Int J Game Theory
https://doi.org/10.1007/s00182-018-0628-8

ORIGINAL PAPER

PSPACE-complete two-color planar placement games

Kyle Burke[1] · **Robert A. Hearn**[2]

Accepted: 30 April 2018

Abstract We show that the classic placement games COL and SNORT are PSPACE-complete, resolving an open question of Schaefer (1978). We then show the related placement games FJORDS and NOGO PSPACE-complete on planar graphs. All but NOGO are shown hard by reductions from BOUNDED 2-PLAYER CONSTRAINT LOGIC; we then reduce COL to NOGO. The only previous complexity results for these games were that COL and SNORT played on general graphs are PSPACE-complete, and NOGO is NP-hard on general graphs.

Keywords Complexity · PSPACE · Combinatorial game

1 Introduction

Conway (2001) introduced the games COL (invented by Colin Vout) and SNORT (invented by Simon Norton). These games are in some sense opposites: in COL, players take turn placing tokens of their own color on vertices of a planar graph (originally, painting regions on a map); adjacent vertices may not hold the same color. The goal is to make the last move. SNORT is the same, but adjacent vertices may not hold *different* colors. Both games have been studied in the context of Combinatorial Game Theory, and from a complexity perspective. Schaefer (1978) showed that SNORT played on general graphs is PSPACE-complete, and noted that "the complexity of the game

✉ Robert A. Hearn
bob@hearn.to

Kyle Burke
kwburke@plymouth.edu

[1] Plymouth State University, Plymouth, USA

[2] Portola Valley, CA, USA

Published online: 13 July 2018

restricted to planar graphs is of obvious interest". While preparing this submission, we discovered that COL played on general graphs was recently shown PSPACE-complete as well (Fenner et al. 2015). The hardness of COL is especially interesting, because COL is a "cold" game—each player can only worsen their position by making the next move. Normally cold games are not hard.

We completely resolve the status of COL and SNORT, played on planar graphs as originally defined. It is remarkable that the complexity of these well-known and extremely natural graph games has remained open for so long.[1]

COL and SNORT are part of a larger family of *placement games*, as are FJORDS and NOGO, defined below. FJORDS is an award-winning board game produced by Rio Grande Games, played on a hexagonal grid. Here we consider a version played on arbitrary planar graphs. NOGO is a simple variant of GO, played on a square grid, where capturing is not allowed. NOGO was first studied by combinatorial game theorists, but has since been an AI testbed at recent Computer Olympiads.[2] Again, we consider the version played on planar graphs, leaving open the complexity of both of these games played on their standard grid graphs.

We show that all of these games are PSPACE-complete, indicating that we should not expect a general theory for any of them. Our reductions are all (directly or indirectly) from BOUNDED 2- PLAYER CONSTRAINT LOGIC, which has become a useful tool for showing two-player bounded-length games hard, especially where planarity is important.

In Sect. 2 we provide background on Combinatorial Game Theory, placement games, and Algorithmic Combinatorial Game Theory. In Sect. 3 we discuss Constraint Logic. In Sect. 4 through 7 we formally define each game and present our hardness reductions.

2 Background

2.1 Combinatorial game theory

Combinatorial Game Theory is the study of games with:

- Two players alternating turns,
- No randomness, and
- Perfect information for both players.

A *ruleset* describes which moves each player can make from any game position. The games we study in this paper use *normal play* rules, meaning if a player cannot make a move on their turn, they lose the game (the last player to move wins).

The two players are commonly known as *Left* and *Right*, or *Blue* and *Red*, or *Black* and *White*. Here, for uniformity we will use Black and White. If both players always have the same move options, the game is called *impartial*; otherwise it is *partisan*. All the games we consider here are partisan.

无偏和有偏博弈

[1] COL was mistakenly cited as PSPACE-complete in Berlekamp et al. (2001) and, apparently as a result, in Cincotti (2009). Both cite (Schaefer 1978), which analyzes SNORT, but not COL.

[2] https://www.game-ai-forum.org/icga-tournaments/game.php?id=47

For more information on combinatorial game theory, the reader is encouraged to look at Berlekamp et al. (2001) and Albert et al. (2007).

2.2 Placement games

A *placement game*, as defined in [3], is a combinatorial game ruleset played on a graph, $G = (V, E)$, that fulfills all of these requirements:

- Vertices are either marked (perhaps by a specific color) or unmarked,
- A move for a player consists of marking exactly one unmarked vertex, and
- Marks may never be moved or removed.

2.3 Algorithmic combinatorial game theory

Algorithmic Combinatorial Game Theory is the application of algorithms to combinatorial games. The difficulty of a ruleset is analyzed by studying the computational complexity of determining whether the current player has a winning strategy. In this paper, we choose White to always be the next player when posing this question.[3] We show that many games are PSPACE-complete, which means that no polynomial-time algorithm exists to determine the winnability of all positions unless such an algorithm exists for all PSPACE problems.

Usually determining the winnability of a ruleset is considered as the computational problem of the same name. We continue to (ab)use that language here: saying for example that GAME X is PSPACE-complete means that the associated winnability decision problem is PSPACE-complete.

The four rulesets we have results for are all placement games. Because marks may never be moved or removed:

一些基础内容

- The maximum number of turns taken during a game is always $n = |V|$.
- The maximum number of options on one turn is always $\leq n$.

Because these are both polynomial in the size of the graph, the decision problem for each of these games is in PSPACE—we can perform a depth-first search of the game tree using polynomial space. Thus, by showing that these games are PSPACE-hard, we also show that they are PSPACE-complete.

For more on algorithmic combinatorial game theory, the reader is encouraged to look at Demaine and Hearn (2009).

3 Constraint logic

3.1 Definition

Constraint Logic is a family of games played on edge-weighted directed graphs, parametrized by number of players (0, 1, 2, team) and whether the game is of bounded

[3] We do this for consistency with established Constraint Logic conventions, and ease of presentation—normally Black, or Left, is the first player in a combinatorial game.

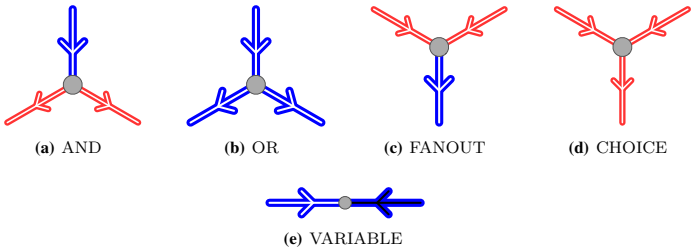


Fig. 1 Basis vertices for Bounded 2CL

or unbounded length, with each specific version complete for its "natural" complexity class (Demaine et al. 2008; Hearn and Demaine 2009). It has proven very useful in showing games and puzzles hard, especially those for which planarity is important.

In a constraint-logic game, a move is to reverse the orientation of an edge in the graph, consistent with constraints at the incident vertices: the sum of weights of inward-directed edges at each vertex (the *inflow*) must total at least a minimum threshold. The game or puzzle is won when a given edge is reversed. Constraint-logic games are hard on planar graphs, even for very restricted sets of vertex types. The games we consider here are two-player, bounded-length games; the corresponding type of constraint logic is called BOUNDED 2- PLAYER CONSTRAINT LOGIC (BOUNDED 2CL). This game has been used to show board games such as AMAZONS and KONANE PSPACE-complete (Hearn 2008).

In BOUNDED 2CL, each edge is controlled by either White or Black, and each player has a distinct goal edge they need to reverse to win. Each edge is only allowed to be reversed once; therefore the game is of (polynomially) bounded length. BOUNDED 2CL is PSPACE-complete even when the graph is planar, and the vertices are all one of five types: AND, OR, FANOUT, CHOICE, or VARIABLE, shown in Fig. 1, with their initial edge orientations.[4] In these vertices, red (light gray) edges denote weight 1, blue (dark gray) edges weight 2, and the minimum inflow constraint is 2. The edges are shown filled with the color that controls them. The AND vertex, for example, needs either the blue edge or both red edges directed inward to satisfy its inflow constraint. If White reverses both red edges inward, then the blue edge may be reversed outward—effectively this is a logical AND of the "inputs". OR works similarly. FANOUT allows a "signal" to be split, and CHOICE gives White an option of selecting one "output" or the other to reverse if the "input" is first reversed. The only vertex type where player interaction can occur is the VARIABLE. Whoever plays first in a VARIABLE prevents the other player from moving there, because that would violate the inflow constraint. Note that Black's moves occur only in VARIABLE gadgets.

The proof that BOUNDED 2CL itself is PSPACE-hard is a trivial reduction from a standard game played on Boolean formulas, G_{pos}(POS CNF) (Hearn and Demaine 2009). The utility of reducing from BOUNDED 2CL to show other games hard, rather than directly from a formula game, is that Constraint Logic has a generic crossover

一般的

[4] We allow "half edges" in our graph, that is, edges connected to only one vertex—there is no way to connect the Black edge in a VARIABLE vertex to any other vertex. These are irrelevant to our reductions.

construction, meaning that all hardness results automatically apply when the graph is planar. If we reduced from a formula game, we would need to find a way to explicitly cross signals for each target game. Though not stated in Hearn and Demaine (2009), it is clear from the G_{pos}(POS CNF) reduction that we may assume the constraint graph to be acyclic.

Lemma 1 BOUNDED 2CL *remains PSPACE-complete when restricted to acyclic constraint graphs.*

Proof By inspection of the reduction employed in (Hearn and Demaine 2009, p. 58) □

检查

For the remainder of the paper, by BOUNDED 2CL we refer to the version restricted to planar, acyclic constraint graphs using only the above vertex types.

3.2 Reducing from constraint Logic

We reduce from BOUNDED 2CL to prove the hardness of COL, SNORT, and PLANAR-FJORDS. For each reduction, we must implement the five vertex gadgets shown in Fig. 1, show how to connect them together, and ensure that our game can be won if and only if the corresponding constraint-logic game can be won. In order to avoid confusion, for the rest of the paper we will always write "constraint-graph vertex" or "constraint-graph edge" for the BOUNDED 2CL elements; otherwise "vertex" and "edge" refer to elements in the target game.

教条的

We refer here and below to *canonical play*, indicating the way that gadgets "should" be played in. In every case we show that deviation from canonical play cannot benefit a player. (If they don't have a winning strategy following canonical play, then they don't have any winning strategy.)

偏离

In all three reductions, we represent a constraint-graph edge state with an *I/O gadget*, with the following properties:

- Each I/O gadget contains an *Active* vertex and an *Inactive* vertex.
- Under canonical play, White will play either the Active or the Inactive vertex, but not both, in every I/O gadget. (Black may not play in all I/O gadgets, depending on the reduction.)
- An I/O gadget where White has played in the Active vertex is said to be *activated*; if White plays in the Inactive vertex, or Black plays in the Active vertex, it is *inactivated*. An activated I/O gadget represents the corresponding constraint-graph edge having flipped from its initial orientation.

We only explicitly represent White-controlled constraint-graph edges; Black's moves in VARIABLE gadgets merely serve to block White, and are not connected to anything else. Indeed, for all three reductions, the I/O gadget also serves as a VARIABLE gadget. We also do not need to explicitly represent Black's goal edge, because White will block Black's win on his first move (Black's goal edge must be in a VARIABLE gadget), and the only question is whether White can win (Hearn and Demaine 2009). Note that the rules allow for no winner; BOUNDED 2CL is not strictly a combinatorial game under normal play, where the last play wins.

Fig. 2 Generic I/O gadget.
A = Active; I = Inactive

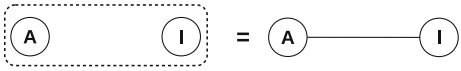


Fig. 3 COL I/O gadget and VARIABLE

The details of the I/O gadget will vary depending on the reduction, but we will use the abstraction shown in Fig. 2 in the constructions for the constraint-graph vertex gadgets to hide the details.

4 COL

COL is a partisan placement game where players alternately mark vertices with their color (Black or White) with the restriction that two neighboring vertices may not have the same color. Thus a move consists of coloring an uncolored vertex not adjacent to another vertex of the player's color. Formally:

Definition 1 (COL) COL is a ruleset played on a planar graph, $G = (V, E, c)$, where c is a coloring of vertices $c\colon V \to \{Black, White, Uncolored\}$ such that $\forall(v, w) \in E\colon$ either $c(v) \neq c(w)$ or $c(v) = c(w) = Uncolored$. An option for player A is a graph $G' = (V, E, c')$ where $\exists x \in V$:

- $\forall v \neq x \in V : c'(v) = c(v)$, and
- $c(x) = Uncolored$, and
- $c'(x) = A$'s color, and
- $\forall(y)$ such that $(x, y) \in E : c(y) \neq c'(x)$.

The computational complexity of COL has remained an unsolved problem since its introduction in the 1970s. We show that COL is PSPACE-complete, even for planar graphs. As remarked in the introduction, it is exceptional for a cold game to be hard.

Gadgets

COL is in PSPACE, as shown in Sect. 2.3. To show hardness, we reduce from BOUNDED 2CL. We need to specify the details of the COL I/O gadget, and create gadgets corresponding to each of the constraint-logic vertex types in Figure 1. We will also need a "goal" gadget attached to White's constraint-graph goal edge, to give White the extra move needed to win the COL game just when White can flip that edge in the BOUNDED 2CL game. Each vertex gadget gives White and Black an equal number of moves under canonical play.

The I/O gadget, shown in Fig. 3, includes a single edge connecting the pair of vertices. This prevents either player from being able to choose both (because two same-colored vertices may not be adjacent). White will need to play either the Active or the Inactive vertex in every I/O gadget to win. Black can only play in the I/O gadgets in VARIABLEs; Black moves in all other output I/O gadgets are blocked by an adjoining black vertex.

The VARIABLE gadget is simply a single I/O gadget. White activates it by playing in its Active vertex, effectively flipping the White output constraint-graph edge; Black

Fig. 4 AND gadget for COL

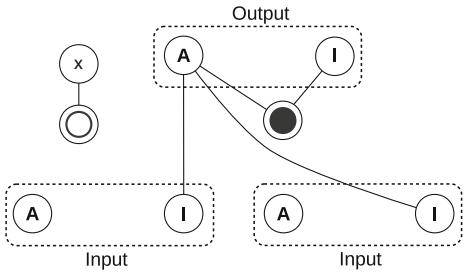
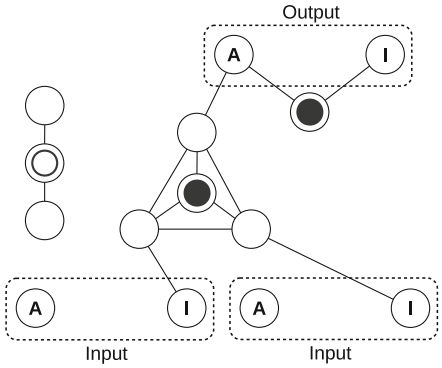


Fig. 5 OR gadget for COL



inactivates it by playing there. It can never help White to choose not to flip a VARIABLE output; doing so only eliminates potential paths to the goal edge. Therefore we may assume the players move as described. Later the Inactive vertices will be played as well, by the opposite player in each case.

The AND gadget, shown in Fig. 4, connects two input I/O gadgets to an output. (Here and in the remaining figures, a filled circle in a vertex denotes a Black token, an empty circle denotes a White token, and characters are labels.) The Active vertex of the output is connected to each input's Inactive vertex; this prevents White from activating the output if either input is inactivated. In order to make up for Black's free play on the disjoint piece (White cannot play at x, because it is adjacent to a White vertex), White must either activate or inactivate the output.

The OR gadget, shown in Fig. 5, connects three I/O gadgets (two input and one output) to a 4-clique where one vertex is colored Black. If either of the inputs is activated, then White can activate the output and also make an extra move in the clique. Black is given two free moves in the disjoint piece; White must play in both the output and the center-clique in order to have any chance to win.

The FANOUT gadget, shown in Fig. 6, connects one input I/O gadget to two outputs. Both Active vertices of the outputs are connected to the Inactive vertex of the input. If the input is activated, then the outputs may be activated. White plays in both outputs, making up for the two free moves provided to Black in the disjoint section.

The CHOICE gadget, shown in Fig. 7, is exactly the same as the FANOUT gadget, except that the Active vertices of the outputs are connected. Thus, White cannot play in both of them.

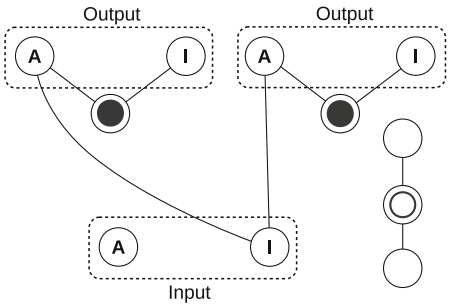


Fig. 6 FANOUT gadget for COL

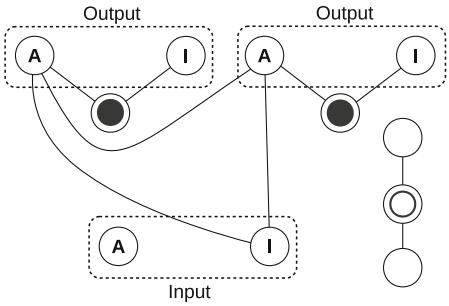


Fig. 7 CHOICE gadget for COL

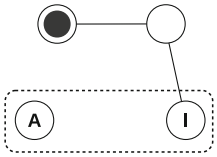


Fig. 8 GOAL gadget for COL

The GOAL gadget, shown in Fig. 8, is attached to the I/O gadget corresponding to White's goal edge in the constraint graph. White can play in the uncolored vertex only if the I/O gadget is not inactivated.

Theorem 1 *Determining whether the next player in* COL *has a winning strategy is PSPACE-complete, even when the maximum vertex degree is five.*

Proof Given a BOUNDED 2CL instance, we construct a corresponding COL instance as described above, identifying input and output I/O gadgets between appropriate vertex gadgets. No vertex in the construction has degree more than five: some Active vertices have three connections on the input side, but none has more than a single connection on the output side. The connection to the paired Inactive vertex makes five. Similarly, Inactive vertices have degree at most four, as do the internal vertices in the gadgets.

Each vertex gadget reproduces the behavior of its corresponding constraint-graph vertex, but only when inputs are played before outputs; we have no mechanism here to enforce that play order. However, it never benefits White to play an output before an input, as all I/O gadgets will eventually have to be activated or inactivated for White to have enough moves to win. So, White cannot cheat by for example activating an

AND output and later deactivating its inputs. We can therefore assume that all moves are made in an order corresponding to legal BOUNDED 2CL play.

Therefore, White can win the BOUNDED 2CL game just when White can win the corresponding COL game, by activating a sequence of outputs reaching the GOAL gadget, and playing in it. □

5 SNORT

SNORT is a partisan placement game where players alternately mark vertices with their color (Black or White) with the restriction that two neighboring vertices may not have *different* colors. Thus a single turn consists of coloring an uncolored vertex not adjacent to a vertex of the other player's color. Formally:

Definition 2 (SNORT) SNORT is a ruleset played on a planar graph, $G = (V, E, c)$, where c is a coloring of vertices $c\colon V \to \{Black, White, Uncolored\}$ such that $\forall (v, w) \in E\colon$ either $c(v) = c(w)$ or at least one of $\{c(v), c(w)\} = Uncolored$. An option for player A is a graph $G' = (V, E, c')$ where $\exists x \in V$:

- $\forall v \neq x \in V : c'(v) = c(v)$, and
- $c(x) = Uncolored$, and
- $c'(x) = A$'s color, and
- $\forall (y)$ such that $(x, y) \in E : c(y) = Uncolored \vee c(y) = c'(x)$.

SNORT has been known to be PSPACE-complete on general graphs since the early days of computational complexity (Schaefer 1978). We improve upon that by using BOUNDED 2CL to show that SNORT is still hard on planar graphs.

Move ordering

For COL, we did not need to explicitly force gadget inputs to be played before outputs, because Black was prevented from playing in any outputs and inappropriately blocking White, and White could gain no advantage playing out of order. For Snort (and also below for PLANAR- FJORDS), our reduction does need to ensure Black has no interest playing out of canonical order, so we define an incentive mechanism, described below, to enforce correct order of play. 激励

The constraint-graph vertex gadgets we construct force White to play in the output I/O gadgets in such a way as to mimic the constraint-graph vertex properties. If White deviates, Black can get some extra moves; any extra moves are enough to prevent White from winning. However, the canonical play order lets White play first in every I/O gadget apart from VARIABLES; if Black deviates from canonical play and plays first in an I/O gadget, they can force White to later play incorrectly in that gadget, inappropriately giving Black extra moves. We prevent this from happening by making out-of-order plays cost more moves than can be thus gained.

We topologically order all I/O gadgets, so that inputs occur before outputs for all constraint-graph vertex gadgets, which we can do because the constraint graph is acyclic, as mentioned in Sect. 3. Each play in either the Active or the Inactive vertex in an I/O gadget grants the player t additional moves available to them, and not to the

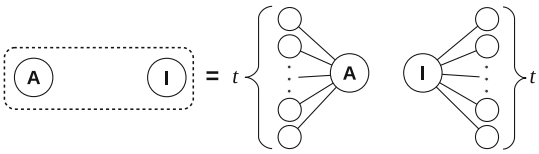


Fig. 9 SNORT I/O gadget and VARIABLE

other player, for some t that decreases by a sufficiently large constant per I/O gadget in the topological order. We give all VARIABLE gadgets the same (maximal) incentive; we can't force the order of play in them and respect the BOUNDED 2CL rules.

If there are n I/O gadgets, and the largest gain Black can get as a result of White playing incorrectly in one is k moves, then spacing the incentives apart by $nk + 1$ is sufficient. (We can use $k = 4$ here, as no gadget gives Black more than 4 moves if White plays incorrectly.) If Black "skips ahead" to play first in a lower-incentive I/O gadget, White can win simply by playing any moves in incentive order. Black can gain at most nk moves, but has already lost at least $nk + 1$. The I/O gadget with the GOAL gadget attached has an incentive of 0; each preceding I/O gadget in the topological order gains an additional incentive of $nk + 1$. The total number of incentive moves we must accommodate in our graphs is then $O(n^3)$, so our reductions remain polynomial.

常用技巧

容纳

Gadgets

SNORT is in PSPACE, as shown in Sect. 2.3. To show hardness, we need to specify the details of the SNORT I/O gadget, and create gadgets corresponding to each of the constraint-logic vertex types in Fig. 1. We will also need a "goal" gadget attached to White's constraint-graph goal edge, to give White the extra move needed to win the SNORT game just when White can flip that edge in the BOUNDED 2CL game. Each vertex gadget gives White and Black an equal number of moves under canonical play.

I/O gadgets, shown in Fig. 9, contain incentives: t extra vertices each attached to the Active and Inactive vertices. A play in either vertex grants that player t additional moves available only to them, because adjacent vertices may not contain different colors. We represent an I/O gadget with an abstraction that hides the extra vertices.

As in the COL reduction, a VARIABLE gadget is simply an I/O gadget. White activates it by playing in the Active vertex; Black inactivates it by doing so, forcing White to later play in the Inactive vertex. It is possible for one player to play in both the Active and Inactive sides in a single variable, but never advantageous to do so (White only benefits from activating variables, and Black from inactivating); we therefore assume that each plays in each VARIABLE.

After all the VARIABLEs have been played, canonical play is for White to play in the I/O gadget with the highest remaining incentive, and for Black to make the other play in this gadget; then White proceeds to the next-highest incentive I/O gadget, etc. As described above, for Black to deviate from this order costs more moves than can be gained. White also loses moves playing out of order, but does not gain anything in compensation, as all the potential extra moves are Black's.

The AND gadget for SNORT is shown in Fig. 10. White needs to earn an extra move in order to make up for Black's disjoint free play. White can earn the move

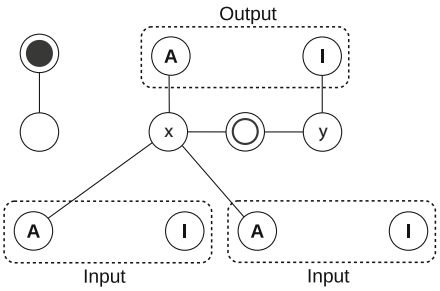


Fig. 10 AND gadget for SNORT

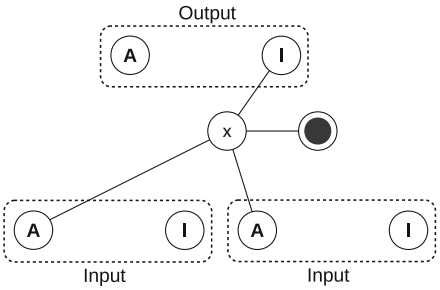


Fig. 11 OR gadget for SNORT

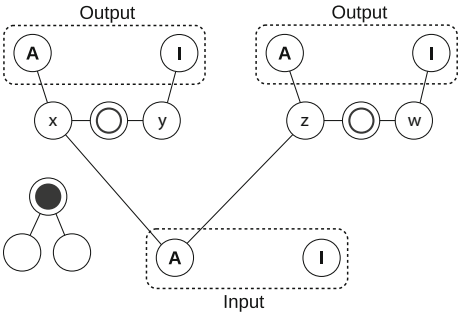


Fig. 12 FANOUT gadget for SNORT

at y by choosing the Inactive output. However, if both of the inputs are activated, then White can activate the output and still gain x for later; a Black play on either input's Active vertex prevents this. Note that if Black got to play first in the output, they could inactivate it, potentially forcing White to later inappropriately activate it, earning Black an extra move. But the incentive mechanism prevents this, by costing Black more than this in incentive difference.

The OR gadget is shown in Fig. 11. White cannot play in the output Active vertex without playing in at least one input Active vertex, or else Black will gain an extra move at x.

The FANOUT gadget, shown in Fig. 12, includes two extra moves for Black that White must make up. White can do so by activating both outputs and later playing on x and z, but only after activating the input; a Black play on the input Active vertex blocks White's desired plays. Otherwise, White must inactivate both outputs, and claim y and w, to keep up.

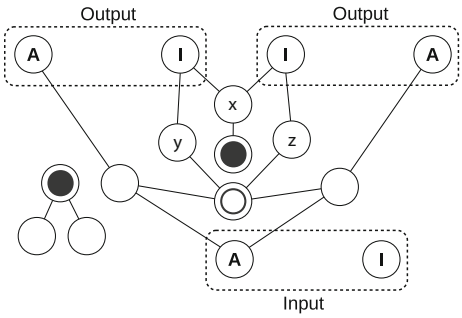


Fig. 13 CHOICE gadget for SNORT

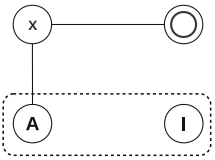


Fig. 14 GOAL gadget for SNORT

The CHOICE gadget is shown in Fig. 13. Again, Black has two extra moves that White must make up within the gadget. If White activates the input, then any White plays in the two outputs will gain the two moves adjacent to those plays. But if White chooses to activate both outputs, Black will also gain a move at x; White cannot afford this, and so must activate at most one output. If Black plays on the input Active vertex, then the only way for White to make the required two extra moves is to inactivate both outputs, later playing y and z.

As with COL, the GOAL gadget for SNORT, shown in Fig. 14, gives White the extra move necessary to win if the corresponding I/O gadget is activated. A Black play at A block's White's play at x.

Theorem 2 *Determining whether the next player in* SNORT *has a winning strategy is PSPACE-complete.*

Proof Given a BOUNDED 2CL instance, we construct a corresponding SNORT instance as described above, identifying input and output I/O gadgets between appropriate vertex gadgets. Each vertex gadget reproduces the behavior of its corresponding constraint-graph vertex, and the incentives ensure that I/O gadget play reproduces legal BOUNDED 2CL play.

Therefore, White can win the BOUNDED 2CL game just when White can win the corresponding SNORT game, by activating a sequence of outputs reaching the GOAL gadget. After the incentives have all been claimed, both players play all the extra moves they have reserved—these will be equal with canonical play—and finally White plays the one extra move enabled in the GOAL gadget. □

6 PLANAR-FJORDS

PLANAR- FJORDS is a partisan placement game where players alternately mark vertices using their color (Black or White), with the restriction that the newly marked vertex

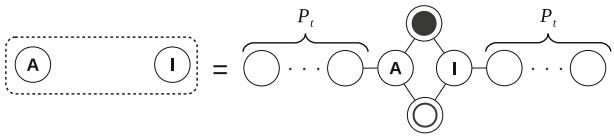


Fig. 15 PLANAR- FJORDS I/O gadget and VARIABLE. P_t is a path graph with t vertices

must be adjacent to a vertex already marked with that player's color. Note that the initial position must contain both colored and uncolored vertices in order to have any move options. Formally:

Definition 3 (PLANAR- FJORDS) PLANAR- FJORDS is a ruleset played on a planar graph, $G = (V, E, c)$, where c is a coloring of vertices $c\colon V \rightarrow \{Black, White, Uncolored\}$. An option for player A is a graph $G' = (V, E, c')$ where $\exists (x, y) \in E$:

- $\forall v \neq x \in V : c'(v) = c(v)$, and
- $c(x) = Uncolored$, and
- $c'(x) = A$'s color, and
- $c(y) = A$'s color.

PLANAR- FJORDS is a generalized version of the board game FJORDS, which is played on a hexagonal grid with some edges and vertices removed. FJORDS was published by Rio Grande games in 2005, but is currently out of print. In the published version of FJORDS, the initial configuration is generated through a randomized process, but once that is complete, the remainder of the game (as described here) is strictly combinatorial. We show that PLANAR- FJORDS is PSPACE-hard; the computational complexity of FJORDS remains an open problem.

Gadgets

PLANAR- FJORDS is in PSPACE, as shown in Sect. 2.3. To show hardness, we reduce from BOUNDED 2CL. We need to specify the details of the PLANAR- FJORDS I/O gadget, and create gadgets corresponding to each of the constraint-logic vertex types in Fig. 1. We will also need a "goal" gadget attached to White's constraint-graph goal edge, to give White the extra move needed to win the PLANAR- FJORDS game just when White can flip that edge in the BOUNDED 2CL game. Each vertex gadget gives White and Black an equal number of moves under canonical play.

Our reduction is similar to the SNORT reduction, in that we provide incentives to ensure that inputs are played before outputs, so canonical BOUNDED 2CL move order is followed. The same argument presented in Sect. 5 shows that the incentives force proper play order here as well. Again, the specific incentive t for each I/O gadget will be determined by a topological sort of the constraint graph, with the incentives spaced sufficiently far apart.

The I/O gadget, shown in Fig. 15, contains incentives: a play on either the Active or the Inactive vertex grants the player an extra t moves later on. It also has extra vertices

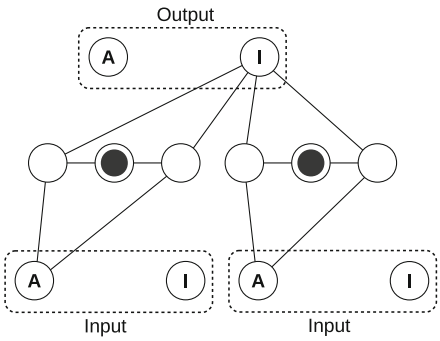


Fig. 16 AND gadget for PLANAR- FJORDS

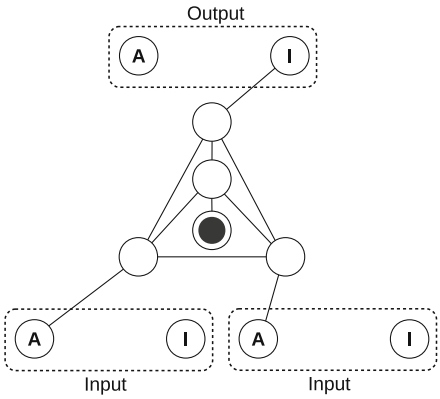


Fig. 17 OR gadget for PLANAR- FJORDS

attached, allowing either player to play on either side. This is necessary, because play can only occur adjacent to existing marked vertices. Again, a VARIABLE gadget is simply an I/O gadget.

The AND gadget is shown in Fig. 16. If White activates the output when one or both inputs are inactive, this will give Black two or four additional moves later, that White cannot afford. If White inactivates the output, or activates both inputs, then Black and White will split these four moves at the end of the game.

The OR gadget is shown in Fig. 17. If White plays on either input Active vertex or the output Inactive vertex, then Black and White will split the four uncolored vertices at the end of the game. But if White activates the output with both inputs inactive, all these moves will go to Black.

The FANOUT gadget is shown in Fig. 18. Essentially this is an upside-down AND gadget, as a constraint-logic FANOUT is also an upside-down AND. The same logic applies here as for the AND: White must either activate the input or inactivate both outputs to keep Black from gaining an extra two or four moves.

The CHOICE gadget is shown in Fig. 19. This is identical to the FANOUT, with the addition of another two-move gain for Black if White activates both outputs. White must play at least two of the input Active vertex and the output Inactive vertices to avoid a loss here.

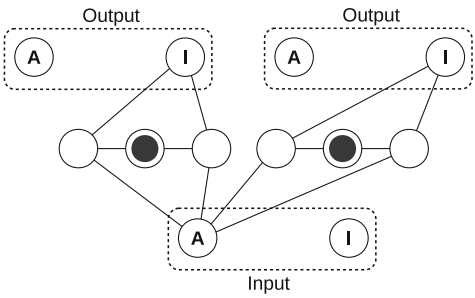


Fig. 18 FANOUT gadget for PLANAR- FJORDS

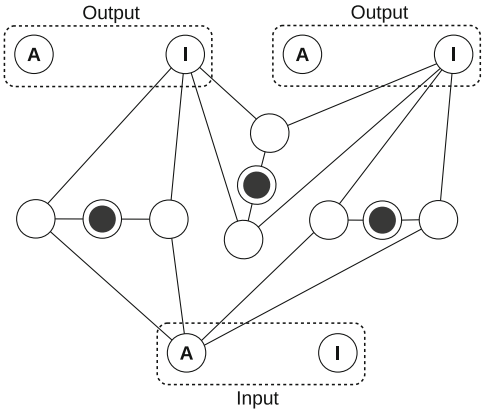


Fig. 19 CHOICE gadget for PLANAR- FJORDS

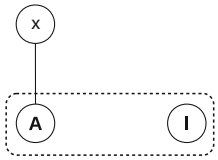


Fig. 20 GOAL gadget for PLANAR- FJORDS

The GOAL gadget is shown in Fig. 20; it is attached to the I/O gadget corresponding to White's goal edge. Again, White gets the extra move at x if the I/O gadget is activated. (In this gadget Black gets the extra move otherwise, though this move is not needed for Black to win.)

Theorem 3 *Determining whether the next player has a winning strategy in* PLANAR-FJORDS *is PSPACE-complete.*

Proof Given a BOUNDED 2CL instance, we construct a corresponding FJORDS instance as described above, identifying input and output I/O gadgets between appropriate vertex gadgets. Each vertex gadget reproduces the behavior of its corresponding constraint-graph vertex, and the incentives ensure that I/O gadget play reproduces legal BOUNDED 2CL play.

Therefore, White can win the BOUNDED 2CL game just when White can win the corresponding FJORDS game, by activating a sequence of outputs reaching the GOAL gadget. After the incentives have all been claimed, both players play all the extra moves

they have reserved—these will be equal with canonical play—and finally White plays the one extra move enabled in the GOAL gadget. □

7 PLANAR-NOGO

PLANAR- NOGO is a partisan placement game where the players alternately mark vertices using their color (Black or White) with the restriction that each connected component of one color must be adjacent to an *Uncolored* vertex. Formally:

Definition 4 (PLANAR- NOGO) PLANAR- NOGO is a ruleset played on any graph, $G = (V, E, c)$, where c is a coloring of vertices $c\colon V \to \{Black, White, Uncolored\}$. A legal position (coloring) has that $\forall$ connected single-color components $C \subseteq V\colon \exists(v, w) \in E$ where:

- $v \in C$, and
- $w \notin C$, and
- $c(w) = Uncolored$.

An option for player A is a graph $G' = (V, E, c')$ where c' is a legal coloring, and $\exists x \in V$:

- $\forall v \neq x \in V : c'(v) = c(v)$, and
- $c(x) = Uncolored$, and
- $c'(x) = A$'s color.

NOGO is the well-known version of PLANAR- NOGO played specifically on a grid graph. Our hardness proof applies only to the more general PLANAR- NOGO; the computational complexity of the grid version remains an open problem.

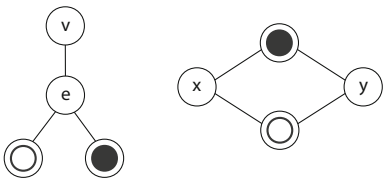
NOGO is itself a non-loopy GO variant where capturing moves are not allowed. In GO, uncolored vertices adjacent to a connected component of one color are known as liberties. GRAPH- NOGO enforces that a liberty must always exist for each connected component. Resolving the computational complexity has been considered an open problem since 2011, when a tournament was played among combinatorial game theorists at the Banff International Research Station. PLANAR- NOGO was discovered to be NP-hard at that meeting; our proof of PSPACE-completeness improves on that.

To prove the hardness of PLANAR- NOGO, we reduce from COL, which was shown to be PSPACE-hard in Theorem 1. The reduction uses only two gadgets.

Theorem 4 *It is PSPACE-complete to determine whether the next player has a winning strategy in* PLANAR- NOGO.

Proof PLANAR- NOGO is in PSPACE, as shown in Sect. 2.3, so we only need to show hardness. Given a COL instance $G = (V, E, c)$, we construct a corresponding PLANAR- NOGO instance $G' = (V', E', c')$.

For every vertex $v \in V$, we have the subgraph shown in Fig. 21a in G'. We color v the same as COL-vertex v is initially colored. Neither player can ever play at e; to do so would remove the liberty from one of the adjacent connected components. As a result, if v is colored, its connected component will always have a liberty.



(a) COL Vertex **(b)** COL Edge

Fig. 21 Gadgets for PLANAR- NOGO. **a** COL Vertex, **b** COL Edge

Also, for each edge $e = (x, y) \in E$, we have the subgraph shown in Fig. 21b in G'. Here x and y are identified with v in the vertex gadgets for COL-vertices x and y, respectively. If x and y are colored the same color, then either the Black or the White vertex in Fig. 21b will have no liberty; this is not allowed. But if x and y are opposite colors, all is well: each is connected to a liberty because of Fig. 21a.

Every uncolored vertex in V corresponds to an uncolored vertex in V', and the only other uncolored vertices in V' are not playable. A move in G exists just when the corresponding move in G' exists. Also, this reduction clearly preserves planarity. Thus, play in the constructed PLANAR- NOGO instance exactly reproduces play in the given COL instance. □

8 Conclusion

We have shown computational hardness results for the four planar placement games COL, SNORT, PLANAR- FJORDS, and PLANAR- NOGO: all are PSPACE-complete. The resolution of COL and SNORT is of particular interest, as these classic problems have been open for decades.

These results highlight the utility of BOUNDED 2CL for showing planar games hard, especially when combined with the incentive mechanism used for SNORT and FJORDS; we expect that there are many further applications awaiting once these techniques become more widely known.

Because the reduction proving hardness for GRAPH- NOGO starts from COL, we also expect that COL will be useful as the source for other reductions. Indeed, we started with a BOUNDED 2CL reduction for GRAPH- NOGO, which also required an incentive mechanism; the COL reduction is much simpler.

9 Future work

There are still many placement games with unknown computational complexity. In particular, it is still unknown whether NOGO (PLANAR- NOGO on grid graphs) and FJORDS (PLANAR- FJORDS on subgraphs of a hexagonal grid) are computationally difficult. Either of these would be a large improvement over the current result. For NOGO in particular it seems very challenging to build gadgets in a grid graph. Is that enough of a restriction to make the game easy? FJORDS may be more approachable, as allowing subgraphs of a hexagonal grid affords much more freedom.

nogo on grid 更难做

fjords on hex 相对好做

讨论初始状态无色或少量节点有颜色的情况

Normally for game complexity one considers the decision question of whether one player has a forced win from a given position. However, because here the game boards (graphs) themselves have structure, we can also ask about the complexity of these games when the graph is initially uncolored, or has only constantly many vertices colored. The reduction for COL on general graphs in Fenner et al. (2015) does show hardness starting from an uncolored graph, so there is hope here.

There is also a version of SNORT played on a grid graph, known as CATS AND DOGS,[5] so it is natural to wonder about its complexity.

Open Problem 1 *Is* NOGO *computationally hard?*

Open Problem 2 *Is* FJORDS *computationally hard?*

Open Problem 3 *What is the hardness of* COL, SNORT, PLANAR- FJORDS, *and* PLANAR- NOGO *if the graph is initially uncolored, or has constantly many colored vertices?*

Open Problem 4 *Is* CATS AND DOGS *computationally hard?*

References

Albert Michael H, Nowakowski Richard J, Wolfe David (2007) Lessons in play: an introduction to combinatorial game theory. A. K. Peters, Wellesley

Berlekamp Elwyn R, Conway John H, Guy Richard K (2001) Winning ways for your mathematical plays, vol 1. A K Peters, Wellesley

Brown JI, Danielle C, Andrew H, Neil M, Rebecca M, Nowakowski RJ, Siegel AA. Polynomial profiles of placement games. To appear in Games of No Chance 5

Alessandro C (2009) Three-player col played on trees is np-complete. In: Proc. of the International Multi-Conference of Engineers and Computer Scientists 2009, 445–447. Newswood Limited

Conway John H (2001) On numbers and games, 2nd edn. A K Peters

Demaine ED, Hearn RA (2008) Constraint logic: a uniform framework for modeling computation as games. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity (Complexity 2008)*, pp 149–162, College Park, Maryland, June 23–26

Demaine Erik D, Hearn RA (2009) Playing games with algorithms: algorithmic combinatorial game theory. In: Albert Michael H, Nowakowski Richard J (eds) Games of No Chance 3, volume 56 of Mathematical Sciences Research Institute Publications. Cambridge University Press, Cambridge, pp 3–56

Fenner SA, Grier D, Messner J, Schaeffer L, Thierauf Thomas (2015) Game values and computational complexity: an analysis via black-white combinatorial games. In: Elbassioni Khaled M, Makino Kazuhisa (eds) ISAAC, volume 9472 of lecture notes in computer science. Springer, Berlin, pp 689–699

Hearn RA (2008) Amazons, Konane, and Cross Purposes are PSPACE-complete. In: Albert Michael H, Nowakowski Richard J (eds) Games of no chance 3. Mathematical Sciences Research Institute Publications, vol 56. Cambridge University Press, Cambridge, pp 287–306

Hearn RA, Demaine ED (2009) Games, puzzles and computation. A K Peters, Wellesley

Schaefer Thomas J (1978) On the complexity of some two-person perfect-information games. J Comput Syst Sci 16(2):185–225

[5] http://www.di.fc.ul.pt/%7Ejpn/cnjm13/index.htm