - 1$ if $u \in \{s, t\}$ or $v \in \{s, t\}$, and $|\ell_{u,v}| - 4$ if $u, v \notin \{s, t\}$). There is an injective homomorphism between the undirected version of H_σ and G' such that each start, end, and cubic vertex gadget in H_σ is mapped to the corresponding vertex in G'. Since P is a (simple) path that spans all start, end, and cubic vertex gadgets in H_σ, there is some (simple) path in G' that spans all vertices in V, hence G contains a Hamiltonian path.

2.3 Uniform budgets

Given a configuration c and a knight k, we denote by $\tau_c(k)$ the number of knights in c that are threatened by k.

We start by proving the following two lemmas which provide some "local" rules that allow us to perform some captures without compromising the solvability the configuration.

▶ **Lemma 1.** *Let C be a configuration containing a b-knight k that threatens only a single knight k'. Assume further that either (i) $b = 1$, or (ii) $b \geq 2$ and $\tau_c(k') = 2$. If C is solvable then it admits a wining sequence of moves that starts with $k \to k'$.*

Proof. Let q and q' be the squares containing k and k' in C, respectively, and let $\sigma = \langle m_1, m_2, \dots \rangle$ be any winning sequence of moves for C. We prove the claim by showing that σ can be transformed into a winning sequence that starts with $k \to k'$.

[6] The only I/O square of the start gadget must be used as output, and the only I/O square of the end gadget must take lively knight as input.

[7] The subdivision of an edge (u, v) into a path with $\ell \geq 1$ internal nodes consists in inserting of the new vertices $w_1, w_2, \dots, w_\ell$, deleting (u, v), an adding the edges in $\{(u, w_1), w(w_\ell, v)\} \cup \{(w_i, w_{i+1}) \mid i = 1, \dots, \ell - 1\}$.

We start by ensuring that σ contains a move of the form $k \to q'$. If this is not already the case, then it must contain $q' \to k$ and such a capture must necessarily be the last move of σ (since $q' \to k$ clears square q' and isolates the knight on q from any other knight in the resulting configuration). Then, replacing $q' \to k$ with $k \to q'$ also results in winning sequence of moves.

Let i be the index of move $k \to q'$ in σ. We now argue that, if m_i is not already the first move, then swapping it with its preceding move m_{i-1} still results into a winning sequence. The claim follows by iteratively performing such a swap until $k \to q'$ becomes the first move. If m does not involve square q nor square q' then performing either of $\langle m_1, \ldots, m_{i-1}, m_i \rangle$ and $\langle m_1, \ldots, m_{i-2}, m_i, m_{i-1} \rangle$ from c results in the same configuration. Otherwise, since m_{i-1} cannot clear q', it must necessarily be of the form $k'' \to q'$ for some knight $k'' \neq k$. If (i) holds, then the configurations obtained from C by performing $\langle m_1, \ldots, m_{i-1}, m_i \rangle$ and $\langle m_1, \ldots, m_{i-2}, m_i, m_{i-1} \rangle$ are identical except, possibly, for the budget of the knight on q' which is *exactly* 0 in the former case and *at least* 0 in the latter. If (ii) holds then, after move m_{i-1}, the only remaining knights are on squares q and q' and m_i is the last move of σ. Then moving m_i immediately before m_{i-1} also results in a winning sequence. ◀

A repeated application of Lemma 1 shows that a knight k_0 that either has budget 0, or has $\tau_c(k_0) = 1$ and a budget in $\{1, \ldots, 10\}$, can be simulated by setting the budget of k_0 to 11 and adding a *retinue* of $\eta = 11 - b$ additional 11-knights $k_1, \ldots, k_\eta$ such that each k_i with $i = 1, \ldots, \eta - 1$ threatens only k_{i-1} and k_{i+1}, while k_η threatens only $k_{\eta-1}$.

To show that Solo-Chess(♞, {11}) is NP-hard, we adapt our reduction of Section 2 as follows: instead of using tiles of size 15×15 we use *super-tiles* of size 57×57 and corresponding *super-gadgets*. Each super-tile can be thought of as a 5×5 grid of (regular) tiles. We first discuss how the start, end, straight edge, corner edge, and cubic vertex gadgets can be turned into super-gadgets. Each such super-gadget is obtained by first placing the corresponding regular gadget g in the center tile of the super-tile, and then using suitable straight edge gadgets to "connect" the I/O squares of g to the perimeter of the super-tile.[8] Finally, we simulate each 0-knight by setting its budget to 11 and adding a retinue of eleven 11-knights as discussed above.[9]

The super gadget g^* corresponding to the edge-deletion gadget spans 4 super-tiles. To obtain the horizontal version of g, we arbitrarily choose one of these super-tiles t and we place knights in the other three super tiles as in the horizontal version of the super straight edge gadget.[10] We arrange the knights in t as follows: first we place an edge deletion gadget g spanning 4 of the 5 tiles in the middle row of t, so that one of the I/O squares of g lies on one side of t; then, we place a straight edge gadget in the missing tile of the middle row in order to "connect" the other I/O square of g to the opposite side of t. Finally, we replace the three 2-knights in g, along with all 0-knights in both g and the straight edge gadget, with 11-knights and we add the corresponding retinues, as discussed above.

All the 11-knights resulting from the above transformations will be entirely contained within the same super-gadget as the original knight. Moreover, none of the additional knights will introduce any inter-cluster threat. In fact, the reason for using super-tiles in place of

[8] Two of the straight edge gadgets used in the super cubic vertex gadgets are rotated by 180 degrees.

[9] This causes the knights placed on the I/O squares belonging to two super-gadgets to have two retinues each (one for each of the two gadget). Although one retinues would suffice, using one retinue per gadget simplifies the description of the reduction.

[10] Defining the super edge deletion gadget in order to only span a single super-tile t would be sufficient to obtain a reduction to Solo-Chess(♞, {11}). We employ four super-tiles in order to re-use the same construction described for the non-uniform case.

regular tiles is having enough free space around the gadgets to fit these new knights. Due to space limitations, the figures showing the resulting super-gadgets are omitted from this manuscript and can be found in the full version of the paper.

We have therefore proved the main results of this section, namely:

▶ **Theorem 2.** Solo-Chess(♞, {11}) *is NP-hard.*

3 Solo-Chess with only rooks

In this section we establish the NP-hardness of Solo-Chess(♜, {2}). Our construction is similar to the one employed by [1] to show the NP-Hardness of Solo-Chess(♜, {0, 1, 2}). Roughly speaking, we would like to simply increase the budget of all rooks to 2, but this allows for some rook capture that were previously forbidden and breaks the reduction. To circumvent this, we are forced to place additional rooks with budget 2, which in turn can perform even more captures. The main technical challenge lies in showing that we force arbitrary winning strategies to follow the intended scheme of the reduction.

We start by proving the NP-hardness of an auxiliary problem named Solo-Chess*(♜, {0, 2}), and then we show (see Section 3.3) how an instance of such a problem can be first transformed into an equivalent instance of Solo-Chess*(♜, {2}) which, in turn, can be converted into another equivalent instance of Solo-Chess(♜, {2}). In the following we will refer to a rook with budget b as a b-rook.

The auxiliary problem Solo-Chess*(♜, {0, 2}) is similar to Solo-Chess(♜, {0, 2}) except for the following two variations:

- we refer to the topmost row (resp. rightmost column) of an instance $\mathcal{I}$ of Solo-Chess*(♜, {0, 2}) as the *goal row* (resp. *goal column*), and to the square on their intersection of the goal row and the goal column as the *goal square*. The goal square must initially contain a special rook called the *goal rook*, and any winning sequence of moves for $\mathcal{I}$ is required to leave the last remaining rook on the goal square;[11]
- the goal column contains no rook other than the goal rook. Similarly, any column containing a 0-rook r, contains no rook other than r.

Our reduction for Solo-Chess*(♜, {0, 2}) is from the (decision version of the) vertex cover problem (VC for short), which a well-known NP-hard problem [6]. The input of VC consists of a graph $G = (V, E)$ and of an integer k, and the goal is that of deciding whether there exists a set $S \subseteq V$ of size at most k such that at least one endvertex of each edge in E lies in S.

We build our instance of Solo-Chess*(♜, {0, 2}) in two steps: first we construct a $(m + 1) \times (n + m)$ chessboard, and then we augment it by inserting some additional columns.

We start by describing the $(m + 1) \times (n + m)$ chessboard. Let $V = \{v_1, \ldots, v_n\}$ and $E = \{e_1, \ldots, e_m\}$. We associate the j-th of the first n columns with vertex v_j. Moreover, we associate the $(i + 1)$-th row with edge e_i, and we refer to the topmost row as the *goal row* and we denote it with γ. To improve readability, we often refer to the squares of the chessboard using the row and column names instead of their integer coordinates, e.g., square (e_3, v_2) is at coordinates $(4, 2)$ and square (γ, v_5) is at coordinates $(1, 5)$. The goal row contains a 2-rook on each of the n columns $v_1, \ldots, v_n$, and for each edge $e_i = (v_h, v_j)$ we place three 2-rooks: an *incidence rook* $r_{i,h}$ on square (e_i, v_h), another *incidence rook* $r_{i,j}$

[11] The budget of the goal rook is irrelevant since any winning sequence of moves cannot clear the goal square.

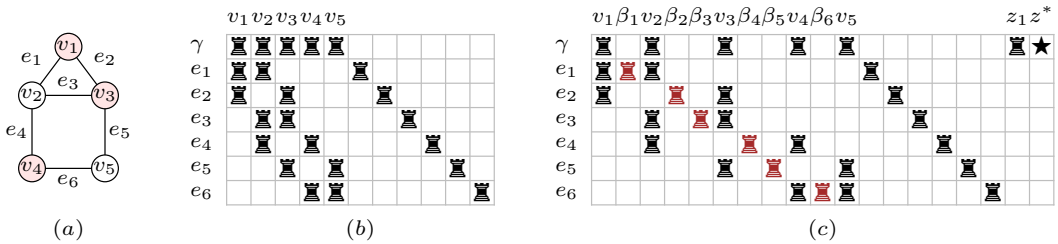


Figure 7 (a) An instance of the (decision version of) vertex cover problem for $k = 3$. The vertices of a possible vertex cover are highlighted in red. (b) The corresponding $(m+1) \times (n+m)$ chessboard constructed in the first step of our reduction. (c) The corresponding instance of Solo-Chess*(♜, $\{0, 2\}$), where 0-rooks are in red and $\Delta = 1$.

on square (e_i, v_j), and a final *collector rook* c_i on row e_i and column $n + i$. Essentially, the sub-chessboard consisting of rows $e_1, \ldots, e_m$ corresponds to the juxtaposition of the (transposed) incidence matrix of G with a $m \times m$ identity matrix, where each non-zero entry corresponds to a 2-rook. See Figure 7 (a) for an example instance of VC and Figure 7 (b) for the corresponding $(m + 1) \times (m + n)$ chessboard.

We now augment the above board. For each edge $e_i = (v_h, v_j)$ we add a new column β_i between v_h and v_j and a *blocker* 0-rook on square (e_i, β_i). Next, let $\Delta = 2k - n$ and add $|\Delta| + 1$ new columns $z_1, \ldots, z_{|\Delta|}, z^*$ to the right of the board. Each of the squares (γ, z_i) for $i = 1, \ldots, |\Delta|$ contains a 2-rook if $\Delta > 0$ and a 0-rook if $\Delta < 0$. Finally, we place the goal rook on square (γ, z^*). See Figure 7 (c) for an example.

3.1 One direction

Before discussing how a vertex cover of G can be turned into a winning sequence of moves for our instance of Solo-Chess*(♜, $\{0, 2\}$), we restate the following characterization from [1] for the solvability of instances on one-dimensional boards. We define the potential of a rook with budget b to be $b - 1$ and the overall potential of a collection of rooks to be the sum of their potentials (where an empty collection of rooks has potential 0). We say that an instance of Solo-Chess(♜, $\mathbb{N}$) is (i, j)-solvable if it can be solved with the additional constraint that the last remaining rook must be placed on square at coordinates (i, j).

▶ **Lemma 3** ([1], reformulated). *Consider an instance of* Solo-Chess(♜, $\mathbb{N}$) *on a board of size* $1 \times n$*, and let* $\phi(j_1, j_2)$ *denote the overall potential of the rooks on columns* $j_1, j_1+1, \ldots, j_2$*. The instance is* $(1, j)$*-solvable iff* j *contains a rook,* $\phi(1, j-1) \geq 0$*, and* $\phi(j+1, n) \geq 0$*.*

If S is a vertex cover of G of size at most k then the following is a winning strategy for our instance of Solo-Chess*(♜, $\{0, 2\}$). For each edge e_i, we choose an endvertex v_h of e_i such that $v_h \in S$, we let v_j be the other endvertex (which might or might not be in S), and we perform the horizontal captures $r_{i,j} \to b_i \to r_{i,h}$ (see Figure 8 (a) and Figure 8 (b)), followed by $c_i \to (e_i, v_h)$ (see Figure 8 (c)). After these captures, each edge row contains only a single 1-rook on a column associated to a vertex in S. Then, for each column $v_i \in S$, we examine all the edge rows e_i in increasing order of i, and for each such row e_i containing a 1-rook in square (e_i, v_i), we perform the vertical capture $(e_i, v_i) \to (\gamma, v_i)$. We are now left with a chessboard where the only non-empty row is the goal row γ (see Figure 8 (d)). In particular, the goal row contains one rook on each column $v_1, \ldots, v_n$ and at most k of these rooks have budget 0 (i.e., those resulting from the previous vertical captures), while the others (not involved in vertical captures) are 2-rooks. The remaining rooks are the $|\Delta|$ rooks on columns

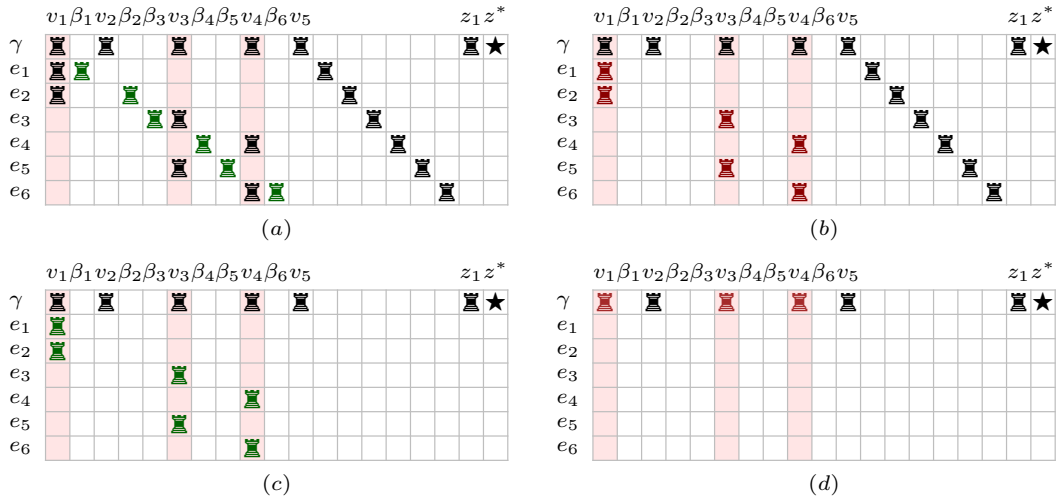


Figure 8 Notable configurations encountered in an winning sequences of moves for the instance of Figure 7 (c). Red columns corresponding to vertices in the vertices in a vertex cover of size 3. 2-rooks, 1-rooks, and 0-rooks are shown in black, greens, and red, respectively. (a) The configuration after all blocker rooks have been captured. (b) The configuration after all the squares initially containing the blocker rooks have been cleared. (c) The configuration after all collector rooks capture some rook on a red column. (d) The configuration after all squares in non-goal rows have been cleared. A winning sequence of moves from configuration (d) is $(\gamma, v_2) \to (\gamma, v_1) \to (\gamma, v_3)$, $(\gamma, v_5) \to (\gamma, v_4) \to (\gamma, v_3)$, $(\gamma, z_1) \to (\gamma, v_3) \to (\gamma, z^*)$.

$z_1, \ldots, z_{|\Delta|}$, with overall potential $\Delta = 2k - n$, and the goal rook (on the rightmost column). Hence the sum of the potentials of all non-goal rooks is at least $0 \cdot k + (n-k) \cdot 2 - n + \Delta \geq 0$ and Lemma 3 implies that this configuration is (γ, z^*)-solvable.

3.2 The other direction

Here we show that a winning sequence of moves for our instance of Solo-Chess*(♜, $\{0, 2\}$) implies the existence of a vertex cover of G of size at most k.

We start by introducing the notions of *depleted row* and *disconnected configuration*, and we argue that any sequence of moves that results in a configuration that is either disconnected or creates a depleted row cannot be winning.

A row is *depleted* if it is not γ, it contains only a single 0-rook, and it contains no 1-rooks or 2-rooks. A configuration C is *disconnected* if the graph whose nodes are non-empty squares in C, and such that two distinct squares are linked by an edge iff they share the same row or the same column, is disconnected. It is immediate to verify that no disconnected configuration is solvable.

▶ **Lemma 4.** *Let $\sigma = \langle m_1, m_2, \ldots \rangle$ be a winning sequence of moves. All configurations encountered during σ contain no depleted row.*

Proof. Let C_ℓ be the configuration obtained after performing the first ℓ moves of σ. Suppose towards a contradiction that some configuration C_h contains some depleted row e_i, and that all $C_{h'}$ with $h' > h$ contain no depleted rows. Let (e_i, v_j) be the square containing the unique 0-rook on row e_i in C_h.[12]

[12] The 0-rook in e_i must necessarily be on a column v_j where v_j is an endvertex v_j, since otherwise C_h would be disconnected.

Since (e_i, v_j) must eventually be cleared by σ, there is some configuration $C_{h'}$ with $h' \geq h$ and some 2-rook r in $C_{h'}$ such that move $m_{h'+1}$ is the capture $r \to (e_i, v_j)$. Since e_i contains no 2-rooks in $C_{h'}$, $r \to (e_i, v_j)$ must be a vertical capture. We split the proof depending on whether r is on the goal row or in some row $e_{i'}$ (with $e_{i'} \in E \setminus \{e_i\}$) in $C_{h'}$.

Suppose that r is in row $e_{i'}$ in $C_{h'}$ and focus on row $e_{i'}$ in $C_{h'+1}$ (whose square $(e_{i'}, v_j)$ is empty), which falls in one of the following three cases:

- If $(e_{i'}, \beta_{i'})$ contains a 0-rook in $C_{h'+1}$ then, among moves $m_{h'+2}, m_{h'+3}, \ldots$, there is some 2-rook $r' \neq r$ on $e_{i'}$ that first captures $(e_{i'}, \beta_{i'})$ and then captures the only other remaining rook on $e_{i'}$. This results in a configuration $C_{h''}$ with $h'' > h' \geq h$ where $e_{i'}$ is depleted, which is a contradiction.
- If $(e_{i'}, \beta_{i'})$ contains a 1-rook in $C_{h'+1}$ then there is some 2-rook $r' \neq r$ on $e_{i'}$ such that some move among $m_1, \ldots, m_{h'}$ is the capture $r' \to (e_{i'}, \beta_{i'})$ and some move among $m_{h'+2}, m_{h'+3}, \ldots$ consists of r' capturing the only remaining rook on $e_{i'}$. Hence there is a configuration $C_{h''}$ with $h'' > h' \geq h$ where $e_{i'}$ is depleted, which is a contradiction.
- If $(e_{i'}, \beta_{i'})$ is empty in $C_{h'+1}$ then, in moves $m_1, \ldots, m_{h'}$, the blocker $c_{i'}$ must have been captured by some 2-rook r' on $e_{i'}$ which then captured some other rook r'' on row $e_{i'}$ other than r'. This can only happen if r, r', and r'' are the leftmost incidence rook, the rightmost incidence rook, and the collector of row $e_{i'}$, respectively. Then, $e_{i'}$ is depleted in $C_{h'+1}$ since its only rook is a 0-rook in square $(e_{i'}, \beta_{i'})$, and this provides the sought contradiction.

Suppose now that r is on the goal row in $C_{h'}$. Then, in $C_{h'+1}$, r is a 1-rook on (e_i, v_j) and either r captures a rook on column v_j, or there must be some 2-rook that first captures r and then captures some other rook on column v_j. In any case, at least one move among $m_{h'+1}, m_{h'+2}, \ldots$ is a vertical capture performed by a 1-rook on column v_j. Let $m_{h''}$ be the last such move and let $(e_{i'}, v_j)$ be its target square, where $e_{i'} \in E$, so that $(e_{i'}, v_j)$ contains a 0-rook in $C_{h''}$.

We describe the state of row $e_{i'}$ with a 4-tuple $t \in \{0, 1, 2, \square, ?\}^4$ whose entries represent the contents of the left incidence square of $e_{i'}$, $(e_{i'}, \beta_{i'})$, the right incidence square of $e_{i'}$, and the collector square of $e_{i'}$, in this order. More precisely, 0, 1, and 2 respectively denote a 0-rook, a 1-rook, and a 2-rook, $\square$ denotes an empty square, and ? denotes any of the above. Moreover, we underline the entry corresponding to the square on column v_j.

- If $(e_{i'}, \beta_{i'})$ is empty in $C_{h''}$, then the state of $e_{i'}$ must be $(\underline{0}, \square, \square, ?)$ or $(\square, \square, \underline{0}, ?)$. In any case, some move $m_{h'''}$ with $h''' > h''$ clears square $(e_{i'}, v_j)$. Since this cannot be a horizontal move (as it would result in $e_{i'}$ being depleted), it must be a vertical move (of a 1-rook), which contradicts our choice of h''.
- If $(e_{i'}, \beta_{i'})$ contains a 0-rook in $C_{h''}$, then the state of $e_{i'}$ must be $(\underline{0}, 0, ?, ?)$ or $(?, 0, \underline{0}, ?)$. Since $(e_{i'}, \beta_{i'})$ must be cleared by a 2-rook (on row $e_{i'}$) that first captures $(e_{i'}, \beta_{i'})$ and then captures another rook on $e_{i'}$, the state of $e_{i'}$ resulting from this latter capture is one of (a) $(\underline{0}, \square, \square, ?)$, (b) $(\underline{?}, \square, \square, 0)$, (c) $(\square, \square, \underline{0}, ?)$, and (d) $(0, \square, \underline{\square}, \square)$. However (a) and (c) lead to a contradiction by using analogous arguments to the ones of the previous case, (d) implies that $e_{i'}$ is depleted in some configuration $C_{h'''}$ with $h''' > h'' > h' \geq h$, and in (b) row $e_{i'}$ cannot be cleared since capturing the 0-rook on the cleaner square results in a disconnected configuration.
- If $(e_{i'}, \beta_{i'})$ contains a 1-rook r' in $C_{h''}$, then the state of $e_{i'}$ must be either $(\underline{0}, 1, \square, ?)$ or $(\square, 1, \underline{0}, ?)$. Either r' captures some other rook in $e_{i'}$, in which case the resulting state of $e_{i'}$ is one of $(\underline{0}, \square, \square, ?)$, $(\underline{?}, \square, \square, 0)$, and $(\square, \square, \underline{0}, ?)$, thus the same arguments as above apply, or some 2-rook (on row $e_{i'}$) first captures on $(e_{i'}, \beta_{i'})$ and then captures another rook on $e_{i'}$, leaving $e_{i'}$ in state $(\underline{0}, \square, \square, \square)$ which corresponds to a depleted row. ◀

We now consider an arbitrary winning sequence of moves and we perform two consecutive transformations, each of which will result in another winning sequence that follows some desirable pattern of moves and is easier to analyze.

First transformation

To perform our first transformation, we observe that each square (e_i, β_i) that initially contains a 0-rook and must be cleared, which means that there must be some 2-rook r that performs the capture $r \to (e_i, \beta_i)$. Moreover, r must be on row e_i and cannot be the collector c_i, since, after capture $c_i \to (e_i, \beta_i)$, either the only non-empty square on row e_i is (e_i, β_i), or there are two non-empty squares (e_i, β_i) and (e_i, v_j) where v_j is an endvertex of e_i. In the former case (e_i, β_i) is disconnected from the goal square, while in the latter case neither $(e_i, v_j) \to (e_i, \beta_i)$ nor $(e_i, \beta_i) \to (e_i, v_j)$ are possible since they would yield either a disconnected configuration or single 0-rook on row e_i (which is not solvable by Lemma 4). We conclude that r is one of the two incidence rooks on row e_i, which is still in its original square.

We now argue that rearranging the moves so that $r \to (e_i, \beta_i)$ becomes the first capture still results in a winning sequence. Indeed, we can iteratively swap $r \to (e_i, \beta_i)$ with its preceding move m since, if m involved square (e_i, β_i) then the configuration obtained after performing all moves up to $r \to (e_i, \beta_i)$ would be disconnected.

Performing the above rearrangement for each row e_i with $i = 1, \ldots, m$, and executing the first m moves yields a solvable configuration C' in which the goal row is identical to that of the initial configuration, and each row e_i contains its collector c_i, exactly one incidence rook, and a 1-rook on square (e_i, β_i). See Figure 8 (a) for an example.

Second transformation

For the second transformation, consider a generic edge $e_i \in E$ and recall that row e_i contains a single incidence rook $r_{i,j}$ on some square (e_i, v_j) (where v_j is an endvertex of e_i).

We first argue that no winning sequence of moves from C' contains a capture that targets square (e_i, β_i). Indeed, if that were the case, there would also be some 2-rook r that performs the capture $r \to (e_i, \beta_i)$ (since (e_i, β_i) must eventually be cleared). The rook r must be either $r_{i,j}$ or the collector c_i. In the former case, the configuration resulting from the move $r_{i,j} \to (e_i, \beta_i)$ is disconnected. In the latter case, immediately after $c_i \to (e_i, \beta_i)$, row e_i contains a 1-rook on (e_i, β_i) and possibly another rook on (e_i, v_j), hence the only possible moves result in either a disconnected configuration or in a single 0-rook on row e_i.

Since (e_i, β_i) must be cleared and the rook r on (e_i, β_i) is never captured, the sequence must include the move $r \to (e_i, v_j)$ (the only other option is $r \to c_i$ which results in a configuration where (e_i, β_i) cannot be cleared). Similarly to the previous transformation, we now argue that $r \to (e_i, v_j)$ can be performed as the first move of a winning sequence by iteratively swapping it with the previous move m. Indeed, if m targets (e_i, v_i) then the configurations obtained by (i) performing all the moves up to $r \to (e_i, v_j)$ and (ii) swapping $r \to (e_i, v_j)$ with m and then performing all the moves up to m, are identical except possibly for the budget of the rook in (e_i, v_j) which is 0 in the former case and *at least* 0 in the latter.

Performing the above rearrangement for each row e_i with $i = 1, \ldots, m$, and executing the first m moves yields a solvable configuration C'' in which the goal row is identical to that of initial configuration, and each row e_i contains its collector c_i and exactly one 0-rook on some square (e_i, v_j) where v_j is an endvertex of e_i. As a consequence the set S containing all vertices $v_j \in V$ such that there exists at least one row e_i for which (e_i, v_j) is non-empty is a vertex cover of G. See Figure 8 (b) for an example.

Concluding the proof

We are now ready to conclude the proof, by showing that S has size at most k via a potential argument. More precisely, given a configuration C, we assign a potential $\psi_j(C)$ to each column $j \neq z^*$ as follows:

- $\psi_j(C) = 1$ if (γ, j) contains a 2-rook and there is no other rook on column j;
- $\psi_j(C) = -1$ if (γ, j) is non-empty and either it contains a 0-rook, or there exists some $e_i \in E$ such that (e_i, j) is non-empty in C (possibly both);
- $\psi_j(C) = 0$ in the remaining cases.

We also let $\psi_{z^*}(C) = 0$ and we define the potential $\psi(C)$ of C as the sum of $\psi_j(C)$ over all columns j.

Fix any winning sequence of moves that starts from configuration C'' and let C_ℓ be the configuration resulting from performing the first ℓ moves of the sequence.

▶ **Lemma 5.** *$\psi(C_\ell)$ is non-increasing w.r.t. ℓ.*

Proof. First of all, notice that no rook on row γ ever performing any vertical capture, since this would result in a disconnected configuration, and the same holds for horizontal captures that target a column containing a collector. Of the remaining captures, only those that target a square on the goal row can affect $\psi(\cdot)$. We consider these captures separately depending on whether they are vertical or horizontal and we study how the potential of the affected column(s) changes as a result of the move.

Any vertical capture from some configuration C_ℓ that targets a square (γ, j) results in a 0-rook on square (γ, j) in configuration $C_{\ell+1}$. Therefore $\phi_j(C_\ell) = \phi_j(C_{\ell+1}) = -1$.

Consider now an horizontal capture $(\gamma, j) \to (\gamma, j')$ performed by a b-rook from some configuration C_ℓ and observe that (γ, j) must be the only non-empty square in column j. If $b = 2$ then $\psi_j(C_\ell) = 1$, $\psi_j(C_{\ell+1}) = 0$, $\psi_{j'}(C_\ell) \geq -1$, and $\psi_{j'}(C_{\ell+1}) \leq 0$. If $b = 1$ and $j' \neq z^*$ then $\psi_j(C_\ell) = \psi_j(C_{\ell+1}) = 0$, $\psi_{j'}(C_\ell) \geq -1$ and $\psi_{j'}(C_{\ell+1}) = -1$. Finally, if $b = 1$ and $j' = z^*$ then $\psi_j(C_\ell) = \psi_j(C_{\ell+1}) = \psi_{j'}(C_\ell) = \psi_{j'}(C_{\ell+1}) = 0$. ◀

For the configuration C'' we have $n - |S| + \max\{0, \Delta\}$ columns $j \neq j^*$ with $\psi_j(C'') = 1$ and $|S| + \max\{0, -\Delta\}$ columns $j \neq j^*$ with $\psi_j(C'') = -1$, hence $\psi(C'') = n - 2|S| + \Delta = 2k - 2|S|$. For the final configuration C^* (which contains a single rook in column j^*) we have $\psi(C^*) = 0$. Using Lemma 5 we have $2k - 2|S| = \psi(C'') \geq \psi(C^*) = 0$, which implies $|S| \leq k$.

3.3 Uniform budgets

Here we show that the 0-rooks in the instances of Solo-Chess*(♜, $\{0, 2\}$) resulting from the previous reduction can be simulated with 2-rooks, thus showing that Solo-Chess*(♜, $\{2\}$) is NP-hard, and then we reduce Solo-Chess*(♜, $\{2\}$) to Solo-Chess(♜, $\{2\}$).

Given a configuration C and a rook r, we denote with $\tau_C(r)$ the number of rooks $r' \neq r$ that are on the same row or on the same column as r.

▶ **Lemma 6.** *Let C be a configuration containing a b-rook r such that $\tau_C(r) = 1$ and let r' be the only rook threatened by r. Assume further that neither r nor r' are on the goal square, and that either (i) $b = 1$, or (ii) $b = 2$ and $\tau_C(r') = 2$. If C is solvable then it admits a wining sequence of moves that starts with the capture $r \to r'$.*

Proof. Let q and q' be the squares containing r and r' in C, respectively. Any winning sequence of moves $\sigma = \langle m_1, m_2, \dots \rangle$ for C cannot contain $q' \to r$, since q is not the goal square and hence such a move would result in a disconnected configuration. Then, some

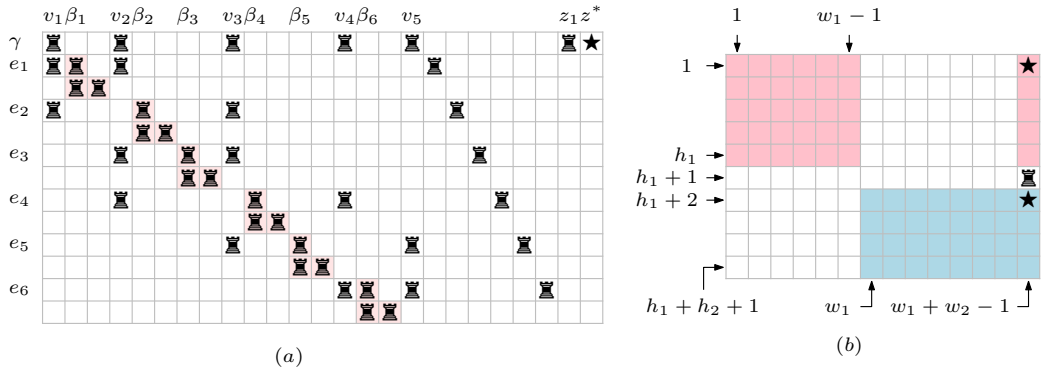


Figure 9 (a) The instance of Solo-Chess*(♜, {2}) by applying our transformation to the instance of Solo-Chess*(♜, {0, 2}) of Figure 7 (c). Red squares contains the 2-rooks replacing the original 0-rooks. (b) A sketch of the instance $\mathcal{I}_{1,2}$ of Solo-Chess(♜, {2}) obtained from the instances $\mathcal{I}_1, \mathcal{I}_2$ of Solo-Chess*(♜, {2}).

move m_i of σ is the capture $r \to q'$. We show that, if $i \neq 1$, swapping m_i with m_{i-1} still results in a winning sequence. Indeed, m_{i-1} cannot involve q and it cannot clear q' (as m_i would then be illegal).

If (ii) holds, then m_{i-1} cannot involve q' (if m_{i-1} targeted q' then the resulting configuration would be disconnected) and $\langle m_1, \ldots, m_i \rangle$ and $\langle m_1, \ldots, m_{i-2}, m_i, m_{i-1} \rangle$ result in the same configuration.

If (i) holds, then either m_{i-1} does not involve q', or m_{i-1} targets q'. In any case, the configurations obtained by performing $\langle m_1, \ldots, m_i \rangle$ and $\langle m_1, \ldots, m_{i-2}, m_i, m_{i-1} \rangle$ are identical except possibly for the budget of the rook in q', which is 0 in the former configuration and *at least* 0 in the latter. ◀

Notice that all the (non-goal) 0-rooks used in our reduction are on columns that do not contain any other rook. Then, to simulate a (non-goal) 0-rook on square (i, j) we first sets its budget to 2, then we insert new row $i + 1$ immediately below i and a new column $j + 1$ immediately to the right of j, and finally we add two 2-rooks in squares $(i + 1, j)$ and $(i + 1, j + 1)$. See Figure 9 (a) for an example instance of Solo-Chess*(♜, {2}) resulting from the above process. Clearly, if the original instance is solvable so is the one obtained after these 0-rooks have been replaced using the above strategy (since it is always possible to "recover" the original configuration, except for some additional empty rows and column, by performing two captures for each 0-rook that has been replaced), and a repeated application of Lemma 6 shows that the converse also holds.

We now reduce Solo-Chess*(♜, {2}) to Solo-Chess(♜, {2}).

Let $\mathcal{I}_1, \mathcal{I}_2$ be two instances of Solo-Chess*(♜, {2}) whose chessboards have sizes $h_1 \times w_1$ and $h_2 \times w_2$, respectively. We construct a new instance $\mathcal{I}_{1,2}$ of Solo-Chess(♜, {2}), whose chessboard has size $(h_1 + h_2 + 1) \times (w_1 + w_2 - 1)$, as follows (see Figure 9 (b)):

- the sub-chessboard of size $h_1 \times w_1$ consisting of the intersection of rows $1, 2, \ldots, h_1$ and columns $1, 2, \ldots, w_1 - 1, w_1 + w_2 - 1$ of $\mathcal{I}_{1,2}$ is a copy of chessboard of $\mathcal{I}_1$ in which the goal-rook is replaced with a 2-rook;
- the sub-chessboard of size $h_2 \times w_2$ consisting of the intersection of rows $h_1 + 2, h_1 + 3, \ldots, h_1 + h_2 + 1$ and columns $w_1, w_2, \ldots, w_1 + w_2 - 1$ of $\mathcal{I}_{1,2}$ is a copy of the chessboard of $\mathcal{I}_2$ in which the goal-rook is replaced with a 2-rook;
- square $(h_1 + 1, w_1 + w_2 - 1)$ contains a 2-rook.

▶ **Lemma 7.** *If both $\mathcal{I}_1$ and $\mathcal{I}_2$ are solvable then $\mathcal{I}_{1,2}$ is solvable. If $\mathcal{I}_{1,2}$ is solvable then at least one of $\mathcal{I}_1$ and $\mathcal{I}_2$ is solvable.*

Proof. Let R_1 be the set of rows $1, 2, \ldots, h_1$, and R_2 be the set of rows $h_1+2, \ldots, h_1+h_2+1$. Let $q_1 = (1, w_1+w_2-1)$, $q_2 = (h_1+2, w_1+w_2-1)$, and $q^* = (h_1+1, w_1+w_2-1)$. Moreover, let r_1, r_2, and r^* be the rooks initially on q_1, q_2, and q^* in $\mathcal{I}_{1,2}$, respectively.

If both $\mathcal{I}_1$ and $\mathcal{I}_2$ are solvable then a winning sequence of moves for $\mathcal{I}_{1,2}$ is obtained by: (i) performing all moves in any winning sequence for $\mathcal{I}_1$, where any capture $r \to (1, w_1)$ targeting $(1, w_1)$ in $\mathcal{I}_1$ is replaced with the capture $r \to q_1$ in $\mathcal{I}_{1,2}$; (ii) performing all moves in any winning sequence for $\mathcal{I}_2$, where any capture $(i, j) \to (i', j')$ in $\mathcal{I}_2$ is replaced with the capture $(h_1+1+i, w_1-1+j) \to (h_1+1+i', w_1-1+j')$ in $\mathcal{I}_{1,2}$; (iii) performing the captures by $r^* \to q_1 \to q_2$.

To show that if $\mathcal{I}_{1,2}$ is solvable then at least one of $\mathcal{I}_1$ and $\mathcal{I}_2$ is solvable, let $\sigma = \langle m_1, m_2, \ldots \rangle$ be a winning sequence of moves $\mathcal{I}_{1,2}$. Moreover, let ℓ be the smallest index such that m_ℓ is a vertical capture on column $w_1 + w_2 - 1$ (such an index must exist) and notice that there is some $h \in \{1, 2\}$ such that m_ℓ is either the capture $q_h \to q^*$ or the capture $r^* \to q_h$. We consider these two cases separately.

If m_ℓ is the capture $q_h \to q^*$, then r_h performs no capture in moves $m_1, \ldots, m_{\ell-1}$. Moreover, after move m_ℓ, all squares belonging to the rows in R_h must be empty (since otherwise the configuration would be disconnected). Since no move among $m_1, \ldots, m_{\ell-1}$ can simultaneously involve both a square of a row in R_1 and a square of a row in R_2, the sub-sequence of moves obtained from $\langle m_1, \ldots, m_{\ell-1} \rangle$ by selecting all moves that involve a square in R_h is a winning sequence of moves for $\mathcal{I}_h$.

If m_ℓ is the capture $r^* \to q_h$, then let $\ell' > \ell$ the only other index such that $m_{\ell'}$ is a vertical capture on column w_1+w_2-1. Such a capture is either $q_h \to q_{3-h}$ or $q_{3-h} \to q_h$. In the former case, the capture $q_h \to q_{3-h}$ has the effect of clearing square q_h and replacing the b-rook in q_{3-h} (where $b \in \{0, 1, 2\}$) with a 0-rook. Since no move m_t with $t \notin \{\ell, \ell'\}$ can simultaneously involve both a square of a row in R_1 and a square of a row in R_2, the sub-sequence of moves of σ that involve a squares of a row in R_{3-h} is a winning sequence of moves for $\mathcal{I}_{3-h}$. In the latter case, after capture $q_{3-h} \to q_h$, all squares belonging to rows in R_{3-h} must be empty, and the sub-sequence of moves obtained from $\langle m_1, \ldots, m_{\ell-1}, m_{\ell+1}, \ldots, m_{\ell'-1} \rangle$ by selecting all moves that involve a square in R_{3-h} is a winning sequence of moves for $\mathcal{I}_h$. ◀

Then, if $\mathcal{I}$ is an instance of Solo-Chess*(♜, {2}), we can perform the above transformation with with $\mathcal{I}_1 = \mathcal{I}_2 = \mathcal{I}$ to obtain an instance $\mathcal{I}_{1,2}$ of Solo-Chess(♜, {2}) that is solvable iff $\mathcal{I}$ is solvable, as ensured by Lemma 7 (see Figure 10). We have thus proved the following:

▶ **Theorem 8.** Solo-Chess(♜, {2}) *is NP-hard.*

References

1 N. R. Aravind, Neeldhara Misra, and Harshil Mittal. Chess is hard even for a single player. In Pierre Fraigniaud and Yushi Uno, editors, *11th International Conference on Fun with Algorithms, FUN 2022, May 30 to June 3, 2022, Island of Favignana, Sicily, Italy*, volume 226 of *LIPIcs*, pages 5:1–5:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPICS.FUN.2022.5.

2 Giuseppe Di Battista, Peter Eades, Roberto Tamassia, and Ioannis G. Tollis. Algorithms for drawing graphs: an annotated bibliography. *Comput. Geom.*, 4:235–282, 1994. doi:10.1016/0925-7721(94)00014-X.

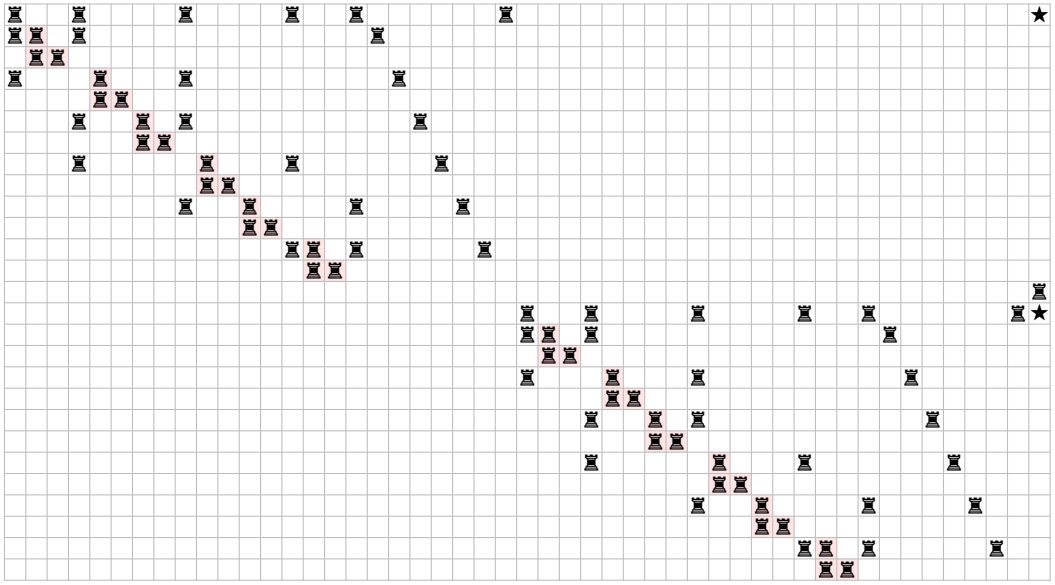


Figure 10 The final instance Solo-Chess(♜, {2}) corresponding to the vertex cover instance of Figure 7 (a).

3 Davide Bilò, Luciano Gualà, Stefano Leucci, Guido Proietti, and Mirko Rossi. On the pspace-completeness of peg duotaire and other peg-jumping games. In Hiro Ito, Stefano Leonardi, Linda Pagli, and Giuseppe Prencipe, editors, *9th International Conference on Fun with Algorithms, FUN 2018, June 13-15, 2018, La Maddalena, Italy*, volume 100 of *LIPIcs*, pages 8:1–8:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.FUN.2018.8.

4 Josh Brunner, Lily Chung, Michael J. Coulombe, Erik D. Demaine, Timothy Gomez, and Jayson Lynch. Complexity of solo chess with unlimited moves. *CoRR*, abs/2302.01405, 2023. doi:10.48550/ARXIV.2302.01405.

5 Aviezri S. Fraenkel and David Lichtenstein. Computing a perfect strategy for n x n chess requires time exponential in n. *J. Comb. Theory, Ser. A*, 31(2):199–214, 1981. doi:10.1016/0097-3165(81)90016-9.

6 M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

7 M. R. Garey, David S. Johnson, and Robert Endre Tarjan. The planar hamiltonian circuit problem is np-complete. *SIAM J. Comput.*, 5(4):704–714, 1976. doi:10.1137/0205049.

8 Luciano Gualà, Stefano Leucci, Emanuele Natale, and Roberto Tauraso. Large peg-army maneuvers. In Erik D. Demaine and Fabrizio Grandoni, editors, *8th International Conference on Fun with Algorithms, FUN 2016, June 8-10, 2016, La Maddalena, Italy*, volume 49 of *LIPIcs*, pages 18:1–18:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPICS.FUN.2016.18.

9 Chess.com LLC. Solo Chess - Capture All the Pieces on the Board. https://www.chess.com/solo-chess. Accessed: 2024-02-25.

10 James A. Storer. On the complexity of chess. *J. Comput. Syst. Sci.*, 27(1):77–100, 1983. doi:10.1016/0022-0000(83)90030-2.

11 Ryuhei Uehara and Shigeki Iwata. Generalized hi-q is np-complete. *IEICE TRANSACTIONS (1976-1990)*, 73(2):270–273, 1990.

12 Leslie G. Valiant. Universality considerations in VLSI circuits. *IEEE Trans. Computers*, 30(2):135–140, 1981. doi:10.1109/TC.1981.6312176.