Tile-Based Knot Assembly with Celtic!

Divya Bajaj, Ryan Knobel, Juan Manuel Perez, Rene Reyes, Ramiro Santos, and Tim Wylie(✉)

University of Texas Rio Grande Valley, Edinburg, TX, USA
timothy.wylie@utrgv.edu

Abstract. In this paper we focus on the intersection of tile assembling systems, edge-matching puzzles, combinatorial games, and knot construction and identity. As a basis, we utilize the game Celtic!, which is a 2-Player board game where the goal of the game is to construct knots where one knot uses more of a player's pieces than the other player over all knots. All pieces must build off an existing knot and a valid knot must be closed. We consider three variations: a 0-player self-assembly variation that deterministically places pieces to form a closed knot of some length, a 1-player puzzle variation where the goal is to form a closed knot of some length, and the original 2-player game with restricted pieces. We show these are P-complete, NP-complete (depending on the pieces), and PSPACE-complete (for a first-player win), respectively. We nearly fully characterize the hardness of the 1-player puzzle based on the pieces.

1 Introduction

Square edge-matching puzzles generally employ the use of (possibly rotatable) tiles, with each side given a specific label that dictates how they may be used in conjunction with other tiles. These types of puzzles have existed for centuries, and are often deceptively difficult to solve. For instance, the Eternity II puzzle [19], introduced in 2007, offered $2,000,000 for the first complete solution. While no restrictions were placed on the methods used, no complete solution has yet to be found, with the closest solutions falling short a few pieces. The work of [6] shows that edge-matching puzzles, including MacMahon Squares [12], Scramble Squares, and Tetra Vex [17], are NP-complete and are equivalent to other types of puzzles such as jigsaw and polyomino packing puzzles.

The puzzles often remain NP-hard even for small instances. In [7], the authors show that even for a 1-by-n puzzle, unsigned edge-matching with rotation is still NP-hard and fixed parameter tractable in the number of unique labels. This work was later improved in [3], where it was shown that for both signed and unsigned edges, 1-by-n puzzles are NP-hard, even to approximate. Naturally, variations of these types of puzzles have been considered, such as having additional inequality constraints between adjacent tiles, triangular edge matchings, and when no target shape is specified [2].

This research was supported in part by National Science Foundation Grant CCF-2329918.

H. Fernau and B. Zhu (Eds.): IWOCA 2025, LNCS 15885, pp. 316–331, 2025.
https://doi.org/10.1007/978-3-031-98740-3_23

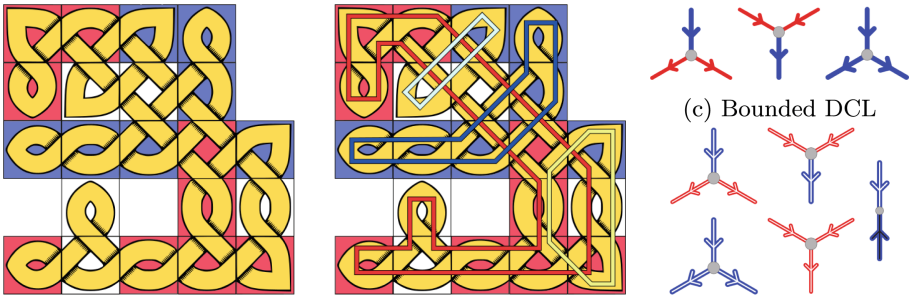


Fig. 1. (a) Final Celtic! board where all knots are closed and no valid moves exist. (b) Celtic! board displaying completed knots. The knot score for Red (R) and Blue (B) players based on the knot. Red knot: R(9) - B(2), Blue knot: R(1) - B(5), Yellow knot: R(4) - B(1), and White knot: R(0) - B(1). Given these knots, the best score for Red is 9 and 5 points for Blue. (c) Gadgets for 0-player non-planar Bounded Deterministic CL. (d) Gadgets for Bounded 2-player CL. (Color figure online)

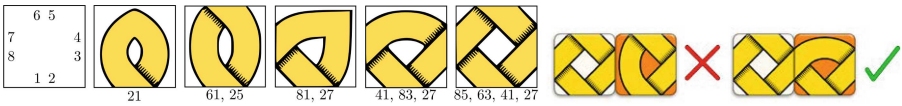


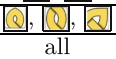
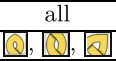
Fig. 2. (a) The 5 types of pieces in the game as well as their path connections (in this orientation). Each player (red and blue) has two copies of each type and there is a shared set of white-backed ones. (b) Invalid vs. valid placement. (Color figure online)

In general, edge-matching puzzles are undecidable with unlimited pieces [1]. More abstractly, generalized edge-matching of squares (without rotation) is closely related to computational complexity as exhibited through Wang tiles [18], and more recently, self-assembly models such as the abstract Tile Assembly Model (aTAM) [20], which are both Turing Universal even with small tile sets.

Our work focuses on a game with similar edge-matching mechanics: Celtic! [5]. Celtic! is a 2-Player (Blue/Red) tile placement game where each player attempts to complete knots by alternatively placing pieces onto a fixed size board. Each piece can be thought of as a rotatable square with a 'closed' or 'open' label on each side, similar to that of unsigned edge-matching with two labels. The goal of the game is alternate playing pieces and completing knots. The player who has the most number of pieces of their color in a knot (over all knots) wins the game. See Fig. 1 for an example of a completed game with scoring.

The pieces of Celtic! use crossing paths, which makes it a type of connection game where the goal is to build connecting paths. Path based pieces and mechanics show up in several games such as Tsuro [14], Squiggle [13], Travel [8],

Table 1. Summary of results for different Celtic! variations. All reductions use a $O(n) \times O(n)$ size board.

Version	Player Pieces	Board Pieces	Complexity	Theorem
0-Player		all	P-complete	Thm. 8
1-Player		all	$O(n^2)$	Thm. 1
		all	$O(n^2)$	Thm. 2
		all	$O(n^2)$	Thm. 3
			NP-complete	Thm. 4
			NP-complete	Thm. 5
		all	OPEN	-
		all	OPEN	-
			NP-complete	Thm. 6
2-Player			PSPACE-complete	Thm. 7

Kaliko/Psyche-Paths/Cram [16], and many others. Please see [4] for an overview of connection-based (and edge-matching) games.

Our Contributions. Table 1 overviews the main results. We characterize the game of Celtic! for 0, 1, and 2 player games based on the types of pieces. We consider the complexity of three variations of the board game. To start, we look at a 1-player variation of Celtic! in which some pieces initially exist on the board and the goal of the player is to form a single closed knot of some length in k moves. We show that for some pieces, this is NP-complete in Sect. 3. We show that it is PSPACE-complete to determine if there is a 1^{st}-player win in a 2-player game (Sect. 4). In Sect. 5, we analyze a 0-player version with deterministic placement and show that making a closed knot of some length is P-complete.

2 Preliminaries

We first detail some game mechanics from the original game, as well as provide formal definitions for the general game and problems afterwards.

2.1 Game Mechanics

Pieces. The game pieces consist of 5 distinct types of pieces, with 5 copies of each type (25 pieces in total), with path segments (Fig. 2a). The Blue and Red players each own 10 pieces (2 of each type with their background color), and they share 5 pieces with a **white** background.

Initial Configuration and Rules. The  piece is placed to begin. Players take turns placing a piece adjacent to already existing pieces to form knots; each piece must be placed within the playing area such that no knot is prevented from closing (Fig. 2b). In the standard game, it must remain in a 5×5 square.

2.2 Definitions

Let $D(p_1 p_2)$ be the ℓ_1-norm (Manhattan distance) between points $p_1 p_2$. We say 2 cells $[x_1 y_1]$,$[x_2 y_2]$ are **adjacent** if $D((x_1, y_1), (x_2, y_2)) = 1$.

Definition 1 (Board and cells). *A **board** B is a fixed $w \times h$ lattice surface composed of **cells**, with each cell $[x,y] \in B$ corresponding to a location in row x and column y, and [0,0] corresponding to the bottom-leftmost cell.*

Definition 2 (Path-segment). *A path-segment is a pair of connected edge points on a piece that represent a continuous path from one point to the other.*

Definition 3 (Piece). *A **piece** $p \in \{$      $\}$ is a rotatable 4-sided unit square, where each side is labeled **open** or **close** denoting the type of side that may or may not attach on this side. Each open side has at most two path-segment points (over and under). Label two points on each side $\langle 1,2,3,4,5,6,7,8 \rangle$ going counterclockwise around the square (Fig. 2a) as all path-segment points, Celtic! uses pieces (ignoring rotation) with path-segments (21)  , (25, 61)  , (27, 81)  , (27, 41, 83)  , (27, 41, 63, 85)  . We use the convention that each path-segment has the over number followed by the under number, and we use this rotation as the canonical orientation.*

Definition 4 (Matching and Paths). *Two pieces $p_1 p_2$ are **matching** if they are in adjacent cells and the touching sides are both labeled open. A **path** is made of a sequence of path-segments from pieces. A path between $p_1 p_2$ conceptually connects the internal path-segment $i_1 o_1$ from p_1 to path-segment $i_2 o_2$ on p_2, and is defined as $P = \langle i_1 o_1 i_2 o_2, \rangle$, where $o_1 i_2$ lie on the touching sides of p_1, p_2, and $o_1 \mod 2 = i_2 \mod 2$.*

Definition 5 (Valid piece). *A piece in cell $[x_1 y_1]$ is said to be **valid** if:*

1. *$\exists [x_i y_j]$ for $(i,j) \in \{(2,1),(1,2),(0,1),(1,0)\}$ such that $[x_1 y_1]$,$[x_i, y_j]$ contain matching pieces, and*
2. *All other adjacent cells $[x_i y_j]$ for $(i,j) \in \{(2,1),(1,2),(0,1),(1,0)\}$ satisfy (a) $[x_1 y_1]$,$[x_i y_j]$ contain matching pieces, (b) the touching sides of the pieces in $[x_1 y_1]$ and $[x_i y_j]$ are both labeled closed, or (c) $[x_i, y_j]$ is empty.*

Definition 6 (Knot). *A **knot** is a continuous path $P = \langle i_1 o_1 i_k o_k \rangle$ through path-segments on valid pieces $\langle p_1 p_k \rangle$ where each path piece p_a identifies a corresponding input and output path-segment $i_a o_a$ on a piece such that*

1. *each p_a matches with p_{a+1} for $1 \leq a \leq k-1$ and p_k matches with p_1, and*
2. *each o_a continues through i_{a+1} for $1 \leq a \leq k-1$ and o_k continues to i_1.*

*Note that multiple path-segments through the same piece may exist. The length (score) of a knot is not the number of path-segments in pieces it crosses, but the number of **distinct** pieces (or grid locations) the total path crosses. This definition ignores the topological properties of a knot in relation to other knots.*

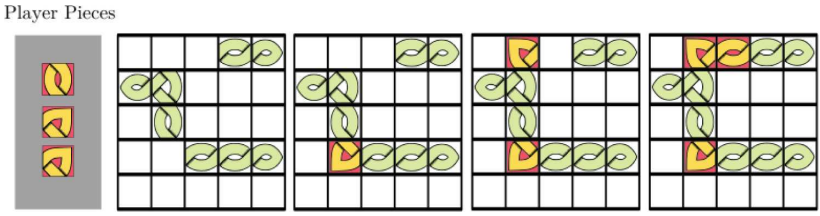


Fig. 3. Example of an initial game board configuration for generalized 1-Player Celtic! and a sequence of valid moves that form a closed knot of length $|K| = 11$ in $k = 3$ moves given 2 pieces and 1 piece.

Problem Definitions. This paper looks at 3 variations of Celtic!: a 0-Player simulation (Fig. 13), a 1-Player puzzle (Fig. 3), and a 2-Player game (Fig. 1).

Definition 7 (0-Player Celtic!). *A 0-player Celtic! simulation is defined as:*
INPUT: *A $w \times h$ board partially filled with pieces* $\in \{$, , , , $\}$, *a* *piece as a starter piece and a positive integer L.*
OUTPUT: *Is it possible to place the starter piece in a cell $[0i]$ rotated* 180 *degrees for some integer* $0 \leq i \leq w$ *and create a knot of length* $\geq L$*?*

Definition 8 (1-Player Celtic!). *A 1-Player Celtic! puzzle is defined as:*
INPUT: *A $w \times h$ board partially filled with pieces, a multi-set S of pieces (with $k = |S|$) and positive integer L where* $k \leq L \leq k + L \leq wh$.
OUTPUT: *Can k valid pieces (from S) be placed such that a closed knot of length* $\geq L$ *is formed?*

Definition 9 (2-Player Celtic!). *A 2-Player Celtic! game is defined as:*
INPUT: *A $w \times h$ board with a white* *piece in cell $[i,j]$, where* $1 \leq i \leq w$ *and* $1 \leq j \leq h$, *multi-sets S_R, S_B, and S_W of red, blue, and white pieces, respectively.*
MOVES: *Players take turns placing a valid move using one of their colored pieces or a white piece. Once no valid moves exist, each player chooses a completed knot and receives a score. The score M_R, M_B for each player is a positive integer denoting the number of pieces of their color from their chosen knot.*
OUTPUT: *Whether there is a Blue player win, a Red player win, or a draw based on* $\max(M_R, M_B)$.

2.3 Constraint Logic

The 0-player and 2-player reductions make use of constraint logic (CL) [9]. Problems in CL are based on a constraint graph where each weighted directed edge has value 2 (blue) or 1 (red), and every vertex has a minimum inflow constraint of 2. The game is based on flipping edges of the graph, where an edge can be flipped only if the minimum inflow constraints are maintained. In [9], they show that all graphs in CL can be reduced to a few simple gadgets, each of degree 3, that are equivalent for the games.

The 0-player gadgets for Bounded Deterministic Constraint Logic (BDCL), Fig. 1c, require that the graph is non-planar and every edge may only be flipped once. They consist of an AND, FANOUT, and OR. BDCL is P-complete.

The Bounded 2-player Constraint Logic (B2CL) gadgets (Fig. 1d) have assigned edges (white is 1^{st} player and black is 2^{nd}) and may only be flipped once by the owner. They consist of an AND, FANOUT, OR, CHOICE, and VARIABLE. B2CL is PSPACE-complete even for planar graphs. Note that the 2^{nd} player can only move in the variable gadget.

3 Generalized 1-Player Celtic!

This section looks at generalized 1-Player Celtic! played on an $O(n) \times O(n)$ board, where the goal is to build a closed knot of length $\geq L$ in k moves. We start with polynomial time algorithms when restricted to placing  ,  , or  pieces. We then show a general framework for the NP-hardness reductions, which is used in the final section to show NP-completeness when restricted to either placing  or  , as well as for other piece combinations.

3.1 Polynomial Cases

Theorem 1. *Building a closed knot of length $\geq L$ with k moves by placing*  *pieces in Generalized 1-Player Celtic!, where the initial board configuration can contain*  ,  ,  ,  , *and*  *pieces, is solvable in $O(n^2)$ time.*

Theorem 2. *Building a closed knot of length $\geq L$ with k moves by placing*  *pieces in Generalized 1-Player Celtic!, where the initial board configuration can contain*  ,  ,  ,  , *and*  *pieces, is solvable in $O(n^2)$ time.*

Theorem 3. *Building a closed knot of length $\geq L$ with k moves by placing*  *pieces in Generalized 1-Player Celtic!, where the initial board configuration can contain*  ,  ,  ,  , *and*  *pieces, is solvable in $O(n^2)$ time.*

3.2 General Framework for Hardness Reductions

We outline the framework used for the hardness reductions in Sect. 3.3. For each result, we reduce from directed, planar Hamiltonian cycle with max degree 3 [15]. At a high-level, given a directed, planar graph, we transform it into an equivalent rectilinearly embedded graph G_e [10]. We space out the vertices in G_e so each edge may be transformed to have roughly the same length (Fig. 4). Denote this new graph G. We then construct the Celtic! board used for the reduction. We replace edges with  and  pieces, leaving a 7×7 empty region centered at each vertex, which is where the vertex gadget will be placed. The gadget consists of a  piece with other pieces placed depending on the allowable pieces, and the main objective being that all incoming edges lie either to the left or right of the central vertex piece, and all outgoing edges lie on the other side. This forces the

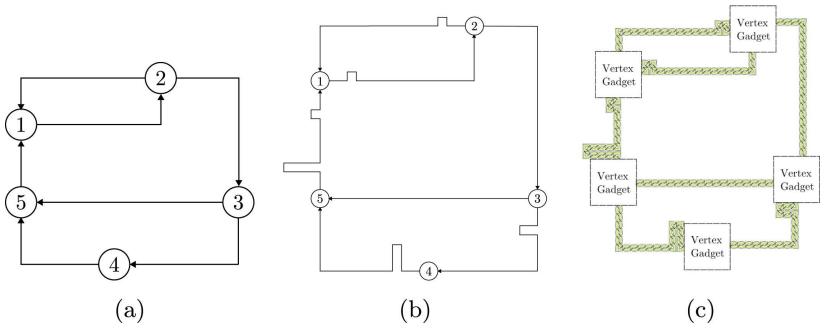


Fig. 4. The 3-step process for creating the initial board for the NP-complete reductions. (a) Example directed, planar graph with max degree 3. (b) The graph is scaled by a factor of $|V|^2$, then transformed to have edges of roughly equal lengths using the free space around each vertex. The edges within 3 units of the vertex are left untouched. (c) The initial board created for the graph. The vertex gadget changes per reduction depending on the pieces used.

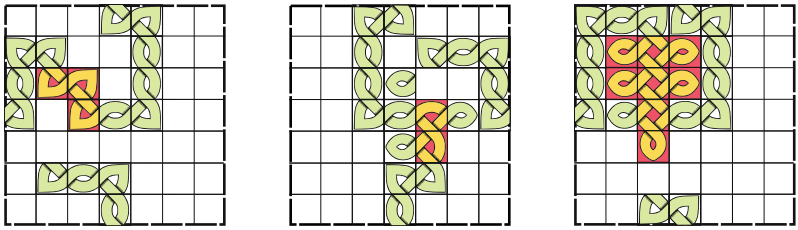


Fig. 5. Vertex gadgets for the (a) piece selecting the western edge, (b) selecting the southern edge, and (c) and selecting the western edge.

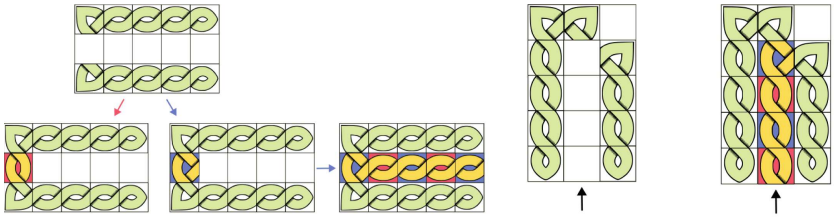
player to place pieces that choose exactly one incoming edge and one outgoing edge per vertex gadget. If these choices are consistent across all vertices, then a knot of length L is formed corresponding to a Hamiltonian cycle in the original graph G_e, otherwise, the knot is not formed.

For each reduction, we give a vertex gadget with the different combinations of input and output on different sides for the construction using the pieces for that problem. Depending on the reduction, some additional pieces are needed to aid with making a choice between the incoming and outgoing edges at each vertex piece. Although not all cases are given, a few examples are shown in Fig. 5.

3.3 Hardness Results

Using the framework discussed in Sect. 3.2, we now show NP-completeness for different piece combinations.

Lemma 1. *Generalized 1-Player Celtic! is in NP.*



(a) Variable Gadget (b) Win Gadget (c) Win Example

Fig. 6. (a) VARIABLE Gadget in the 2-player game representing the VARIABLE Ver-tex in a Bounded 2CL. If Red player moves first, the knot is closed with , preventing Blue player from continuing an edge to other gadgets. If Blue player moves first, is placed, which restricts the red player to only continue the knot to other gadgets. (b) In the WINNING Gadget, only the Blue player is allowed to finally close the knot. Once the blue player closes the knot, the knot can no longer be extended. (c) Example moves in the WINNING gadget. The red player can only continue in the gadget, while blue player makes the last move. (Color figure online)

Theorem 4. *Building a closed knot of length* $\geq L$ *with* k *moves by placing pieces in Generalized 1-Player Celtic!, where the initial board configuration can contain and pieces, is NP-complete.*

Theorem 5. *Building a closed knot of length* $\geq L$ *with* k *moves by placing pieces in Generalized 1-Player Celtic!, where the initial board configuration can contain , , and pieces, is NP-complete.*

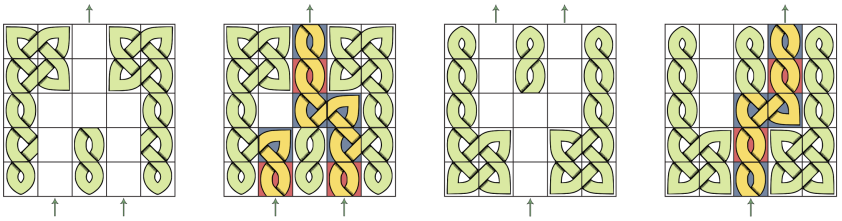
Theorem 6. *Building a closed knot of length* $\geq L$ *with* k *moves by placing and pieces in Generalized 1-Player Celtic!, where the initial board configuration can contain , , and pieces, is NP-complete.*

4 Constraint-Graph Reduction for 2-Player

We now analyze generalized 2-player Celtic! played on an $O(n) \times O(n)$ board where both Blue and Red players have a multi-set of pieces.

Lemma 2. *Given an* $O(n) \times O(n)$ *board configuration where both Blue and Red player have a multi-set of pieces containing pieces* [, , ,]. *Deciding whether there is a sequence of moves to force a Blue player (player 1) win starting at a given board position is PSPACE-hard.*

Proof. To show that determining if Blue player has a forced win is PSPACE-hard, we reduce Bounded 2-player Constraint Logic, where a sequence of moves in the 2-Player Celtic! game represents the flipping of an edge in the Constraint Graph whose configuration represents the constraint-satisfaction problem of 2CL. The problem is defined as: Does the Blue player have a forced win?



(a) OR Gadget (b) OR Example (c) CHOICE (d) CHOICE Ex.

Fig. 7. (a) OR Gadget in the 2-player game representing the OR vertex in a Bounded 2CL. For the player to continue with a valid knot, the player can connect a knot from the right side using the or from the left side using the . The player can then complete the knot from the opposite side as well. (b) An example of a sequence of moves in the OR gadget when the knot from the right side is continued. (c) CHOICE Gadget in the 2-player game representing the CHOICE vertex in a Bounded 2CL. For the player to continue a valid knot, the player can use either or . Depending on the piece, the knot continues right or left. No knot can be completed from the opposite side. (d) Moves in the CHOICE gadget continuing the knot to the right side.

We reduce Bounded 2CL by designing the VARIABLE, AND, OR, CHOICE and FANOUT gadgets for our game that are joined to create a constraint graph representing the sequence of moves. There is a Winning gadget that the Blue player can use to finish the knot. We show that the constraint graph accepts the configuration when the Blue player wins.

VARIABLE Gadget. (Figure 6a) We need to show that this gadget satisfies the same constraints as that of a variable vertex in the Bounded 2CL constraint graph. If the Red player moves first, it closes the knot using the piece preventing Blue player from continuing an edge to other gadgets. But if the Blue player moves first, it can place a piece that restricts the red player to only continue the knot to other gadgets.

WIRE Gadget. A wire is a walled path of width 1 that connect gadgets.

WINNING Gadget. (Figure 6b) The winning gadget is used by the blue player to close the knot once they reach it. In all other gadgets, the players continue with the knot, and they finally lead into the winning gadget. Here only the blue player is able to close the knot using the as shown in Fig. 6c.

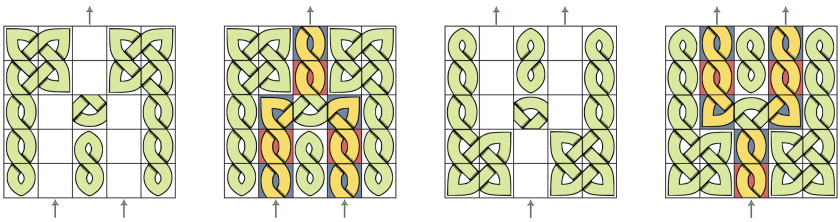
OR Gadget. (Figures 7a, 7b) The construction of this OR gadget in our game satisfies the same constraints as a Bounded 2CL OR vertex. For the player to continue with a valid knot, the player can connect a knot from the right side using the or from the left side using the . Another knot can later be used to close the opposite Input side using .

CHOICE Gadget. (Figure 7c) The construction of this gadget satisfies the same constraints as that of a CHOICE vertex in a Bounded 2CL whereas equence of moves mark whether the knot can be extended towards left or right. For the player to continue with a valid knot, the player can use either or . Depending on the piece used, the knot will then continue right or left. No knot can be completed from the opposite side.

AND Gadget. (Figure 8a) We now show that the construction of this gadget satisfies the same constraints as that of the AND vertex in a Bounded 2CL constraint. For the player to continue with a valid knot, the player can use both and . If the player only connects a knot from one side, the knot will not be closed. Hence, the pieces in the knot will not count towards the players.

FANOUT Gadget. (Figure 8c) This gadget satisfies the same constraints as that of the FANOUT vertex in a Bounded 2CL constraint graph. The construction is quite similar to the AND gadget where the Output edge is now the Input edge. If the player is able to connect to the Input edge, then the player can complete the knot by connecting on both sides using and .

Since the Red player only has , they can use it in the Variable gadget to block, or continue with existing knots in all the gadgets. The Blue player makes moves in the Variable gadgets. If the Blue player is able to assign the Variables to true (place in the Variable gadget), they can then satisfy other constraints by continuing knots in other gadgets. Once the knot reaches the Winning gadget, the Blue player can close it, thus ending and winning the game. If both players make alternate moves and Blue player makes the first move, the winning gadget



(a) AND Gadget (b) AND Example (c) FANOUT (d) FANOUT Ex.

Fig. 8. (a) AND Gadget in the 2-player game representing the AND vertex in a Bounded 2CL. For the player to continue with a valid knot, the player can use both and . If the player only connects a knot from one side, the knot will not be closed. Hence, the pieces in the knot will not count towards the players. (b) An example of moves in the AND gadget. To complete the knot the player has to connect from both sides. (c) FANOUT Gadget in the 2-player game representing the FANOUT vertex in a Bounded 2CL. The player can complete the knot by connecting on both sides using and . (d) An example of moves in the FANOUT gadget. The player is able to continue knots on both sides.

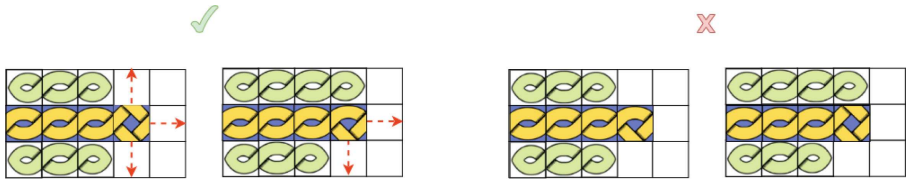


Fig. 9. Valid(left)andinvalid(right)moves.Intheinvalidmoves,theleft hasanempty cellonthenorthside,sotheplayermustplacea piece.Intheotherexample,there isanadjacent piece to the north with a closed side, thus the played piece needs a closed side on its north.

will ensure that the Blue player is the last player to make a move. Thus, Blue has at least one piece more than Red, making them the winner.

The construction of these gadgets satisfies all constraints as that of vertices in a constraint graph of a Bounded 2CL. We know that finding the variables assignments for the constraint logic problem for a Bounded 2CL is PSPACE-complete. Such variable assignments correspond to the Blue and Red player making moves in the variable gadgets, setting them to true or false. Therefore deciding if the Blue player can make certain moves in the variable gadgets such that the Blue player always wins is PSPACE-hard. □

Lemma 3. *Given an $O(n) \times O(n)$ board configuration where both Blue and Red player have a multi-set of pieces containing pieces* [ ]. *Deciding whether there is a sequence of moves the Blue player can make that results in a forced win starting at a given board position is in PSPACE (Fig. 9).*

Theorem 7. *Given an $O(n) \times O(n)$ board configuration where both Blue and Red player have a multi-set of pieces containing pieces* [ ]. *Deciding if the given board configuration is a 1^{st} player win is PSPACE-complete.*

5 Constraint Logic Reduction for 0-Player

This section considers a 0-player variation of Celtic! with rule constraints to allow for automation, and shows that 0-player Celtic! is P-complete.

5.1 0-Player Decisions

We define placement rules for an automated 0-player to make decisions.

- Structural Pieces - Existing white pieces used as walls or requiring specific pieces to traverse.
- Signal Pieces - The automated blue pieces that propagate through the board.
- Close South Rule - A piece is always placed to close a southward path unless there is another adjacent cell that requires a different piece (Fig. 10a).
- 3-Open Piece Rule - A piece is placed when 3 sides are possible (Fig. 10b).
- 4-Open Piece Rule - A is placed when 4 open sides are possible, which also allows for crossovers in the graph (Fig. 10c) (Fig. 11).

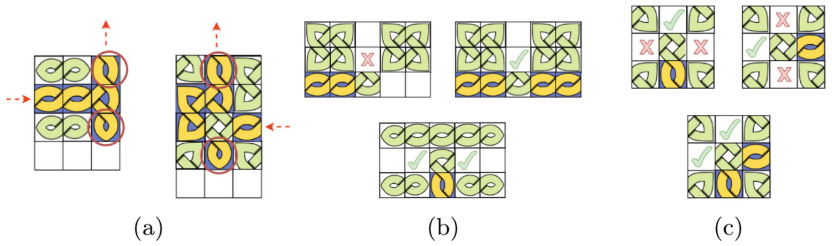


Fig. 10. (a) An example of the Close South Movements rule. The red arrows show the direction that the player is placing the pieces. As it continues north, any movement south will result in a piece to close that path. (b) The conditions needed in order to continue through a piece. (c) The conditions necessary to continue through a piece. (Color figure online)

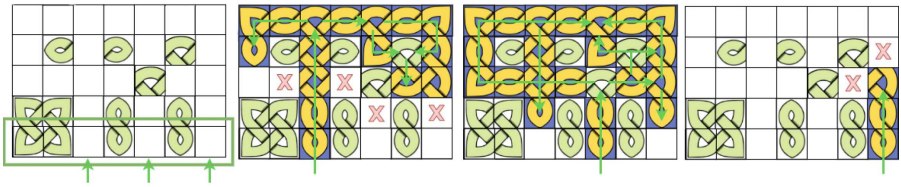


Fig. 11. Example 0-player simulations of a board, depending on where the starter piece is placed. Here, the longest knot is created by placing it in the middle. If the goal is to get a knot ≥ 5, the player can place the starter piece in the left or middle starting location. The green arrows show the path the simulation takes and the red X's are the directions it can not continue through. (Color figure online)

5.2 0-Player Bounded Deterministic Constraint Logic Gadgets

Bounded DCL is P-complete [9] since we can use it to simulate monotone boolean circuits, which only requires us to simulate simple AND, OR and FANOUT gadgets. We reduce from Bounded DCL to 0-Player Celtic! simulation. We show the construction of the AND, OR and FANOUT in Fig. 12.

WIRE Gadget. A wire is a walled (by structural pieces) path of width 1 that connects gadgets.

OR Gadget. The OR gadget has two inputs and one output. If a signal piece propagates through the left input, by the **4-Open Sided Rule**, it will propagate vertically through the 4 open-sided structural piece (Fig. 12). As it propagates north, it splits into 2 signals: one of them continues as the output of the gadget, the other takes a turn left. By the **Close South Movement Rule** and **4-Open Sided Rule** it will then connect with the 4 open-sided piece and continue horizontally until it closes going southward through the other input path. Similarly, if the signal piece propagates on the right input, the signal traverses the

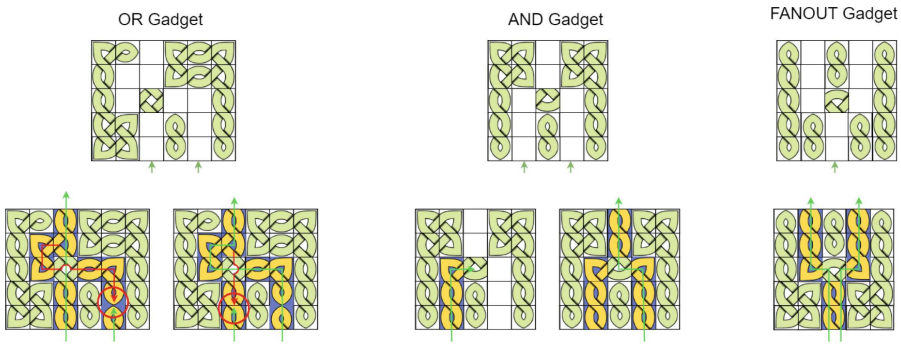


Fig. 12. Examples of an OR, AND, and FANOUT gadget, and how they are traversed. The green arrows show inputs propagating inward, red arrows depict southward propagation getting closed. In the OR gadget, one input causes the other input path to close, and any other signal coming in will close. The AND requires both inputs in order to continue outwards. FANOUT splits into 2. (Color figure online)

structural piece horizontally, continuing north with its output, and eventually propagating south vertically through the piece and closes the other input. If another signal tries to propagate through the opposite side it will immediately close, but the signal has already propagated from the first input.

AND Gadget. By the **3-Open Sided Rule**, a signal piece coming from either input cannot continue through the output unless the automated player has connected signal pieces to both parallel sides of the structural piece (Fig. 12).

FANOUT Gadget. By the **3-Open Sided Rule**, a signal piece propagating through the perpendicular open side of the piece allows the automated player to split the signal and propagate through both outputs (Fig. 12).

Simulating DCL with Gadgets. Figure 13 demonstrates how the gadgets on a board can simulate bounded DCL. The goal is to attempt to reach the output from the top AND gadget. Notice it can only reach the output by playing in the middle starting position where the signal splits in the FANOUT gadget, through the 2 OR gadgets, and then into the final AND gadget. The goal of a 0-Player Celtic! simulation is to generate a knot of size $\geq L$ so you can extend the output of the last top gadget to be longer than any possible knot before that, and let L equal to that. Thus, the only knot that can achieve this is the knot that can output through the last gadget.

Theorem 8. *0-Player Celtic! is P-complete.*

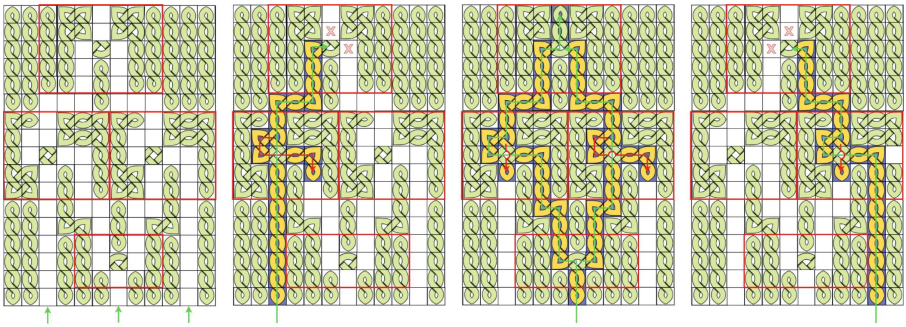


Fig. 13. Example of the 0-Player Celtic! reduction from DCL. The red squares highlight the gadgets locations. On top we have an AND with inputs coming from the outputs of the middle OR gadgets. At the bottom, we can place our starter piece as input into one of the ORs, or we can input it into the FANOUT. Here, only the middle starting position will reach the opposite side of the board. (Color figure online)

6 Generalized Celtic! with Boards of Odd Dimension

This section considers generalized Celtic! played on boards of odd dimension, where the count of each , , , and white pieces are even and the set of pieces each player gets is equal. This includes when the white pieces are excluded altogether, or if the number of white pieces are assumed infinite. We show that through a *strategy-stealing* argument, where player 2 takes advantage of the game's symmetry to prevent player 1 from having a winning strategy, the game is guaranteed to be a draw. Let S_B, S_R and S_W be multi-sets of blue, red and white pieces, respectively.

Theorem 9. *Celtic! played on an $n \times n$ board, where n is odd, is a guaranteed draw when $S_B = S_R$ and the count of each , , , , $\in S_W$ is even.*

Proof. Although no formal proof is provided due to space, the basic idea is strategy stealing based on board symmetry. If the board size is odd, and there are an even number of each neutral white piece, the second player can always mirror the play of the first player on the other side of the board. This would ensure that all knots end up with an equal number of each color or that there is an equivalent mirrored knot with the player colors inverted.

7 Conclusion

This work characterizes and analyzes the complexity of 3 variations of Celtic!: The 0, 1, and 2 player variants depending on the pieces allowed. In the 1-Player version, we show NP-completeness or membership in P depending on the pieces allowed. For the 2-player variation, deciding if there is a first-player win is PSPACE-complete. We show that a 0-Player variation with forced play,

determining if a closed knot of some length can be formed after an initial starter piece is P-complete. This work naturally leads to some open questions, such as those in the table and below.

- What is the complexity with different pieces for Red or Blue player?
- What is the complexity of the different games given a board of fixed dimension? What restricted board instances are polynomial time solvable?
- What is the complexity of the two open cases in Table 1?
- We show that for any general board of odd dimension with even numbers of each piece, the game always sends in a draw. Without these restrictions, is the standard game a first-player win?
- What if we look at pattern complexity based on the knots? General patterns could be answered similar to the PATS (patterned self-assembly tile set synthesis) problem [11], but if we consider the topology of the knots, the pattern is no longer based solely on the tile-type, or even color, at some location.

References

1. Berger, R.L.: The undecidability of the domino problem. Mem. Am. Math. Soc. (1966)
2. Bosboom, J., et al.: Edge matching with inequalities, triangles, unknown shape, and two players (2020). arXiv:2002.03887
3. Bosboom, J., Demaine, E.D., Demaine, M.L., Hesterberg, A., Manurangsi, P., Yodpinyanee, A.: Even 1 $\times n$ edge-matching and jigsaw puzzles are really hard (2016). arXiv:1701.00146
4. Browne, C.: Connection Games: Variations on a Theme. AK Peters, Ltd. (2005)
5. Browne, C.: Celtic! [Board Game] (2009)
6. Demaine, E.D., Demaine, M.L.: Jigsaw puzzles, edge matching, and polyomino packing: connections and complexity. Graphs Comb. **23**, 195–208 (2007)
7. Ebbesen, M., Fischer, P., Witt, C.: Edge-matching problems with rotations (2017). arXiv:1703.09421
8. Hardesty, F.W., Schenck, I.W.: Competitive road building and travel game. US grant 3309092A (1967)
9. Hearn, R.A., Demaine, E.D.: Games, Puzzles, and Computation. CRC Press, Boca Raton (2009)
10. Liu, Y., Morgana, A., Simeone, B.: A linear algorithm for 2-bend embeddings of planar graphs in the two-dimensional grid. Disc. Appl. Math. **81**(1–3), 69–91 (1998)
11. Ma, X., Lombardi, F.: Synthesis of tile sets for dna self-assembly. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **27**(5), 963–967 (2008)
12. MacMahon, P.A.: New Mathematical Pastimes. Cambridge University Press, Cambridge (1921)
13. McMurchie, T.: Squiggle Game. US grant 4180271A (1979)
14. McMurchie, T.: Tsuro. Calliope Games (2005)
15. Plesńik, J.: The np-completeness of the hamiltonian cycle problem in planar diagraphs with degree bound two. Inf. Process. Lett. **8**(4), 199–201 (1979)
16. Schensted, C., Titus, C.: Kalico (Psyche-Paths). Funtastic, Kadon (1968)
17. Takenaga, Y., Walsh, T.: Tetravex is np-complete. Inf. Process. Lett. **99**(5), 171–174 (2006)

18. Wang, H.: Proving theorems by pattern recognition–ii. Bell Syst. Techn. J. **40**(1), 1–41 (1961)
19. Wikipedia: Eternity ii, en.wikipedia.org/wiki/Eternity_II_puzzle (2024)
20. Winfree, E.: Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology (1998)