Chess Is Hard Even for a Single Player

N.R. Aravind
Department of Computer Science and Engineering, Indian Institute of Technology, Hydrebad, India

Neeldhara Misra
Department of Computer Science and Engineering, Indian Institute of Technology, Gandhinagar, India

Harshil Mittal
Department of Computer Science and Engineering, Indian Institute of Technology, Gandhinagar, India

Abstract

We introduce a generalization of "Solo Chess", a single-player variant of the game that can be played on chess.com. The standard version of the game is played on a regular 8×8 chessboard by a single player, with only white pieces, using the following rules: every move must capture a piece, no piece may capture more than 2 times, and if there is a King on the board, it must be the final piece. The goal is to clear the board, i.e, make a sequence of captures after which only one piece is left.

We generalize this game to unbounded boards with n pieces, each of which have a given number of captures that they are permitted to make. We show that GENERALIZED SOLO CHESS is NP-complete, even when it is played by only rooks that have at most two captures remaining. It also turns out to be NP-complete even when every piece is a queen with exactly two captures remaining in the initial configuration. In contrast, we show that solvable instances of GENERALIZED SOLO CHESS can be completely characterized when the game is: a) played by rooks on a one-dimensional board, and b) played by pawns with two captures left on a 2D board.

Inspired by GENERALIZED SOLO CHESS, we also introduce the GRAPH CAPTURE GAME, which involves clearing a graph of tokens via captures along edges. This game subsumes GENERALIZED SOLO CHESS played by knights. We show that the GRAPH CAPTURE GAME is NP-complete for undirected graphs and DAGs.

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics; Theory of computation → Design and analysis of algorithms

Keywords and phrases chess, strategy, board games, NP-complete

Digital Object Identifier 10.4230/LIPIcs.FUN.2022.5

Related Version *Full Version*: `https://arxiv.org/abs/2203.14864` [1]

Funding The second author acknowledges support from IIT Gandhinagar and the SERB-MATRICS grant MTR/2017/001033. The second and third authors also acknowledge support from the SERB-ECR grant ECR/2018/002967.

Acknowledgements The authors would like to thank the anonymous reviewers of FUN 2022 for their useful feedback.

1 Introduction

Chess, the perfect-information two-player board game, needs to introduction. With origins dating back to as early as the 7th century, organized chess arose in the 19th century to become one of the world's most popular games in current times. At the time of this writing, the recent pandemic years witnessed a phenomenal growth of the already popular game, among spectators and amateur players alike. One of the most active computer chess sites, chess.com, is reported to have more than 75 million members, and about four million people sign in everyday.

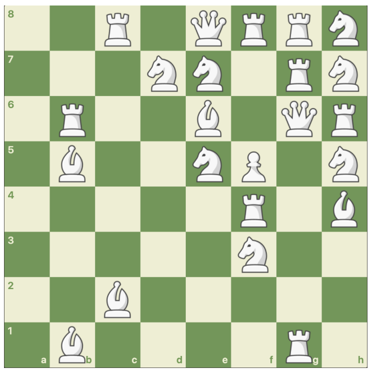


Figure 1 An example of a Solo Chess configuration.

As the reader likely knows already, chess is played on a square chessboard with 64 squares arranged in an eight-by-eight grid. At the start, each player (one controlling the white pieces, the other controlling the black pieces) controls sixteen pieces: one king, one queen, two rooks, two bishops, two knights, and eight pawns. The object of the game is to checkmate the opponent's king, whereby the king is under immediate attack (in "check") and there is no way for it to escape. There are also several ways a game can end in a draw. The movements of the individual pieces are subject to different constraints. While several chess engines exist for this classical version of the game, it is also known that the generalized version of chess, played on a $n \times n$ board by two players with $2n$ pieces is complete for the class EXPTIME [5]. Cooperative versions of chess are also known to be hard [3].

The game of Solo Chess is an arguably natural single-player variant of the game. We consider here a version that can be found among the chess puzzles on chess.com. The game is played on a regular 8×8 chessboard by a single player, with only white pieces, using the following rules: every move must capture a piece, no piece may capture more than 2 times, and if there is a King on the board, it must be the final piece. Given a board with, say, n pieces in some configuration, the goal is to play a sequence of captures that "clear" the board. To the best of our knowledge, chess.com presents its players only with solvable configurations, even if this may not always be obvious[1]. The solutions, however, need not be unique.

While the focus of our contribution here is on computational aspects of determining if a solo chess instance is solvable, we refer the reader to [2] for a comprehensive and entertaining introduction to combinatorial game theory at large.

Our Contributions

We introduce a natural generalization of Solo Chess that we call Generalized Solo Chess(P, d), where $P \subseteq \{$♕, ♖, ♗, ♙, ♘$\}$ is a collection of piece types and $d \in \mathbb{N}$. This version of the game is played on the infinite integer lattice where we are given, initially, the positions of n pieces, each of which is one of the types given in P. We are also given, for each piece, the number of captures it can make – and further, this bound is at most d. The goal is to figure out if there is a sequence of $(n-1)$ valid captures such that: a) no piece captures more than the number of times it is allowed to capture; and b) the sequence of captures, when played out, "clears the board", i.e, only one piece remains at the end.

[1] c.f. "Crazy Mode".

We focus on settings where $|P| = 1$, i.e, when all pieces are of the same type. When the game is played only with rooks, we show that the problem is NP-complete even when $d = 2$, but is tractable when the game is restricted to a one-dimensional board for arbitrary d.

▶ **Theorem 1** (A characterization for rooks on 1D boards). *Generalized Solo Chess (♖, d) with n rooks can be decided in $O(n)$ time for any $d \in \mathbb{N}$.*

▶ **Theorem 2** (Intractability for rooks). *Generalized Solo Chess (♖, 2) is* NP*-complete.*

When all pieces are queens, note that Generalized Solo Chess played on a one-dimensional board is equivalent to the game played by rooks. On the other hand, on a two-dimensional board, the game turns out to be hard even when all pieces can capture twice in the initial configuration, which is in the spirit of the regular game and is a strengthening of the hardness that we have for rooks.

▶ **Theorem 3** (Intractability for queens). *Generalized Solo Chess (♕, 2) is* NP*-complete even when all queens are allowed to capture at most twice.*

When all pieces are bishops, no piece has a valid move if the game is restricted to a one-dimensional board. On the other hand, it is easy to check that Generalized Solo Chess played on a two-dimensional board with bishops only can be reduced to Generalized Solo Chess played on a two-dimensional board with rooks only, by simply "rotating" the board 45 degrees. Therefore, we do not discuss the case of bishops explicitly.

We now turn to the case when the game is played only with pawns: as with bishops, the game is not interesting on a one-dimensional board. However, when played on a two-dimensional board with pawns that have two captures left, it turns out that we can efficiently characterize the solvable instances.

▶ **Theorem 4** (A characterization for pawns). *Generalized Solo Chess (♙, 2) with n white pawns, each of which can capture at most twice, can be decided in $O(n)$ time.*

When the game is played by knights only, again the game is trivial on a one-dimensional board. On a two-dimensional board, consider the following graph that is naturally associated with any configuration of knights: we introduce a vertex for every occupied position, and a pair of vertices are adjacent if and only if the corresponding positions are mutually attacking. Note that for all other pieces considered so far, an attacking pair of positions need not imply that a capture is feasible, since there may be blocking pieces in some intermediate locations. Knights are unique in that the obstacles are immaterial. This motivates the Graph Capture game: here we are given a graph with tokens on vertices, and the goal is to clear the tokens by a sequence of captures. The tokens can capture along edges and the number of captures that the tokens can make is given as a part of the input.

Note that Generalized Solo Chess(♘, d) is a special case of of Graph Capture(d). We show that solvable instances of the latter on undirected graphs are characterized by the presence of a rooted spanning tree with the property that every internal node has at least one leaf neighbor. However, we also show that finding such spanning trees is intractable. We also show that Graph Capture(d) is NP-complete on DAGs.

▶ **Theorem 5** (Intractability of the graph capture game). *Graph Capture(2) is NP-complete on undirected graphs and DAGs even when every token can capture at most twice.*

We remark that Theorem 5 has no immediate implications for Generalized Solo Chess (♘, d).

The rest of the paper is organized as follows. We establish the notation that we will use in Section 2. The proof of Theorems 1 and 2 is given in Section 3.1.1 and Section 3.1.2, respectively. The proof of Theorem 3 is discussed in Section 3.2 and the proof of Theorem 4 is given in Section 3.3. Finally, the proof of Theorem 5 is shown separately for undirected graphs and DAGs in Sections 4.1 and 4.2.

2 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. We consider the following generalization[2] of Solo Chess. We fix a subset P of {♛, ♖, ♝, ♙, ♘} and a positive integer d. The generalized game is played on an infinite two-dimensional board with n pieces. For each piece, we are given an initial location and the maximum number of captures the piece is permitted to make. Such an instance is solvable if there exists a sequence σ of $(n-1)$ valid captures with each piece making at most as many captures as it is allowed to make. We note that a capture is valid if it respects the usual rules of movements in chess. The formal definition of the problem is the following.

GENERALIZED SOLO CHESS(P, d):

Input: A configuration C, which is specified by a list of n triplets (p, z, m), where $p \in P$, $z \in \mathbb{N} \times \mathbb{N}$, and $0 \leqslant m \leqslant d$. We use C_i to refer to the i^{th} triplet in C.
Output: Decide if there exists a sequence of $(n-1)$ captures starting from the board position described by C, such that the piece corresponding to $C[i][0]$ moves at most $C[i][2]$ times for all $1 \leqslant i \leqslant n$.

We note that GENERALIZED SOLO CHESS(P, d) is interesting when $d \geqslant 2$. Indeed, when $d = 1$, it can be efficiently determined if an instance of GENERALIZED SOLO CHESS$(P, 1)$ is solvable:

▶ **Observation 6.** *When $d = 1$, a configuration C is winning if and only if there's a square z containing a piece, such that z is reachable in one move from every other piece.*

Proof. The sufficiency of this condition is clear; to see the necessity, for each square y on which a capture was made, let $p(y)$ be the last piece to capture on y. Then $p(y)$ must be the last piece standing (as it can neither move again nor be captured), and further, y is the occupied square at the end of the game. Since there is exactly one occupied square at the end, this shows that all captures were made to the same square. ◀

Most of our results rely only on elementary graph-theoretic terminology and the notions of polynomial time reductions and NP-completeness. We refer the reader to the texts [6, 7] for the relevant background. The well-known [6, 4] NP-complete problems that we use in our reductions are the following:

1. RED-BLUE DOMINATING SET. Given a bipartite graph $G = (R \uplus B, E)$ and a positive integer k, determine if there is a subset $S \subseteq R$, $|S| \leqslant k$ such that $N[v] \cap S \neq \emptyset$ for all $v \in B$.

[2] Since our focus us on the case when the game is played by pieces of one type only, we do not involve the ♔ in our set of pieces. Note that because of the convention that kings are never captured, any such involving only kings is trivial.

2. Colorful Red-Blue Dominating Set. Given a bipartite graph $G = (R \uplus B, E)$ where the red vertices are partitioned into k disjoint parts, determine if there is a choice of exactly one vertex from each part such that every blue vertex has at least one neighbor among the chosen vertices.
3. 3-SAT. Given a CNF formula with at most three literals per clause, determine if there is a truth assignment to the variables that satisfies the formula.

3 Solo Chess with a single piece

3.1 Rooks

3.1.1 1-Dimensional boards

In this section we consider Generalized Solo Chess restricted to one-dimensional board played by rooks. It will be convenient to reason about such instances by using strings to represent game configurations; to this end we introduce some terminology.

▶ **Definition 7** (Configuration). *A* configuration *is string* s *over* $\{0, 1, 2, \ldots, d, \square\}$. *It denotes a board of size* $1 \times N$*, where* N *is the length of the string. The cell* $(1, j)$ *is empty if* $s[j] = \square$*, and is otherwise occupied by a rook with* b *moves left where* $b := s[j]$.

We refer the reader to Figure 2 for an example and how a given board position translates to a configuration as defined above. Informally, a configuration is *solvable* if there is a valid sequence of moves that clears the board.

▶ **Definition 8** (ℓ-solvable configuration). *Let* s *be a configuration of length* N *and let* $1 \leqslant \ell \leqslant N$. *We say that* s *is* ℓ*-solvable if there exists a sequence of moves that clears the corresponding board with the final rook at the cell* $(1, \ell)$.

Our main goal in this section is to establish the following:

▶ **Theorem 1** (A characterization for rooks on 1D boards). *Generalized Solo Chess* $(♖, d)$ *with* n *rooks can be decided in* $O(n)$ *time for any* $d \in \mathbb{N}$.

Note that for any sequence (say σ) of moves that clears the board with final rook at the cell $(1, \ell)$, no move of σ empties the cell $(1, \ell)$ and thus, there's no move of σ wherein the cells containing the captured piece and the capturing piece are at different sides, i.e., one at left and the other at right, of the cell $(1, \ell)$. So, note that s is ℓ-solvable if and only if there is a position such that the sub-configurations to the left and right of location ℓ are independently solvable. We now develop a criteria for solving 1D configurations where the target piece is one of the extreme locations on the board. Note that when $d = 2$, the first criteria below amounts to saying that s is N-solvable iff $s[N] \neq \square$ and $s[1, \ldots, N-1]$ has at least as many 2's as 0's; and the second criteria states that s is 1-solvable iff $s[1] \neq \square$ and $s[2, \ldots, N]$ has at least as many 2's as 0's. A direct proof of this simpler statement is given in the Appendix [1].

▶ **Lemma 9.** *For every configuration* s *of length* N*,*
1. s *is* N*-solvable iff* $s[N] \neq \square$ *and* $\sum_{\substack{1 \leqslant i \leqslant N-1: \\ s[i] \notin \{0, \square\}}} (s[i] - 1) \geqslant$ *number of* 0*'s in* $s[1, \ldots, N-1]$
2. s *is* 1*-solvable iff* $s[1] \neq \square$ *and* $\sum_{\substack{2 \leqslant i \leqslant N: \\ s[i] \notin \{0, \square\}}} (s[i] - 1) \geqslant$ *number of* 0*'s in* $s[2, \ldots, N]$

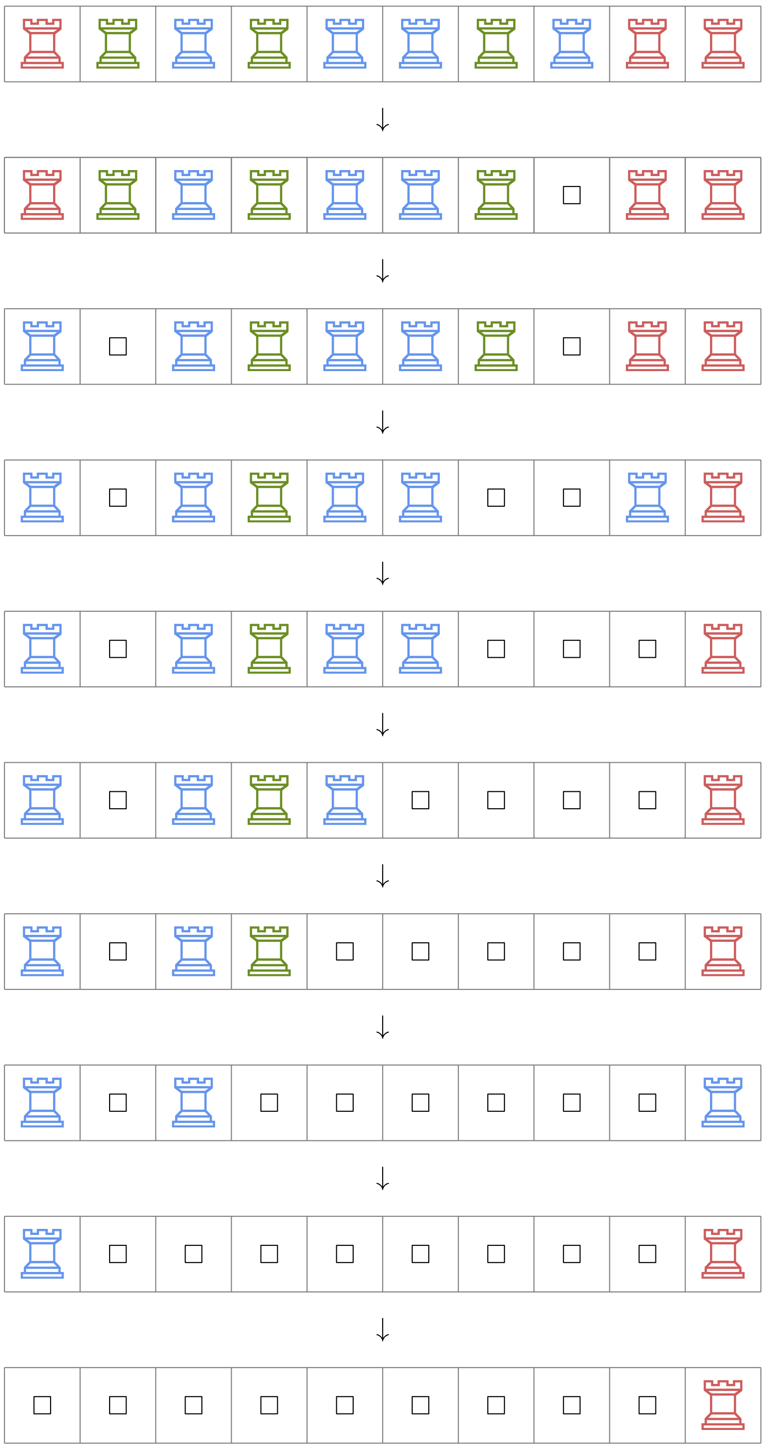


Figure 2 An example of a valid sequence of captures that clears the board. The initial configuration corresponds to the string `0212112100`. In other words, the red, blue, and green rooks denote rooks with zero, one, and two moves left, respectively. Notice that this is not a unique solution – there are several other valid sequences that also successfully clear this board.

Proof. We argue the first claim since the proof of the second is symmetric. For the forward implication, we show (using induction on m) that the following statement is true for all integers $m \geqslant 0$: For every configuration s such that $\sum_{\substack{1\leqslant i\leqslant N-1:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) = m$, if s is N-solvable, then $s[N] \neq \square$ and $m \geqslant$ number of 0's in $s[1,\ldots,N-1]$. For the base case, consider $m = 0$. The only configurations s with $\sum_{\substack{1\leqslant i\leqslant N-1:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) = 0$ are the ones for which $s[1,\ldots,N-1]$ is a string over $\{0,1,\square\}$. Among these, the only N-solvable configurations s are the ones for which $s[1,\ldots,N-1]$ is a string over $\{1,\square\}$ and $s[N] \neq \square$. Thus, the statement is true for $m = 0$.

As induction hypothesis, assume that the statement is true for all integers $0 \leqslant m \leqslant p$, for some integer $p \geqslant 0$. Let's argue that the statement is true for $m = p+1$. Let s be a configuration such that $\sum_{\substack{1\leqslant i\leqslant N-1:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) = p+1$ and s is N-solvable. As s is N-solvable, there exists a sequence of moves (say σ) that clears the corresponding $1\times N$ board with the final rook at the cell $(1,N)$. Clearly, $s[N] \neq \square$. Let $t \geqslant 1$ be the least integer such that the capturing piece in t^{th} move of σ is not a 1-rook. Let $\tilde{s}$ denote the configuration corresponding to the board obtained after t^{th} move of σ. Note that $\tilde{s}$ is N-solvable and $\sum_{\substack{1\leqslant i\leqslant N-1:\\ \tilde{s}[i]\notin\{0,\square\}}} (\tilde{s}[i]-1) \leqslant p$. Using induction hypothesis, $p \geqslant$ number of $0's$ in $\tilde{s}[1,\ldots,N-1]$. Also, number of 0's in $\tilde{s}[1,\ldots,N-1] \geqslant$ number of 0's in $s[1,\ldots,N-1]-1$; this is because the number of 0-rooks at the cells $(1,1),\ldots,(1,N-1)$ doesn't decrease in the first $t-1$ moves of σ, and decreases by at most 1 in the t^{th} move of σ. Therefore, we have $p+1 \geqslant$ number of 0's in $s[1,\ldots,N-1]$, as desired.

For the converse, we show (using induction on m) that the following statement is true for all integers $m \geqslant 0$: For every configuration s such that $s[N] \neq \square$, if $s[1,\ldots,N-1]$ has exactly m $0's$ and $\sum_{\substack{1\leqslant i\leqslant N-1:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) \geqslant m$, then s is N-solvable.

For the base case, consider $m = 0$. Let s be a configuration such that $s[N] \neq \square$ and $s[1,\ldots,N-1]$ has no 0's. For each $1 \leqslant i < N$, the cell $(1,i)$ in the corresponding board is either empty or has a $1/2/\ldots/d$-rook. The board can be cleared with the final piece at $(1,N)$ by making the $1/2/\ldots/d$-rooks (if any) to capture rook at the cell $(1,N)$. Thus, s is N-solvable. So, the statement is true for $m = 0$.

As induction hypothesis, assume that the statement is true for all integers $0 \leqslant m \leqslant p$, for some integer $p \geqslant 0$. Let's argue that the statement is true for $m = p+1$. Let s be a configuration such that $s[N] \neq \square$, $s[1,\ldots,N-1]$ has exactly $(p+1)$ $0's$ and $\sum_{\substack{1\leqslant i\leqslant N-1:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) \geqslant p+1$. While there is a 1-rook in the cells $(1,1),\ldots,(1,N-1)$ that can capture a 0-rook in the cells $(1,1),\ldots,(1,N-1)$, make such a move. Once no such move can be made, there exist integers $1 \leqslant u < v < N$ such that $s[u,\ldots,v] = 0\square^{\lambda}x$ or $s[u,\ldots,v] = x\square^{\lambda}0$, for some $\lambda \geqslant 0$ and some $2 \leqslant x \leqslant d$. In the former (resp. latter) case, the x-rook at the cell $(1,v)$ (resp. $(1,u)$) can be made to capture the 0-rook at the cell $(1,u)$ (resp. $(1,v)$), and the configuration corresponding to the resulting board is N-solvable by induction hypothesis. ◀

We conclude that a configuration s of length N is solvable iff there exists $1 \leqslant \ell \leqslant N$ such that

- $s[\ell] \neq \square$,

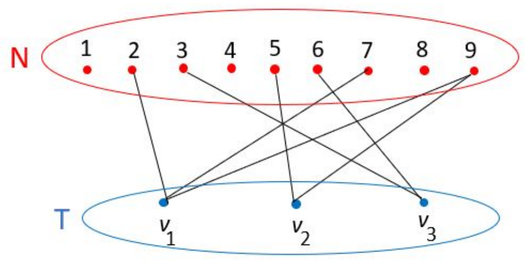


Figure 3 An instance of Red-Blue Dominating Set.

- $\sum_{\substack{1\leqslant i\leqslant \ell-1:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) \geqslant$ number of 0's in $s[1,\ldots,\ell-1]$, and
- $\sum_{\substack{\ell+1\leqslant i\leqslant N:\\ s[i]\notin\{0,\square\}}} \big(s[i]-1\big) \geqslant$ number of 0's in $s[\ell+1,\ldots,N]$.

3.1.2 2-Dimensional boards

▶ **Theorem 2** (Intractability for rooks). *Generalized Solo Chess (♖, 2) is* NP*-complete.*

Proof. We reduce from the Red-Blue Dominating Set problem. Let $\mathcal{I} := \langle G = (N \cup T, E); k\rangle$ be an instance of Red-Blue Dominating Set. Recall that G is a bipartite graph with bipartition N and T; and $\mathcal{I}$ is a Yes-instance if and only if there exists a subset $S \subseteq N$ of size at most k such that every vertex v in T has a neighbor in S. We let the vertices in N be denoted by $[n]$ and let $T := \{v_1, \ldots, v_m\}$. We refer to the vertices of N and T as non-terminals and terminals, respectively.

We first describe the construction of the reduced instance of Generalized Solo Chess (♖, 2) based on $\mathcal{I}$. The game takes place on a $(2m + 1) \times (n + m + k + 1)$ board. The initial position of the rooks is as follows:

- *Non-terminal* rooks. For all $i \in [n]$, we place a 1-rook in the cell $(2m + 1, i)$.
- *Terminal* rooks. For all $j \in [m]$, we place a 1-rook in the cell $(2j - 1, \ell)$ for each ℓ such that $\ell \in N(v_j)$.
- *Collector* rooks. For all $j \in [m]$, we place a 2-rook in the cell $(2j - 1, n + j)$.
- *Cleaner* rooks. For all $\ell \in [k]$, we place a 2-rook in the cell $(2m + 1, n + m + \ell)$.
- Target location. Finally, we place on 1-rook at the location $(2m + 1, n + m + k + 1)$.

The non-terminal and terminal rooks correspond to the non-terminal and terminal vertices in the graph, and their relative positioning as described above captures the graph structure. The rooks on every row are expected to "clear to one of the columns corresponding to a vertex they are dominated by", and the other auxiliary rooks added to the board above help with clearing the board after this phase, as explained further below.

Forward Direction

Assume that $\mathcal{I}$ is a YES-instance. That is, there exists $S \subseteq [n]$ of size at most k such that every vertex in T has a neighbour in S. For each $j \in [m]$, let $f(j)$ denote an arbitrary but fixed $i \in S$ such that $i \in N(v_j)$. Consider the following sequence of moves (c.f. Figure 5):

- For each $j \in [m]$ and each $\ell \in N(v_j) \setminus \{f(j)\}$, the terminal 1-rook at the cell $(2j - 1, \ell)$ captures rook at the cell $(2j - 1, f(j))$.

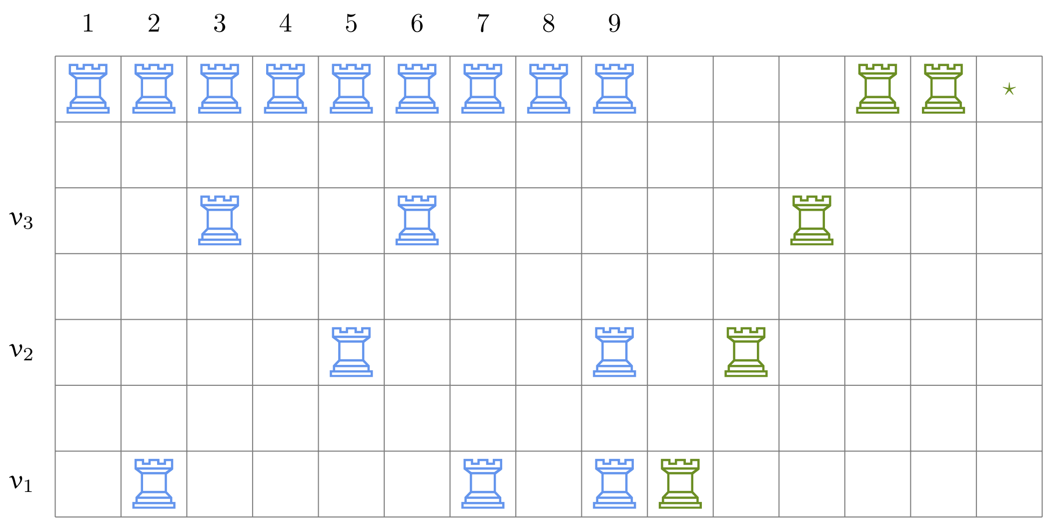


Figure 4 The reduced instance of Generalized Solo Chess played by rooks corresponding to the instance shown in Figure 3.

- For each $j \in [m]$, the collector 2-rook at the cell $(2j-1, n+j)$ captures the rook at the cell $(2j-1, f(j))$, and the 1-rook at the cell $(2j-1, f(j))$ so obtained then captures rook at the cell $(2m+1, f(j))$.
- For each $i \in [n]$, if there's a non-terminal 1-rook at the cell $(2m+1, i)$, then it captures one of the 0-rooks at the cells $(2m+1, f(1)), \ldots, (2m+1, f(m))$.

Now, the board is empty except for the top row which has one 1-rook at the target location, k cleaner 2-rooks and at most k 0-rooks, i.e., the 0-rooks at the cells $(2m+1, f(1)), \ldots, (2m+1, f(m))$. Using Lemma 9, this corresponds to a $(n+m+k+1)$-solvable configuration.

Reverse Direction

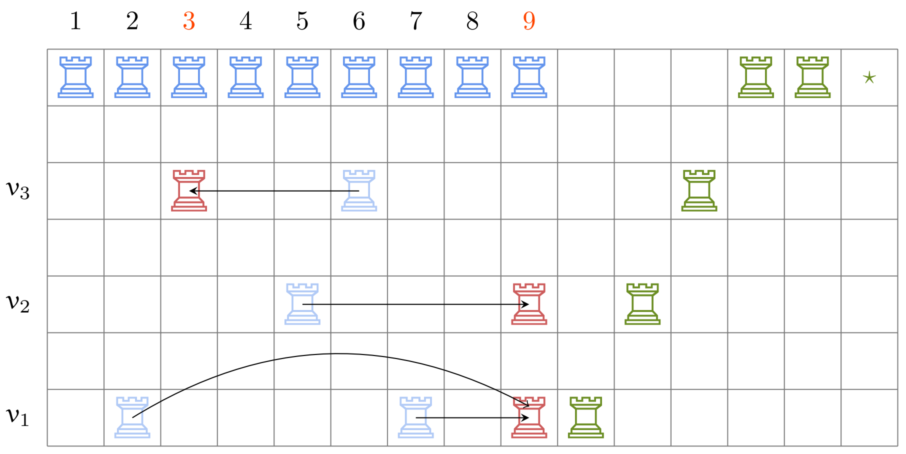
Suppose the reduced instance is solvable. We first make some claims about any valid sequence of s moves, denoted by σ, that clears the board. Let $\rho_\sigma((x,y),\ell)$ denote the type of the piece at the location (x,y) after ℓ moves of σ have been played. If (x,y) is an empty location after ℓ moves of σ have been played, then we let $\rho_\sigma((x,y),\ell) = \square$.

Let $\zeta(t)$ denote the set of locations $(2m+1,\cdot)$ occupied by red rooks on the top row of the board after t moves of σ have been made, in other words: $\zeta(t) = \{i \mid \rho_\sigma((2m+1,i),t) = ♖\}$.

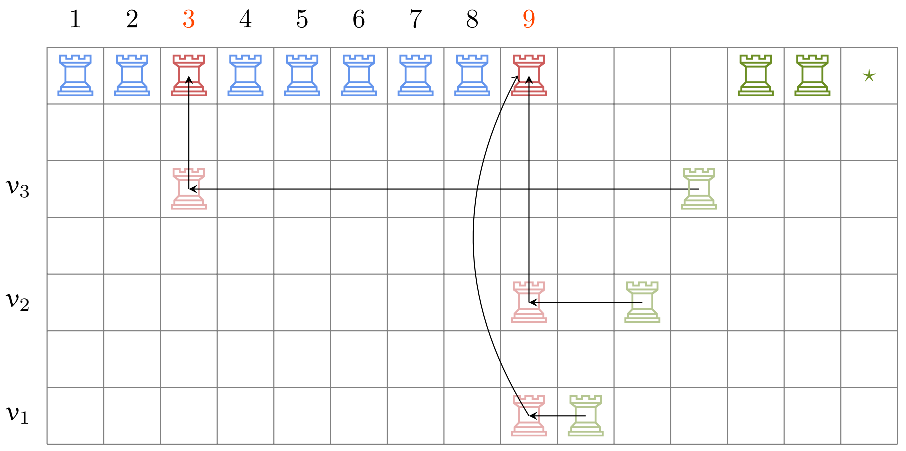
▷ Claim 10. $|\bigcup_{1\leqslant t\leqslant s} \zeta(t)| \leqslant k$.

Proof. Let $i \in \bigcup_{1\leqslant t\leqslant s} \zeta(t)$. That is, there exists $1 \leqslant t \leqslant s$ such that the cell $(2m+1, i)$ has a 0-rook after t moves of σ. Let $p > t$ denote the first move of σ that empties the cell $(2m+1, i)$. Note that the cell $(2m+1, i)$ has a 1-rook before the p^{th} move of σ. So, there exists $t < q < p$ such that a 2-rook captures 0-rook at cell $(2m+1, i)$ in q^{th} move of σ. Also, such a 2-rook is one among the k cleaner rooks. Thus, $\left|\bigcup_{1\leqslant t\leqslant s} \zeta(t)\right| \leqslant k$. ◁

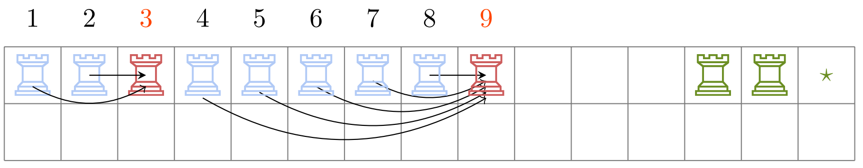
Let $j \in [m]$ and $i \in [n]$. We say that i is an j-*affected index* if there is some $t \in [s]$ such that the rook at position $(2j-1, i)$ was captured by the green rook originally at position $(2j-1, n+j)$ in the t^{th} move of σ.



(a) All blue rooks on rows corresponding to blue vertices clear row-wise to the dominating set vertices.



(b) The green rooks on the rows corresponding to blue vertices "pick up" the red rooks and capture along the column to get the rook on the top row.



(c) All blue rooks on the top row capture one of the red rooks leaving us in a solvable state with the two green rooks making the final captures.

Figure 5 All illustration of the forward direction.

▷ Claim 11. For a fixed $j \in [m]$, there is exactly one $i \in [n]$ such that i is a j-affected index.

Proof. Let $j \in [m]$. Let $t \in [s]$ denote the first move of σ wherein the collector 2-rook (say g) at the cell $(2j-1, n+j)$ either gets captured (Case 1) or captures (Case 2).

In Case 1, a terminal 1-rook in the row $2j-1$ captures g in the t^{th} move of σ. After the t^{th} move of σ, the cell $(2j-1, n+j)$ has a 0-rook. Let $p > t$ denote the first move of σ that empties the cell $(2j-1, n+j)$. Note that the cell $(2j-1, n+j)$ has a 1-rook before the p-th move of σ. So, there exists $t < q < p$ such that a 2-rook captures 0-rook at the cell $(2j-1, n+j)$ in q^{th} move of σ. However, no such 2-rook exists on the board. Thus, Case 1 does not arise.

In Case 2, there exists $i \in [n]$ such that g captures the rook at the cell $(2j-1, i)$ in the t^{th} move of σ. Note that i is the unique j-affected index. ◁

We call $i \in [n]$ an *affected index* if there is some $t \in [s]$ such that the position $(2m+1, i)$ was occupied by a red rook after t moves of σ.

▷ Claim 12. If $i \in [n]$ is a j-affected index for some $j \in [m]$, then i is also an affected index.

Proof. Assume that $i \in [n]$ is a j-affected index for some $j \in [m]$. That is, there exists $t \in [s]$ such that in the t^{th} move of σ, the collector 2-rook at the cell $(2j-1, n+j)$ captures the rook at the cell $(2j-1, i)$. After the t^{th} move of σ, the cell $(2j-1, i)$ has a 1-rook. Let $t' > t$ denote the first move of σ wherein the 1-rook at the cell $(2j-1, i)$ either gets captured (Case 1) or captures (Case 2).

In Case 1, a 1-rook captures the 1-rook at the cell $(2j-1, i)$ in the t'^{th} move of σ. After the t'^{th} move of σ, the cell $(2j-1, i)$ has a 0-rook. Let $p > t'$ denote the first move of σ that empties the cell $(2j-1, i)$. Note that the cell $(2j-1, i)$ has a 1-rook before the p^{th} move of σ. So, there exists $t' < q < p$ such that a 2-rook captures 0-rook at the cell $(2j-1, i)$ in the q^{th} move of σ. However, no such 2-rook exists on the board. Thus, Case 1 does not arise.

In Case 2, in the t'^{th} move of σ, the 1-rook at the cell $(2j-1, i)$ captures either rook at the cell $(2m+1, i)$ (Subcase 1), or rook at the cell $(2j'-1, i)$ for some $j' \in [m] \setminus \{j\}$ (Subcase 2).

In Subcase 1, the cell $(2m+1, i)$ has a 0-rook after t' moves of σ. So, i is an affected index.

In Subcase 2, the cell $(2j'-1, i)$ has a 0-rook after t' moves of σ. Let $p' > t'$ denote the first move of σ that empties the cell $(2j'-1, i)$. Note that the cell $(2j'-1, i)$ has a 1-rook before the p'^{th} move of σ. So, there exists $t' < q' < p'$ such that a 2-rook captures 0-rook at the cell $(2j'-1, i)$ in the q'^{th} move of σ. Note that this 2-rook is the collector rook at the cell $(2j'-1, n+j')$. After the q'^{th} move of σ, the cell $(2j'-1, i)$ has a 1-rook. Let $t'' > q$ denote the first move of σ wherein the 1-rook at the cell $(2j'-1, i)$ either gets captured or captures. As before, it can be argued that in the t''^{th} move, the 1-rook at the cell $(2j'-1, i)$ is not captured, and it either captures rook at the cell $(2m+1, i)$ (in which case we are done), or rook at the cell $(2j''-1, i)$ for some $j'' \in [m] \setminus \{j, j'\}$ (in which case the collector 2-rook at the cell $(2j''-1, n+j'')$ captures the 0-rook at the cell $(2j''-1, i)$ in some subsequent move). Repeatedly using the same argument proves the claim. ◁

Consider $S := \{\ell \mid \ell \in [n] \text{ and } \ell \text{ is an affected index}\}$. We claim that S is a dominating set in G. Indeed, consider any non-terminal vertex $v_j \in B$. If i is the unique j-affected index, then i is also an affected index, and therefore belongs to the dominating set. Note that $i \in N(v_j)$ by construction, therefore we are done. Also, by Claim 6, we have that $|S| \leqslant k$. This concludes the proof in the reverse direction. ◀

3.2 Queens

Recall the reduction described for the proof of Theorem 2. It is straightforward to check that if we introduce a large number – say $O(n^2)$ many – empty columns between every pair of consecutive columns of the original board, then we can also replace the rooks by queens and the reduction will remain valid. This is because the vast empty spaces essentially “nullify” the additional diagonal moves of the queens, thereby reducing their behavior to being equivalent to rooks. Also note that the operation of adding empty columns does not affect the forward direction: all pairs of mutually attacking locations remain mutually attacking even after this modification.

We now present the following strengthening of this hardness result. Recall that in the previous reduction, we had pieces that were allowed to capture twice and others that were allowed to capture once. With queens, however, we can adapt the reduction so that every piece is allowed to capture twice, bringing this closer to the spirit of traditional solo chess:

▶ **Theorem 3** (Intractability for queens). *Generalized Solo Chess (♕, 2) is* NP*-complete even when all queens are allowed to capture at most twice.*

We note that this result can be achieved by replacing every queen that is allowed to capture once with the following pair of queens that are both allowed to capture twice, with the queen on the bottom right replacing the "original" 1-queen:

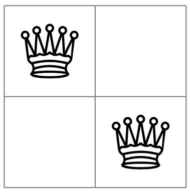


Figure 6 The reduced instance of Generalized Solo Chess played by rooks corresponding to the instance shown in Figure 3.

We call the queen on the top-left corner the supporting queen, and refer to the other queen as its partner. Once all the 1-queens of the reduced instance are replaced in this way, we ensure that all supporting queens have the property that they do not attack any queen other than their partner. To achieve this, we shift them north-west along their diagonals appropriately if required. Note that the fact that the supporting queens attack only their partners forces that they are never captured by another piece, and that they capture their partner queen, which replaces the partner with a 1-queen, as desired. We omit the details here.

3.3 Pawns

In contrast to the cases of Rooks, Queens and Bishops, we show that Generalized Solo Chess(♙,2) can be decided by an algorithm whose running time is linear in the number of pawns when all pawns are allowed to capture at most twice in the initial configuration.

▶ **Theorem 4** (A characterization for pawns). *Generalized Solo Chess (♙, 2) with n white pawns, each of which can capture at most twice, can be decided in $O(n)$ time.*

We denote by V the set of squares that initially contain a pawn, and by t the location of the target pawn. For $u, v \in V$, we say that u is a *parent* of v (and that v is a *child* of u) if u is diagonally one capture move away from v. We denote by $C(v)$ the children of v. Further if two vertices share a common parent, then we call them *siblings* of each other. We say that an initial configuration of pawns is *super-solvable* if the final capturing pawn has one move remaining after the final capture.

▶ **Definition 13.** *We say that a configuration of Generalized Solo Chess(♙, 2) with position set V is a skewed binary tree rooted at square v if the following are true:*

(a) *All pawns are on squares of the same color.*
(b) *All squares in $V \setminus \{v\}$ are below v.*
(c) *Every square in $V \setminus \{v\}$ has a parent in V.*
(d) *Every non-empty row below v contains exactly two squares of V with a common parent, except possibly the last (bottom-most) row which may contain one square of V.*

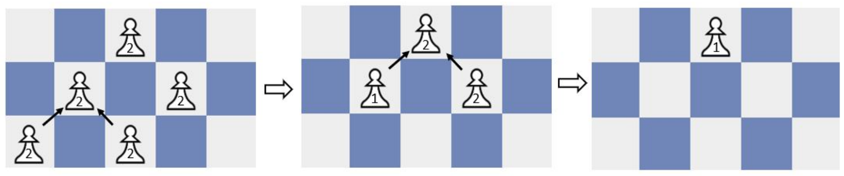


Figure 7 The initial configuration in this example is a skewed binary tree. Note that it is super-solvable because the shown sequence of moves clears the board such that the final pawn has one move left - here, the 2-pawn at the cell $(2, 4)$ does the final capture and becomes a 1-pawn.

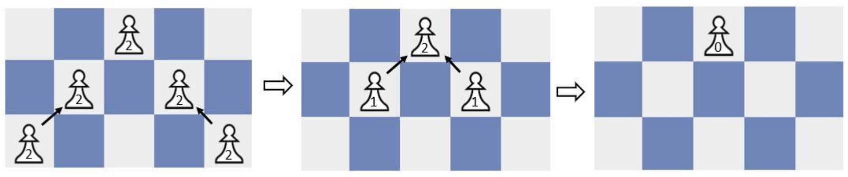


Figure 8 The initial configuration in this example is not a skewed binary tree. Note that it is not super-solvable because any sequence of moves that clears the board (one such sequence is shown) is such that the final pawn has no moves left.

The following result is the key to the characterization of solvable instances.

▶ **Lemma 14.** *An instance of* Generalized Solo Chess*(♙,2) is super-solvable if and only if the initial configuration is a skewed binary tree.*

In particular, we can verify in linear time whether a given configuration of Generalized Solo Chess(♙,2) is super-solvable, as each of the properties (a)-(d) can be checked in linear time.

We first observe that pawns can capture only in the forward direction (upward for W pawns) and only pawns on squares of the same color. Thus, we shall henceforth assume that all pawns are on squares of the same color and also that there is exactly one pawn whose initial square has the largest y co-ordinate; we shall call this the target pawn. If our assumption is false, we report the instance as a NO instance, and do not proceed further. We now describe the proof of Lemma 14.

Proof. We prove the claim by induction on $|V|$. If $|V| = 1$, the instance is trivially super-solvable and also satisfies the definition of a skewed binary tree. If $|V| = 2$, then the instance is super-solvable if and only if the unique vertex $v \in V \setminus \{t\}$ is a child of t, and this configuration is a skewed binary tree.

Thus, we suppose that $|V| \geqslant 3$. The necessity of conditions (a), (b), (c) has already been noted so that in the rest of this section we consider only configurations that satisfy (a), (b) and (c). We shall now establish the necessity of condition (d).

Firstly, we claim that t has two children. Suppose not, and let v be the only child of t. Then the last capture must be from v to t, and the last but one capture must be at v, so that the token at v has only one move remaining. When this token captures at t, it has zero moves left after the capture.

Thus, we can assume that t has two children u, v. Consider a valid super-solvable sequence σ; let u be the vertex from which the final capture was made at t. Then the token at u must have had two moves left before this capture and therefore no capture in σ was ever made at u. Also, the last but one capture in σ must have been made from v to t. This implies that the sequence obtained from σ by excluding the final capture is a valid super-solvable sequence for $V \setminus \{t, u\}$. Since $V \setminus \{t, u\}$ is super-solvable, by the induction hypothesis, property (d) holds; i.e. there are exactly two squares in every row below v, except possibly for the bottom-most non-empty row. This shows that property (d) holds for V as well.

For the other direction, suppose that a given configuration with V as the set of squares is a skewed binary tree, and that $|V| \geqslant 3$. Then by definition t has two children u, v and it must be the case that one of u, v, say u has no child outside $C(v)$. Then $V \setminus \{u, v\}$ must induce a skewed binary tree; let σ be a super-solvable sequence for $V \setminus \{u, v\}$. Appending the captures $u \to t, v \to t$ yields a super-solvable sequence for the original configuration.

This completes the proof of Lemma 14. ◀

We now proceed to the proof of Theorem 4.

Proof. Let V be the initial position set and t be the target square.

Case 1: t has a single child x. In this case, we note that the instance is solvable if and only if the configuration restricted to $V \setminus \{t\}$ with x as target is super-solvable, which by Lemma 14 in linear time.

Case 2: t has two children x, y, and one of them, say y, has no child other than the common child of x, y. In this case, the instance is solvable if and only if the configuration restricted to $V \setminus \{t, y\}$ with target x is super-solvable, which we can verify in linear time.

Case 3: t has two children x, y, and $|C(x) \cup C(y)| = 3$; let $C(x) \cup C(y) = \{a, b, c\}$. Then the instance is solvable if and only if there's a re-labeling u, v, w of $\{a, b, c\}$ such that $C(u) \cup C(v) \subseteq C(w)$ and the configuration restricted to $V \setminus \{t, x, y, u, v\}$ with target w is super-solvable. This can again be verified in linear time.

This completes the proof of Theorem 4. ◀

4 Graph Capture Game

We introduce a game on graphs, which generalizes GENERALIZED SOLO CHESS(♘,d) when played on undirected graphs and GENERALIZED SOLO CHESS(♙,d) when played on directed graphs.

GRAPH CAPTURE(G,d):

Input: A graph $G = (V, E)$.
Output: Decide if there exists a sequence of token captures (along the edges of G) such that only a single token remains, with the constraint that each token may capture at most d times.

Our main result in this section is the following:

▶ **Theorem 5** (Intractability of the graph capture game). *GRAPH CAPTURE(2) is NP-complete on undirected graphs and DAGs even when every token can capture at most twice.*

In the rest of this section, we say that G is solvable if GRAPH CAPTURE$(G, 2)$ is a YES-instance.

4.1 Undirected Graphs

We prove Theorem 5 for undirected graphs.

A rooted tree is a pair (T, v), with v denoting the root vertex; given a rooted tree (T, v) and a vertex w of T, we denote by $C(w)$ the children of w, and by $T(w)$ the subtree rooted at w.

▶ **Lemma 15.** *A graph $G = (V, E)$ is solvable if and only G contains a vertex v and a spanning tree T such that every internal node of the rooted tree (T, v) has a leaf neighbor.*

We prove Lemma 15 in the Appendix [1] and now turn to a proof of part (a) in Theorem 5.

Proof. We proceed by a reduction from Colorful Red-Blue Dominating Set. Let $\langle G = (R \uplus B, E); k\rangle$ be an instance of Colorful Red-Blue Dominating Set with color classes $V_1 \cup \cdots \cup V_k$. We assume, without loss of generality, that $|V_1| = \cdots = |V_k| = n$ and let $V_j := \{v_1^{(j)}, \ldots, v_n^{(j)}\}$. We begin by describing the construction of the reduced instance. We begin with the graph G and make the following additions:

1. For all $i \in [n]$ and $j \in [k]$, introduce a vertex $u_i^{(j)}$ and make it adjacent to $v_i^{(j)}$. We call these the *red partner vertices.*
2. For each $u_i^{(j)}$, introduce two neighbors $p_i^{(j)}$ and $q_i^{(j)}$, and finally, introduce two vertices $r_i^{(j)}$ and $s_i^{(j)}$ that are adjacent only to $p_i^{(j)}$ and $q_i^{(j)}$ respectively. In other words, each $u_i^{(j)}$ has two degree two neighbors, which in turn have a leaf neighbor each. Combined, we refer to the collection of vertices $S_j := \{u_i^{(j)}, p_i^{(j)}, q_i^{(j)}, r_i^{(j)}, s_i^{(j)} \mid i \in [n]\}$ as the *selection gadget* for V_j.
3. For all $j \in [k]$, introduce a vertex w_j and make it adjacent to $u_i^{(j)}$ for all $i \in [n]$. We call these vertices the *guards.*
4. We finally add a vertex $\star$ that is adjacent to all the red partner vertices. We also add the vertices p and q and the edges $(\star, p)$ and (p, q).

We let H denote the graph thus constructed based on G and ask if H has a rooted spanning tree for which every internal note has a leaf neighbor. This completes a description of the construction. We briefly describe the intuition for the equivalence of the two instances. Because of the vertices selection gadgets, the partner vertices are forced to find their leaf neighbors in any spanning tree among the red vertices that they partner – except for at most one, which can use the guard vertex as the leaf neighbor. This leads to one red vertex being left "free" of being a leaf neighbor to a partner vertex in each color class, hence we have a selection of one blue vertex per color class. Since these are the only possible entry points for the blue vertices into the spanning tree, the vertices "chosen" by the selection gadget must correspond to a dominating set. We now formalize this intuition.

Forward Direction

Assume that $\langle G = (R \uplus B, E); k\rangle$ is a YES instance. That is, there exist $1 \leqslant j_1, \ldots, j_k \leqslant n$ such that every vertex in B has a neighbour in $\{v_{j_1}^{(1)}, \ldots, v_{j_k}^{(k)}\}$. For each $b \in B$, let $f(b)$ denote an arbitrary but fixed $\ell \in [k]$ such that $v_{j_\ell}^{(\ell)} \in N(b)$.

Let T denote the spanning tree of H with the following edge set:

- T contains all the edges of $H\left[\{\star, p, q\} \cup R \cup \bigcup_{1 \leqslant \ell \leqslant k} S_\ell\right]$
- For every $b \in B$, T contains the edge $\{v_{j_{f(b)}}^{(f(b))}, b\}$
- For every $1 \leqslant \ell \leqslant k$, T contains the edge $\{u_{j_\ell}^{(\ell)}, w_\ell\}$

Note that every internal node of the rooted tree $(T, \star)$ has a leaf neighbour.

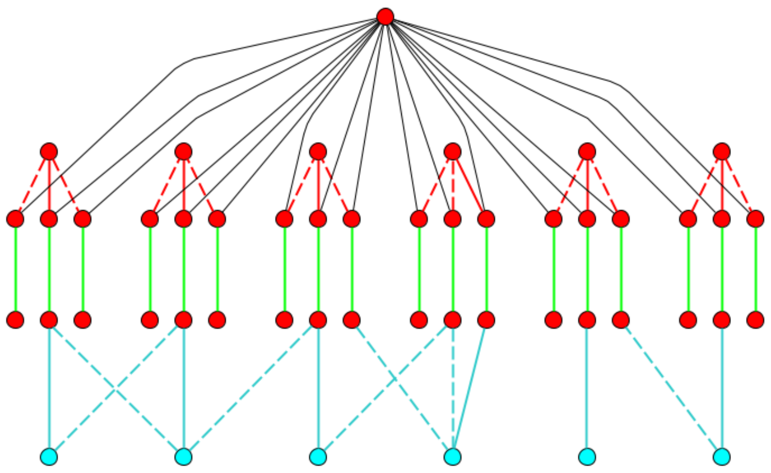


Figure 9 An illustration of the reduction from Red-Blue dominating set. The solid lines belong to the spanning tree. The "spikes" from the selection gadget are omitted for clarity.

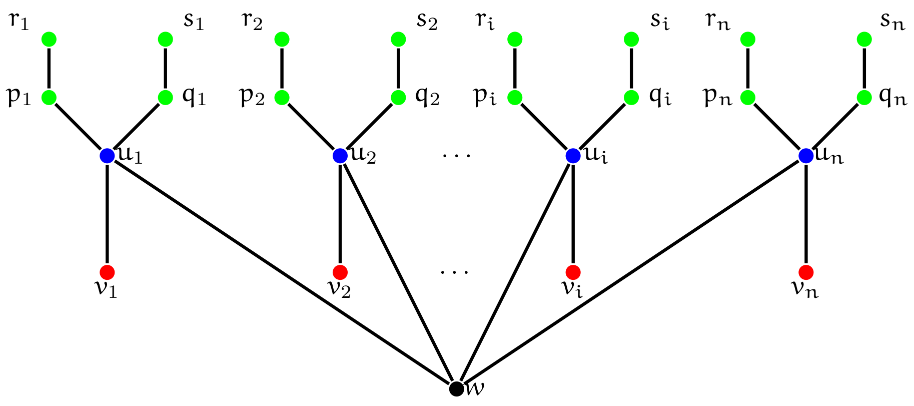


Figure 10 A schematic showing the selection gadget for one color class.

Reverse Direction

Assume that there exist $r \in V(H)$ and a spanning tree (say T) of H such that every internal node of the rooted tree (T, r) has a leaf neighbour. In T, $\star$ is adjacent to p and at least one red partner vertex. So, $\star$ has at least two neighbours in T. Thus, $\star$ is not a leaf node of (T, r).

The only neighbours of $\star$ in T are p and some vertices of $\{u_i^{(\ell)} \mid 1 \leqslant \ell \leqslant k, 1 \leqslant i \leqslant n\}$. Note that p is not a leaf node of (T, r) as p has two neighbours, i.e., $\star$ and q, in T. Also, for every $1 \leqslant \ell \leqslant k$ and every $1 \leqslant i \leqslant n$, $u_i^{(\ell)}$ is not a leaf node of (T, r) because $u_i^{(\ell)}$ has at least two neighbours, i.e., $p_i^{(\ell)}$ and $q_i^{(\ell)}$, in T. Therefore, $\star$ has no leaf neighbours in (T, r). So, $\star$ is not an internal node of (T, r). Hence, we have $r = \star$.

Let $1 \leqslant \ell \leqslant k$. The only neighbours of w_ℓ in T are some vertices of $\{u_i^\ell \mid 1 \leqslant i \leqslant n\}$. As argued earlier, no red partner vertex is a leaf node of (T, r). So, w_ℓ has no leaf neighbours in (T, r). Thus, w_ℓ is not an internal node of (T, r). That is, w_ℓ is a leaf node of (T, r). Hence, there exists a unique integer (say $g(\ell)$) in $[n]$ such that $u_{g(\ell)}^{(\ell)}$ is the neighbour of w_ℓ in T.

Now, it suffices to show that every vertex in B has a neighbour in $\{v_{g(1)}^{(1)}, \dots, v_{g(k)}^{(k)}\}$. Let $b \in B$. There exist $1 \leqslant \ell \leqslant k$ and $1 \leqslant i \leqslant n$ such that $v_i^{(\ell)} \in N_T(b)$. As shown above, $u_i^{(\ell)}$ is not a leaf node of (T, r). That is, $u_i^{(\ell)}$ is an internal node of (T, r). So, $u_i^{(\ell)}$ has a leaf neighbour (say z) in (T, r). No vertex of $V(T) \setminus \{p_i^{(\ell)}, q_i^{(\ell)}, \star, w_\ell, v_i^{(\ell)}\}$ is adjacent to $u_i^{(\ell)}$ in T. Note that

- $p_i^{(\ell)}$ is not a leaf node of (T, r) as $p_i^{(\ell)}$ has at least two neighbours, i.e., $u_i^{(\ell)}$ and $r_i^{(\ell)}$, in T.
- $q_i^{(\ell)}$ is not a leaf node of (T, r) as $q_i^{(\ell)}$ has at least two neighbours, i.e., $u_i^{(\ell)}$ and $s_i^{(\ell)}$, in T.
- If $v_i^{(\ell)}$ is a neighbour of $u_i^{(\ell)}$ in T, then $v_i^{(\ell)}$ is not a leaf node of (T, r) because in such a case, $v_i^{(\ell)}$ has at least two neighbours, i.e., b and $u_i^{(\ell)}$, in T.

Therefore, we have $z = w_\ell$ and hence, $i = g(\ell)$.

This completes the argument for equivalence. ◀

4.2 Directed Acyclic Graphs

We now prove Theorem 5 for DAGs.

Proof. Let φ be a given instance of 3-SAT, with clauses $C_1, C_2, \dots, C_m$ over variables $x_1, x_2, \dots, x_n$.

We construct the following directed graph $G = (W, E)$, where $W = U \cup V \cup \{w\}$; $U = \{u_1, u_2, \dots, u_m\}$ and $V = \{v_1, v_2, \dots, v_n\} \cup \{v_1^T, v_2^T, \dots, v_n^T\} \cup \{v_1^F, v_2^F, \dots, v_n^F\}$. Intuitively, each vertex in U represents a clause and vertices v_i^T, v_i^F correspond to an assignment of T, F respectively to x_i.

The edge set is $E = E_1 \cup E_2$, where

$$\begin{aligned} E_1 = &\{(u_i, v_j^T) | x_j \in C_i, 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\} \\ &\cup \\ &\{(u_i, v_j^F) | \neg x_j \in C_i, 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n\} \end{aligned}$$

and

$$E_2 = \{(v_i^T, v_i) | 1 \leqslant i \leqslant n\} \quad \cup \quad \{(v_i^F, v_i) | 1 \leqslant i \leqslant n\} \quad \cup \quad \{(v_i, w) | 1 \leqslant i \leqslant n\}.$$

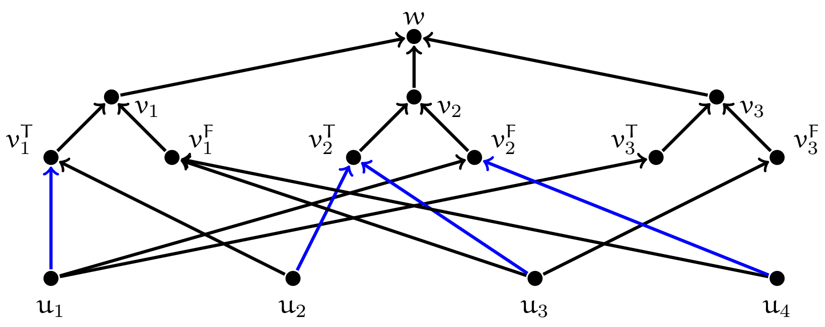


Figure 11 The DAG corresponding to the set of clauses $C_1 = \{x_1, \neg x_2, x_3\}, C_2 = \{x_2, x_3\}, C_3 = \{\neg x_1, x_2, \neg x_3\}, C_4 = \{\neg x_2, \neg x_3\}$. The blue edges indicate the captures in Phase 1 for the satisfying assignment $x_1 = T$, $x_2 = T$, $x_3 = F$.

The graph G can clearly be computed in time polynomial in the input size (number of variables and clauses).

It thus suffices to show that φ is satisfiable iff G is solvable.

First, suppose that φ is satisfiable and let A be a satisfying assignment for φ. Consider the following sequence of captures:

Phase 1: For each $i \in \{1, 2, \ldots, m\}$, let j be the least index such that C_i is satisfied by x_j or $\neg x_j$ in A. Then the token at u_i captures the token at v_j^T (if $x_j = T$) or the token at v_j^F (if $x_j = F$).

Phase 2: Now, for each $i \in \{1, 2, \ldots, n\}$: if $x_i = T$, then the token at v_i^T captures the token at v_i, and then the token at v_i^F captures the token at v_i; otherwise $v_i = F$ and the token at v_i^F captures the token at x, and then the token at v_i^T captures the token at v_i;

Phase 3: For each $i \in \{1, 2, \ldots, n\}$, the token at v_i captures the token at w.

We show the validity of this capture sequence by considering each phase.

At the end of Phase 1, there are no tokens remaining at any of the u_is, and further for each $i \in \{1, 2, \ldots, n\}$, exactly one of the tokens among $\{v_i^T, v_i^F\}$ has 1 move remaining, and the other token has 2 moves remaining.

In Phase 2, the token among $\{v_i^T, v_i^F\}$ with 2 moves remaining is the last to capture at v_i; thus at the end of Phase 2, there is one token at each v_i with one move remaining and further one more token at w; there are no other tokens.

In Phase 3, it is thus feasible for each token at v_i to successively capture at w and finally there is exactly one token remaining - at the vertex w.

Now, we suppose that G is solvable; let σ be a valid sequence of captures. We claim that for each $i \in \{1, 2, \ldots, n\}$, captures were not made at both v_i^T and v_i^F in σ. For contradiction, suppose that captures were made both at v_i^T and at v_i^F; let the last of these captures be at move t_1 of σ. Since all the tokens in $\{v_i^T, v_i^F\}$ must make their last capture at v_i, there must be a capture from $\{v_i^T, v_i^F\}$ to z that appears in σ later than t; let the last such capture happen at move $t_2 > t_1$. After move t_2, there are no tokens in $\{v_i^T, v_i^F\}$ and the token at v_i has zero moves remaining; this implies that the token at v_i cannot be cleared, which is the desired contradiction.

Now, let I be the set of indices i such that a capture was made at v_i^T. Consider the assignment: for each $i \in \{1, 2, \ldots, n\}$, we set $x_i = T$ if $i \in I$ and $x_i = F$ otherwise. We claim that every clause is satisfied by this assignment. Let C_i be an arbitrary clause; if the token at u_i made a capture at some v_j^T, then C_i contains the literal x_j, and $j \in I$ so that $x_j = T$

and C_i is satisfied. If the token at u_i made a capture at some v_j^F, then C_i contains the literal $\neg x_j$. Also, by the claim in the previous paragraph, no capture was made at v_j^T, therefore $j \notin I$ and $x_j = F$, so that C_i is satisfied. ◀

5 Concluding Remarks

We introduced Generalized Solo Chess based on the Solo Chess game that is played on a 8×8 board. We focused mostly on scenarios that involve only pieces of one type, and showed that determining if a given instance is solvable is intractable when playing with rooks, bishops, and queens; while it is tractable for pawns. While we leave the case of knights open, we do show that a natural generalization of Generalized Solo Chess restricted to knights, Graph Capture, is hard even on DAGs and general undirected graphs. We also show that solvable instances of Generalized Solo Chess played by rooks only admits an efficient characterization when the game is restricted to one-dimensional boards.

Our work leaves open a few concrete open problems, which we enlist below:

1. What is the complexity of Generalized Solo Chess played by rooks only, for the special case when all rooks are allowed to capture at most twice initially? Notice that if we replace rooks by queens in this question, we show NP-completeness (Theorem 3).
2. What is the complexity of Generalized Solo Chess played by pawns only, when the pawns are allowed at most a designated number of captures? Recall that if all pawns can capture at most twice, we have an efficient characterization (Theorem 4).
3. What is the complexity of Generalized Solo Chess played by knights only?

There are also several broad directions for future work, and we suggest some that we think are both natural and interesting problems to consider:

1. For NO-instances of Generalized Solo Chess, a natural optimization objective is to play as many moves as possible, or, equivalently, leave as few pieces as possible on the board. It would also be interesting to find the smallest d for which a board can be cleared if every piece was allowed to capture at most d times. These are natural optimization versions that we did not explicitly consider but we believe would be interesting to explore.
2. Does Generalized Solo Chess become easier if the number of pieces in every row or column is bounded? Note that the reduction in Theorem 2 can be used to show that Generalized Solo Chess(♖, 2) is NP-complete even when the number of rooks per column is a constant, if we initiate the reduction from an instance of Red-Blue Dominating Set where every red vertex has constant degree.
3. What is the complexity of Generalized Solo Chess when played on boards of dimension $M \times N$, where one of M or N is a constant? It is not hard to generalize Theorem 1 to $2 \times N$ boards, however, a general result – say parameterized by one of the dimensions – remains open.
4. Variants of Generalized Solo Chess where the pieces are limited not by the number of captures but the total distance moved on the board, or the distance moved per step, are also interesting to consider. Note that if the distance moved per step is lower bounded, then this forbids "nearby captures", while if it is upper bounded, then "faraway captures" are disallowed.
5. It would also be interesting to restrict the number of capturing pieces instead of the number of captures per piece. For example, it seems intuitive to posit that if we are permitted only one capturing piece, then it must trace a "Hamiltonian path" of sorts among the pieces on the board.

6. Finally, we did not explicitly study Generalized Solo Chess with multiple piece types on the board – although most such variants would be hard given the complexity results established already for pieces of one kind, it would be interesting to investigate special cases and exact algorithms in this setting.

References

1 N. R. Aravind, Neeldhara Misra, and Harshil Mittal. Chess is hard even for a single player, 2022. doi:10.48550/ARXIV.2203.14864.

2 Elwyn R Berlekamp, John H Conway, and Richard K Guy. *Winning ways for your mathematical plays*. A K Peters, Natick, MA, 2 edition, January 2001.

3 Josh Brunner, Erik D Demaine, Dylan Hendrickson, and Julian Wellman. Complexity of Retrograde and Helpmate Chess Problems: Even Cooperative Chess is Hard. *arXiv preprint arXiv:2010.09271*, 2020.

4 Marek Cygan, Fedor V Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michał Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*, volume 5. Springer, 2015.

5 Aviezri S Fraenkel and David Lichtenstein. Computing a Perfect Strategy for n × n Chess Requires Time Exponential in n. *Journal of Combinatorial Theory, Series A*, 31(2):199–214, 1981.

6 Michael R Garey and David S Johnson. *Computers and Intractability*, volume 174. freeman San Francisco, 1979.

7 Douglas Brent West et al. *Introduction to Graph Theory*, volume 2. Prentice hall Upper Saddle River, 2001.