

Tiling rectangles with holey polyominoes

Dmitry Kamenetsky and Tristrom Cooke
 dkamenetsky@gmail.com, tcooke@internode.on.net
 Adelaide, Australia

October 29, 2015

Abstract

We present a new type of polyominoes that can have transparent squares (holes). We show how these polyominoes can tile rectangles and we categorise them according to their tiling ability. We were able to categorise all but 6 polyominoes with 5 or fewer visible squares.

1 Introduction

Polyominoes are geometric shapes made from squares. We introduce *holey* polyominoes as polyominoes that contain transparent squares, making them disjoint. We now give formal definitions.

Definition 1.1. A *polyomino* is a geometric shape formed by the union of non-overlapping squares edge to edge such that at least two corners of the squares touch.

Definition 1.2. A *holey polyomino* of order (n, k) is a polyomino with n visible squares and k transparent squares. k must be the least number of transparent squares required to connect all the visible squares.

From now on we refer to holey polyominoes of order (n, k) as (n, k) -polyominoes. For example, Figure 1 shows a $(4, 1)$ -polyomino. Visible squares are shown in blue, while the transparent squares are in white. Although we can make all visible squares connected via squares a and c, it is sufficient to just use square b, making $k = 1$. Figure 2 shows all four possible $(3, 1)$ -polyominoes. Note that regular polyominoes are a subset of holey polyominoes; a regular polyomino of order n is a $(n, 0)$ -polyomino.

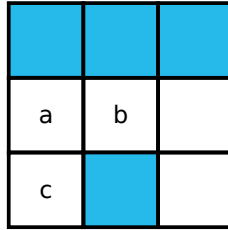


Figure 1: A $(4, 1)$ -polyomino. We can make all visible squares (in blue) connected by making square b visible.

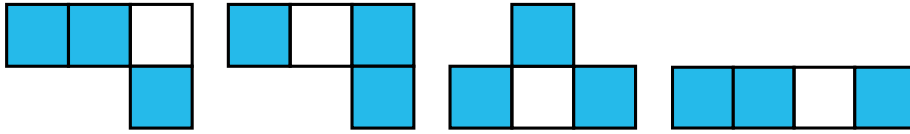


Figure 2: All $(3,1)$ -polyominoes.

We assume that polyominoes are free, meaning that they can be flipped, mirrored and rotated at will. Figure 3 shows all 8 congruent versions of a single $(2,2)$ -polyomino.

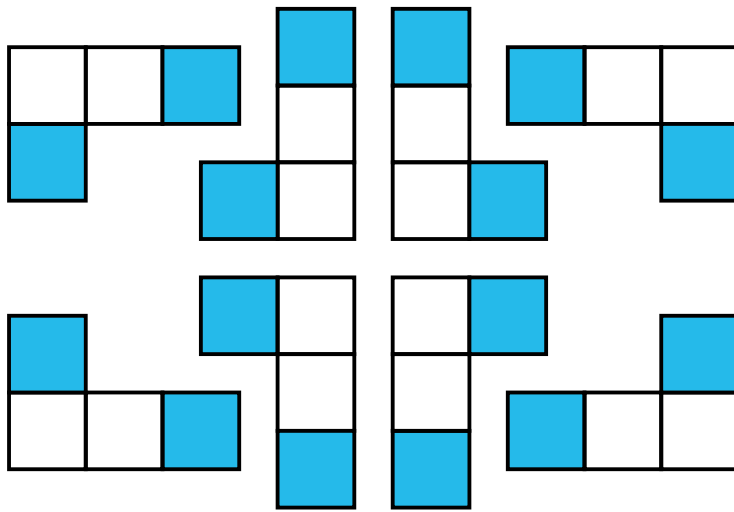


Figure 3: All 8 congruent versions of a $(2,2)$ -polyomino.

2 Related work

The concept of polyominoes can be traced back to ancient times. The famous British puzzle expert Henry Dudeney and the Fairy Chess Review created problems involving n -ominoes, which they represented as figures cut from checkerboards [9]. The term *polyominoes* was coined by Solomon Golomb in 1953 and later popularised by Martin Gardner [6]. Polyominoes gained a great deal of popularity through tiling puzzles and games such as Tetris and Blokus.

Numerous variations to polyominoes have been suggested over the years, such as: polyiamonds (from equilateral triangles), polyhexes (from regular hexagons), polycubes (from cubes). All of these however, do not involve any transparent squares. Perhaps the closest variation is polyplets: polyomino-like objects made by attaching squares joined either at sides or corners (see Figure 4 top row). Note that polyplets are a subset of holey polyominoes. Another related idea is rounded polyominoes [8] (see Figure 4 bottom row). These are polyplets with rounded corners and bridges connecting diagonally adjacent squares. Unlike polyplets and holey polyominoes, rounded polyominoes can be made in the physical world.

Polyomino tiling problems ask whether copies of a single polyomino can tile (cover) a given region, such as a plane or a rectangle. In 1960's Golomb [5] and Klarner [9, 10] were the first to study these problems for particular polyominoes. For the problem of rectangular tiling, results have been found for various polyominoes [11, 3]. Mark Reid provides extensive literature on this subject on his site [12].

Tiling with disjoint polyominoes has been studied for certain classes of polyominoes. Chvatal et al., [1, problem 8] showed that any single-row $(3, k)$ -polyomino can tile a $1 \times n$ rectangle. Gordon [7] showed that any n -dimensional $(3, k)$ -polyomino can tile the \mathbb{Z}^n lattice. In their problem 9, Chvatal et al., [1] ask whether every $(4, k)$ -polyomino tiles the plane? In 1985 this question was answered with affirmative by Coppersmith [2]. Friedman [4] has collected results about tiling rectangles with single-row (n, k) -polyominoes for $n + k \leq 9$ and $k \geq 1$.

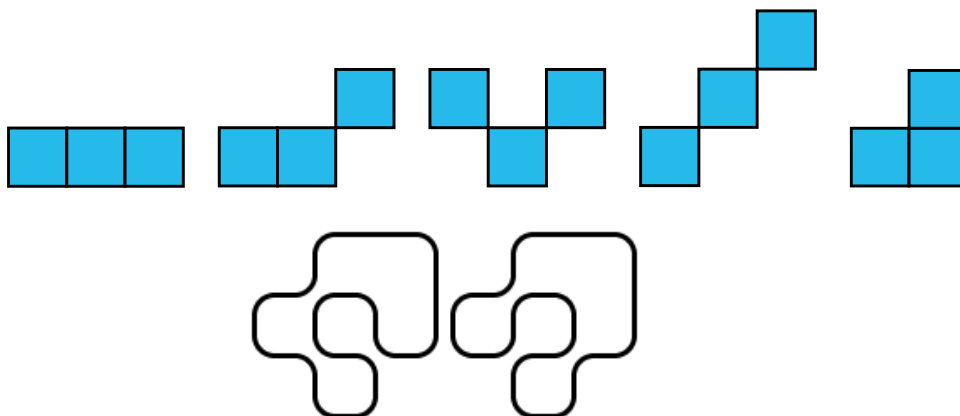


Figure 4: Top row: all polyplets of order 3. Bottom row: two rounded pentominoes with the same squares, but different connections (bridges). Image courtesy of http://www.ericharshbarger.org/pentominoes/article_09.html

3 Rectangular tilings

We now consider the problem of tiling rectangles with holey polyominoes. During tiling, a visible square may lie on top of a transparent square (or vice-versa), but it cannot lie on top of another visible square. Naturally, transparent squares can lie on top of other transparent squares. If possible, we are interested in finding the smallest rectangle (in area) that can be tiled by a single (n, k) -polyomino.

We have investigated rectangular tilings of the following classes of polyominoes: $(2, 1)$, $(2, 2)$, $(3, 1)$, $(3, 2)$, $(4, 1)$ and $(5, 1)$. We call a holey polyomino *rectifiable* (or *solved*) if it can tile a rectangle. We call it *unrectifiable* (or *impossible*) if we can show that it cannot tile a rectangle. Otherwise, we call it *unknown*. Table 1 summarises our results.

3.1 Algorithm

We use the classical Depth First Search (DFS) algorithm to find the status of each polyomino (see Algorithm 1). The root node of the search tree

| n | k | Solved | Impossible | Unknown | Total |
|---|---|--------|------------|---------|-------|
| 2 | 1 | 2 | 0 | 0 | 2 |
| 2 | 2 | 2 | 0 | 0 | 2 |
| 3 | 1 | 4 | 0 | 0 | 4 |
| 3 | 2 | 9 | 2 | 0 | 11 |
| 4 | 1 | 14 | 6 | 0 | 20 |
| 5 | 1 | 28 | 34 | 6 | 68 |

Table 1: Summary of results.

is the empty $N \times N$ grid. A move consists of placing a single polyomino in the current grid. The search terminates when one of the following conditions is met:

1. We completely fill a rectangular area with tiles. If this happens then we have found a solution and we can terminate (see lines 4-8). The tiled rectangle may be smaller than $N \times N$.
2. We have visited every possible node in the tree without finding a solution. This means that there is no solution with a rectangle whose maximum side is N or less. This however, does not exclude solutions whose maximum side is greater than N .

A move consists of placing a tile in an empty grid location. We try all possible orientations and shifts of the tile, as given by the $Rot(\cdot)$ function on line 19. For any given node there can be a great number of possible moves that can be made. Trying all moves exhaustively may be too slow or even infeasible. Hence we would like to make moves that bring us closer to search termination.

For this reason we order our moves from most to least constrained. An empty location is considered to be more constrained if there are less unique ways of filling it with tiles. Line 10 of the algorithm computes how constrained each empty location is. Line 11 computes the most constrained location (r^*, c^*) .

If a polyomino is rectifiable then it must be able to tile the corner of a rectangle. Since corners are the most constrained locations in the grid, we begin our search there. As soon as we find a location that cannot be filled by a tile, we can backtrack (see lines 13-16).

Solve is a recursive function that takes the following parameters:

- *grid*: a working array of placed tiles. Initially all squares are empty.
- *tile*: the tile that we are trying to place.
- *moves*: a list of moves that we have already made.
- *R*: the index of the first row that is completely empty.
- *C*: the index of the first column that is completely empty.

The algorithm begins by initialising an empty $N \times N$ *grid* and an empty list of *moves*. We then call $Solve(grid, tile, moves, 0, 0)$. If a solution exists, then this algorithm will find one, although it may not be the smallest.

Algorithm 1 : Algorithm for finding solutions.

function Solve(grid, tile, moves, R, C)

```
1  //set of empty locations within bounds
2   $E := \{(r, c) : \text{grid}(r, c) \text{ is empty, } r < R, c < C\}$ 
3
4  //no more empty locations, so we found a  $R \times C$  solution
5  if  $E = \emptyset$ 
6    grid.print()
7    terminate
8  end
9
10  $\forall (r, c) \in E : G(r, c) := \# \text{ ways to fill grid}(r, c)$ 
11  $(r^*, c^*) := \text{argmin}_{(r, c)} G(r, c)$  //most constrained location
12
13 //no solution, so backtrack
14 if  $G(r^*, c^*) = 0$ 
15   return
16 end
17
18 //for each rotated and shifted version of the tile
19  $\forall t \in \text{Rot}(\text{tile})$ 
20   move = new Move( $t, r^*, c^*$ )
21   if grid.isValidMove(move)
22     grid.makeMove(move)
23     moves.add(move)
24
25    $R' := \max(R, r^* + t.\text{height})$  //compute new bounds
26    $C' := \max(C, c^* + t.\text{width})$ 
27
28   Solve(grid, tile, moves,  $R', C'$ ) //recurse
29
30   grid.undoMove(move)
31   moves.remove(move)
32 end
33 end
```

3.2 Rectifiable polyominoes

For certain classes of polyominoes (such as the $(2, k)$ class) we were able to show that all polyominoes within that class are rectifiable.

Theorem 1. *Every $(2, k)$ -polyomino is rectifiable.*

Proof. Consider a large finite grid with the top-left corner being at row=0 and column=0. We can describe any $(2, k)$ -polyomino based on the location of its two visible squares in the grid. So (r_1, c_1, r_2, c_2) represents a polyomino with its first visible square located at (r_1, c_1) and second square at (r_2, c_2) . We now show how any $(2, k)$ -polyomino can tile a rectangle. Without loss of generality, we place the first polyomino at $(0, 0, r, c)$ as shown in Figure 5(a). We then place $(c-1)$ polyominoes into locations $(0, 1, r, c+1), \dots, (0, c-1, r, 2c-1)$ as shown in Figure 5(b). We now place c tiles into locations $(0, c, r, 0), \dots, (0, 2c-1, r, c-1)$ as shown in Figure 5(c). We now have 2 rows of squares of length $2c$ at rows 0 and r . Now replicate this structure and shift it one row down. After $(r-1)$ repetitions of this process we will get a filled rectangle with $2r$ rows and $2c$ columns. \square

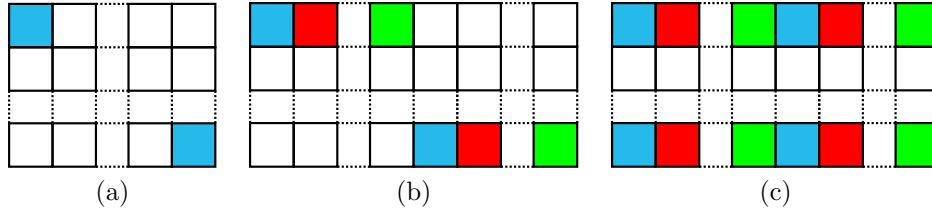


Figure 5: Steps involved in showing that every $(2, k)$ -polyomino is rectifiable.

Tables 2-8 show the smallest known tiling sizes. Some of the smaller solutions are simple enough to find by hand and are optimal - tilings with the smallest area. Tables 9-10 show the actual tilings for some non-trivial pieces. These solutions are not guaranteed to be optimal.

| Piece | Smallest Tiling | Piece | Smallest Tiling |
|-------|-----------------|-------|-----------------|
| | 1x4 | | 2x2 |

Table 2: Smallest known solutions for $n=2$ and $k=1$.

| Piece | Smallest Tiling | Piece | Smallest Tiling |
|-------|-----------------|-------|-----------------|
| | 1x6 | | 2x4 |

Table 3: Smallest known solutions for $n=2$ and $k=2$.


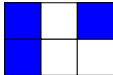
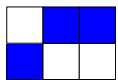
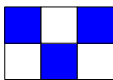
| Piece | Smallest Tiling | Piece | Smallest Tiling |
|---|-----------------|---|-----------------|
|  | 1x6 |  | 3x4 |
|  | 2x3 |  | 2x3 |

Table 4: Smallest known solutions for $n=3$ and $k=1$.

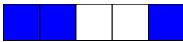

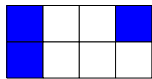
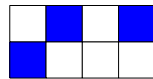
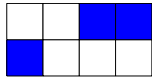
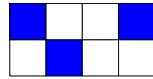
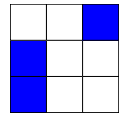
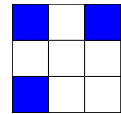
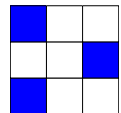
| Piece | Smallest Tiling | Piece | Smallest Tiling |
|---|-----------------|---|-----------------|
|  | 1x6 |  | 1x6 |
|  | 2x9 |  | 2x6 |
|  | 2x6 |  | 2x6 |
|  | 3x4 |  | 4x6 |
|  | 3x4 | | |

Table 5: Smallest known solutions for $n=3$ and $k=2$.


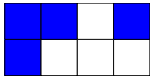
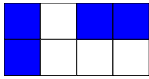
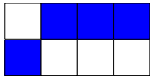
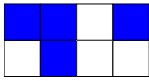
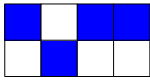
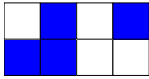
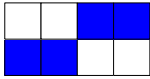
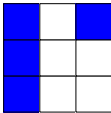
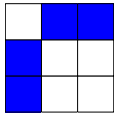
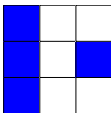
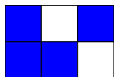
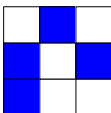
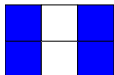
| Piece | Smallest Tiling | Piece | Smallest Tiling |
|---|-----------------|---|-----------------|
|  | 1x8 |  | 4x4 |
|  | 4x4 |  | 2x4 |
|  | 2x4 |  | 2x4 |
|  | 2x4 |  | 2x4 |
|  | 4x4 |  | 14x36 |
|  | 4x4 |  | 2x4 |
|  | 8x12 |  | 2x4 |

Table 6: Smallest known solutions for $n=4$ and $k=1$.



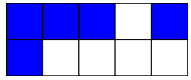
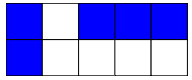
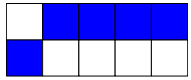
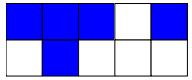
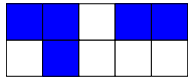
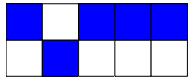
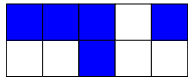
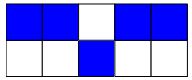
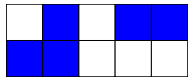
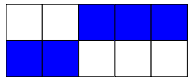
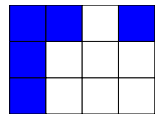
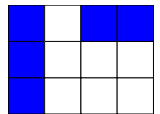
| Piece | Smallest Tiling | Piece | Smallest Tiling |
|---|-----------------|--|-----------------|
|  | 1x10 |  | 6x20 |
|  | 10x10 |  | 10x10 |
|  | 2x5 |  | 2x5 |
|  | 12x15 |  | 2x5 |
|  | 15x18 |  | 2x5 |
|  | 20x22 |  | 2x5 |
|  | 4x5 |  | 18x20 |

Table 7: Smallest known solutions for $n=5$ and $k=1$.

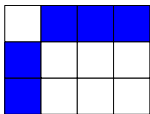
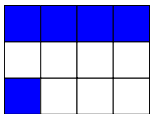
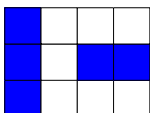
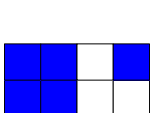
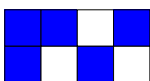
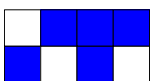
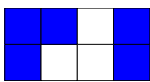
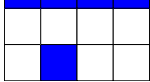
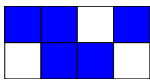
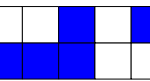
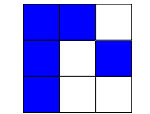
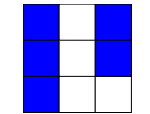
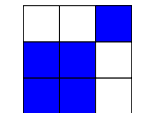
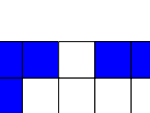
| Piece | Smallest Tiling | Piece | Smallest Tiling |
|---|-----------------|---|-----------------|
|  | 20x20 |  | 4x5 |
|  | 4x5 |  | 6x10 |
|  | 2x5 |  | 4x10 |
|  | 2x5 |  | 5x6 |
|  | 4x10 |  | 10x12 |
|  | 6x10 |  | 4x5 |
|  | 20x24 |  | 11x40 |

Table 8: Smallest known solutions for $n=5$ and $k=1$.


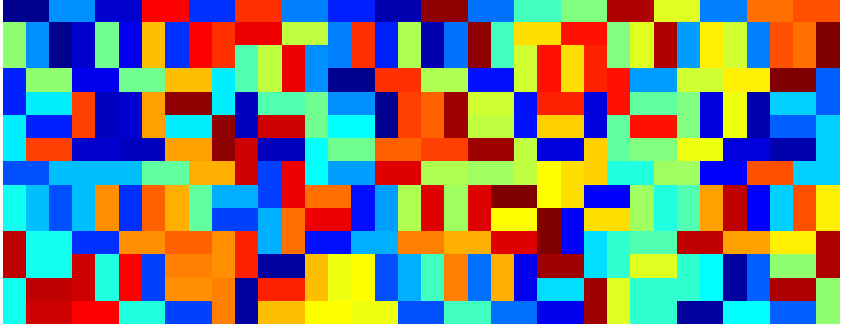
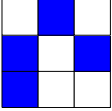
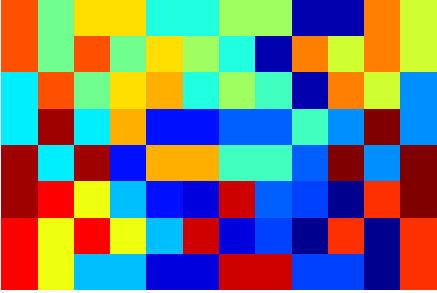

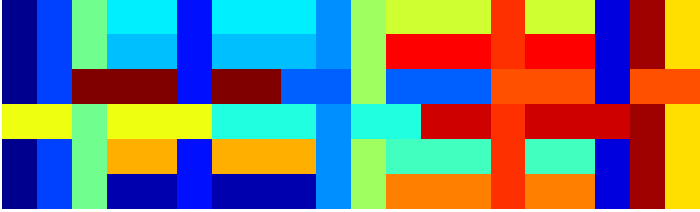
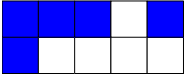
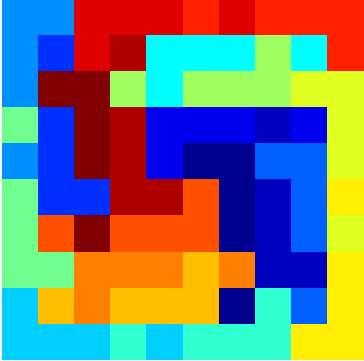
| Piece | Solution | Size |
|---|--|-------|
|  |  | 14x36 |
|  |  | 8x12 |
|  |  | 6x20 |
|  |  | 10x10 |

Table 9: Non-trivial solutions 1.

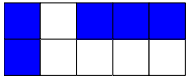
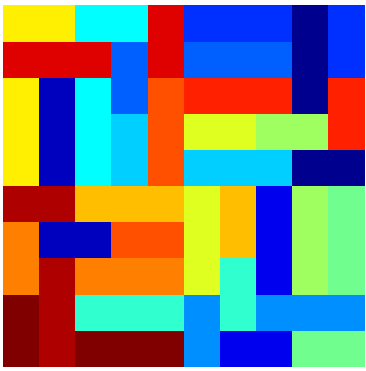
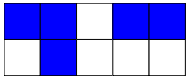
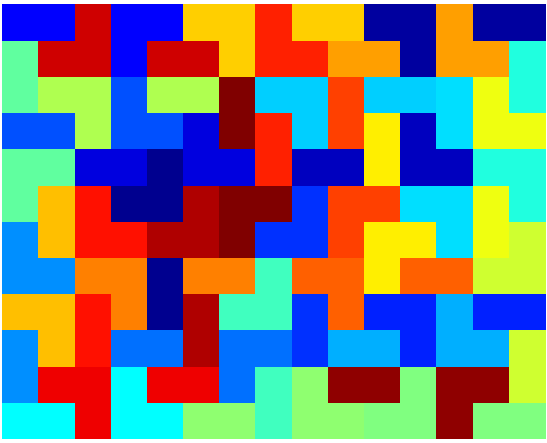

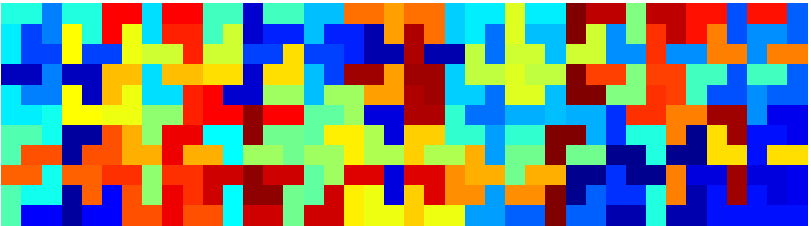
| Piece | Solution | Size |
|---|--|-------|
|  |  | 10x10 |
|  |  | 12x15 |
|  |  | 11x40 |

Table 10: Non-trivial solutions 2.

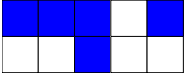
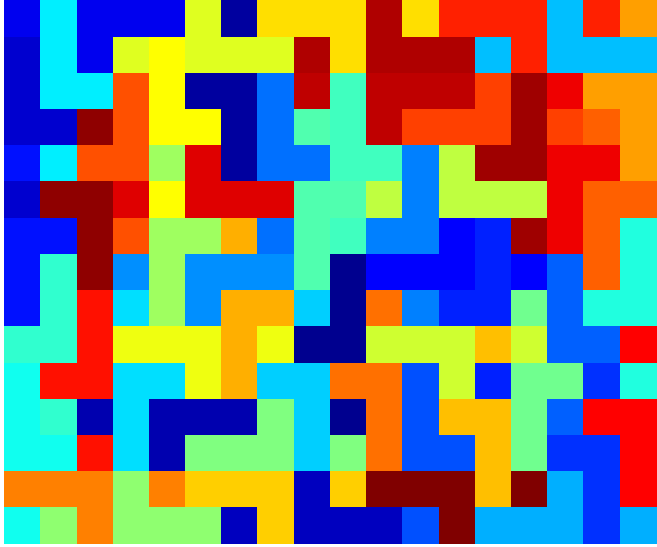
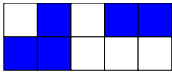
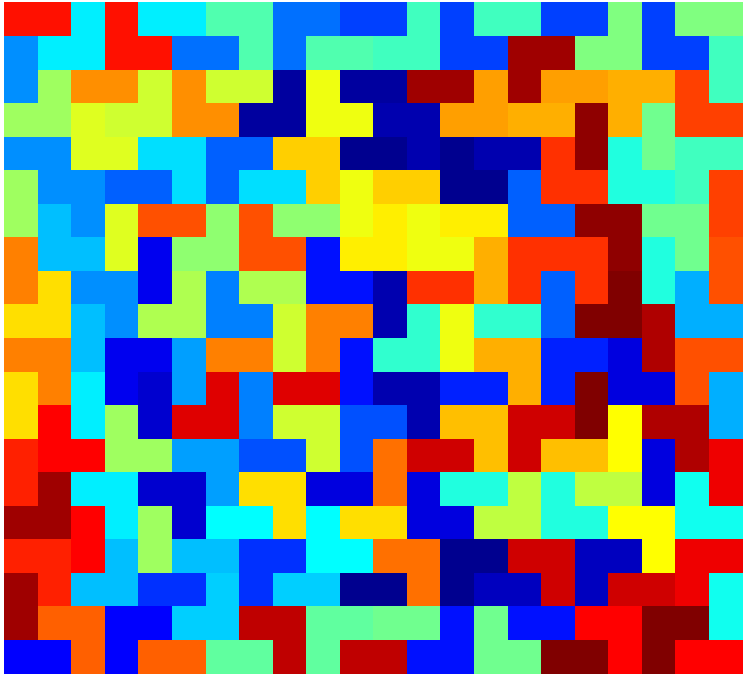
| Piece | Solution | Size |
|---|--|-------|
|  |  | 15x18 |
|  |  | 20x22 |

Table 11: Non-trivial solutions 3.

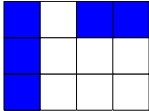
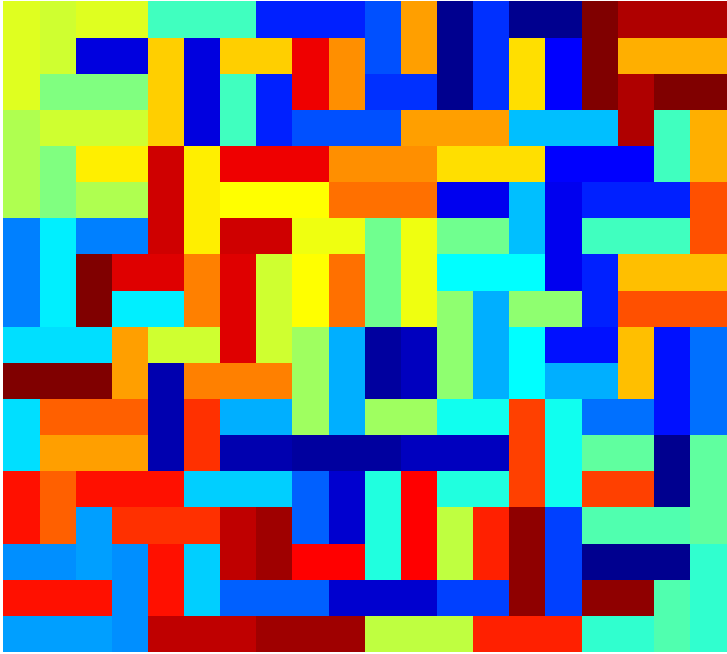
| Piece | Solution | Size |
|--|---|-------|
|  |  | 18x20 |

Table 12: Non-trivial solutions 4.

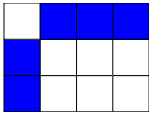
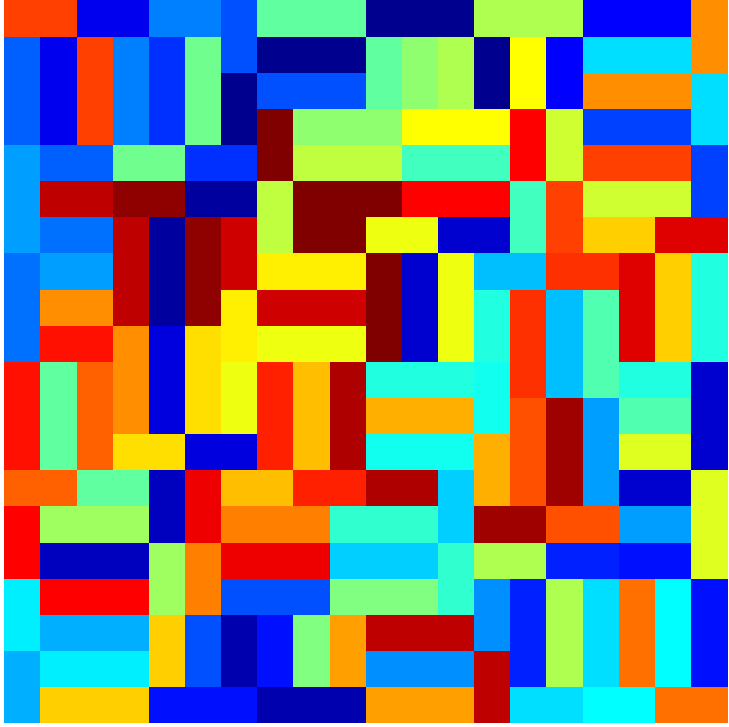

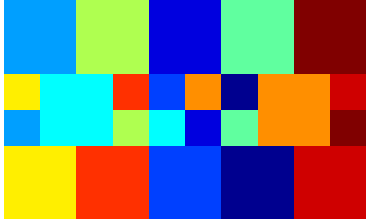
| Piece | Solution | Size |
|---|--|-------|
|  |  | 20x20 |
|  |  | 6x10 |

Table 13: Non-trivial solutions 5.

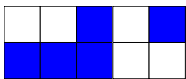
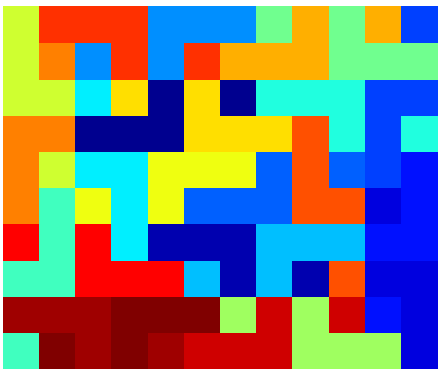
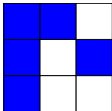
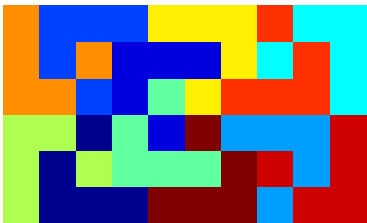
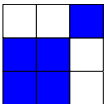
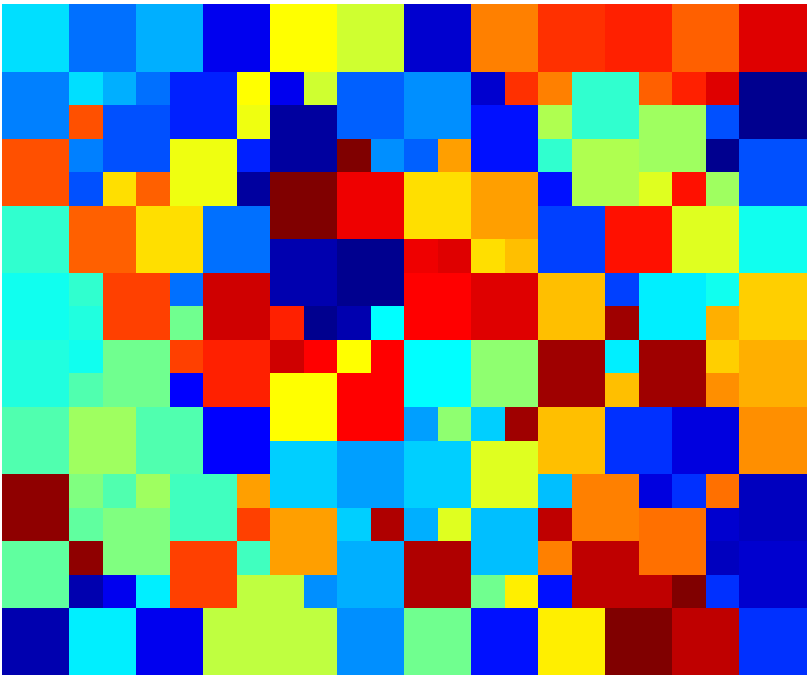
| Piece | Solution | Size |
|---|--|-------|
|  |  | 10x12 |
|  |  | 6x10 |
|  |  | 20x24 |

Table 14: Non-trivial solutions 6.

3.3 Impossible polyominoes

In some trivial cases it is possible to determine if a polyomino is unrectifiable. First we define a tile's *bounding box*:

Definition 3.1. A tile's *bounding box* is the smallest rectangle that surrounds the tile's visible cells.

Theorem 2. *A tile is not rectifiable if every corner cell of its bounding box is transparent.*

Proof. If every corner cell of a polyomino's bounding box is transparent then it cannot tile a corner. Hence such a polyomino cannot tile a rectangle. \square

For example, this theorem can be applied to the right-most tile in Table 16. Note that there are rectifiable tiles with 3 (out of 4) transparent corners in their bounding box, for example see second row in Table 9.

Definition 3.2. An empty (transparent) square cannot be *filled* with a tile if there is no way of legally placing that tile such that the square becomes visible.

Theorem 3. *A tile is not rectifiable if it has a transparent square inside its bounding box that cannot be filled with that tile.*

Proof. For a tile to be rectifiable, it must be possible to fill every transparent square inside its bounding box. Hence if there is a transparent square that cannot be filled then the tile is not rectifiable. \square

The above theorem can be applied to the tile shown in Figure 6, where it is impossible to fill in the centre square.

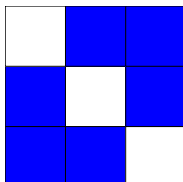


Figure 6: This polyomino is unrectifiable, because it is impossible to fill its centre square.

Apart from the simple cases covered by Theorems 2-3, we do not have an efficient method for verifying that a piece is impossible. In fact there is no single pattern of visible and transparent squares whose existence in the polyomino would signal that the polyomino is impossible. This is shown in Theorem 4.

Definition 3.3. A polyomino P *contains* polyomino T , if T can be placed on top of P (superpositioned) such that its visible squares lie on top of P 's visible squares and its transparent squares lie on top of P 's transparent squares. Note that a polyomino always contains itself.

Theorem 4. *For every polyomino T there exists a rectifiable polyomino P that contains T .*

Proof. If T is already rectifiable then set P to T . Otherwise we need to show that any unrectifiable polyomino T can be made rectifiable by adding extra squares to it. Consider an arbitrary unrectifiable polyomino T (such as the one in Figure 7(a)). Create the inverse polyomino T' by making all visible squares in T transparent and vice versa. Create a polyomino P by rotating T' 180 degrees and placing it next to T (see Figure 7(b)). Clearly P contains T . P is rectifiable, because it forms a rectangle when combined with another copy (see Figure 7(c)). \square

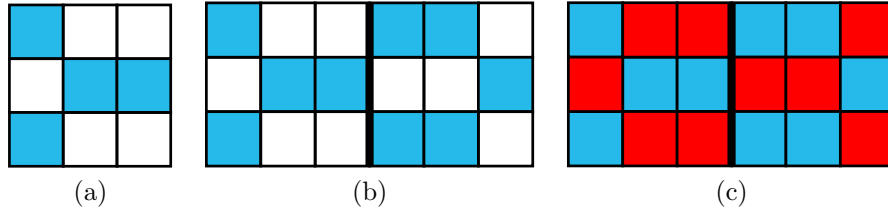


Figure 7: Proof that every unrectifiable polyomino can be made rectifiable. (a) Unrectifiable polyomino T . (b) Rectifiable polyomino P composed of T and its inverse T' . (c) Rectangle tiles by P .

Theorem 4 shows how any unrectifiable polyomino can be converted to a rectifiable one by doubling the number of its squares. However, for some polyominoes it is possible to make the conversion without such a drastic change. In the following example we make the conversion by making a single transparent square visible. Consider the polyomino in Figure 8(left). If we make square a visible, the resulting polyomino tiles a rectangle (see Figure 8(right)).

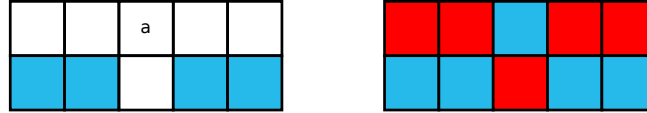


Figure 8: An impossible polyomino (left) is made rectifiable by making square a visible. The resulting polyomino tiles a rectangle (right).

We designed an algorithm which can prove that a polyomino is unrectifiable (see Algorithm 2). The idea is to start with a sufficiently large empty grid¹ and place new tiles in an empty cell with the smallest value as shown in Figure 9. Let $Z(\cdot)$ be a function that takes a grid and returns such a location (see line 1). This ordering aims to fill the top-left corner, which is a pre-requisite to filling a rectangle. If we cannot fill this corner after trying all possible tile placements then the polyomino is not rectifiable.

If Algorithm 2 terminates at line 6 then the piece is rectifiable since we have used it to tile a rectangle. If it terminates at line 11 then we do not have a proof that the piece is unrectifiable. We can try to enlarge the original grid, but this may give us the same result. In this case we may never be able to prove that the piece is unrectifiable using this

¹In our experiments we found that a 100×100 grid was sufficient.

²Here the smallest index is given by the order defined by $Z(\cdot)$.

| | | | | | |
|--|-----|-----|-----|-----|-----|
| | 1 | 2 | 4 | 7 | ... |
| | 3 | 5 | 8 | ... | |
| | 6 | 9 | ... | | |
| | 10 | ... | | | |
| | ... | | | | |

Figure 9:

Algorithm 2 : Algorithm for proving whether a polyomino is unrectifiable.

function Prove(grid, tile)

```

1  (r, c) := Z(grid)    //first empty using order from Figure 9
2
3  //we found a solution
4  if (r, c) = ∅
5    grid.print()
6    terminate
7  end
8
9  //grid is too small, need to enlarge it
10 if r = grid.height
11   terminate
12 end
13
14 //for each rotated version of the tile
15 ∀t ∈ Rot(tile)
16   //move that places t's visible cell with the smallest index2 at (r, c)
17   move = new Move(t, r, c)
18   if grid.isValidMove(move)
19     grid.makeMove(move)
20     Prove(grid, tile)    //recurse
21     grid.undoMove(move)
22   end
23 end
```

method. Otherwise, if the algorithm terminates normally then it has proven that the tile is unrectifiable. We used Algorithm 2 to find most of the impossible pieces in Tables 15-17.

Algorithm 2 was not sufficient for some pieces and we had to rely on other methods. Tom Sirgedas [13] presented a nice proof of the impossibility of the last piece in Table 16. His proof is based on the fact that this piece cannot tile any $3 \times n$ rectangle.

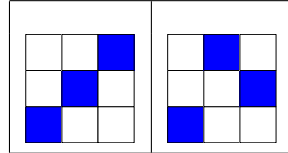


Table 15: Impossible pieces for $n=3$ and $k=2$.

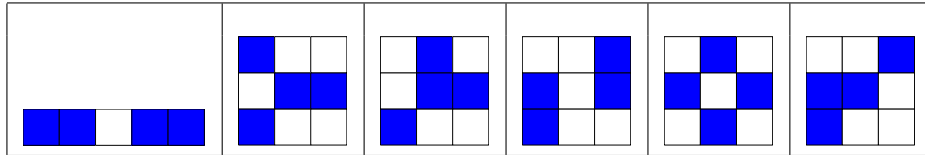


Table 16: Impossible pieces for $n=4$ and $k=1$.

| | | | | |
|--|--|--|--|--|
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

Table 17: Impossible pieces for $n=5$ and $k=1$.

3.4 Unknown polyominoes

We were not able to determine the status of a small number of polyominoes (Table 18). The table lists bounds on the minimum and maximum sides of the rectangle, which should help future solvers. We obtained the maximum results in column 3 by calling Algorithm 1 and incrementally increasing the dimensions of the grid. For the minimum results we used a different algorithm, but we do not describe it here.

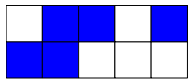
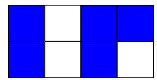
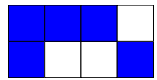
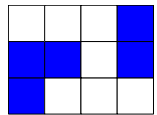
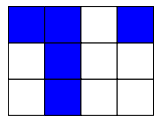
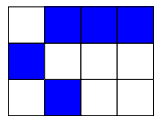
| Piece | min side cannot be $\leq K$ | max side cannot be $\leq K$ |
|---|-----------------------------|-----------------------------|
|  | 18 | 26 |
|  | 17 | 20 |
|  | 20 | 28 |
|  | 20 | 28 |
|  | 20 | 24 |
|  | 19 | 26 |

Table 18: Results for unknown pieces.

4 Future work

Future work will focus on creating faster and more powerful solvers for this problem. Our efforts will concentrate on determining whether the unknown polyominoes listed in Table 18 are rectifiable. We would also like to obtain results for a larger set of polyominoes, such as those listed in Table 19.

There are many more tiling problems that can be explored with holey polyominoes. We can investigate tiling of the plane or tiling of larger copies of polyominoes. We can also attempt tiling with sets of polyominoes. Finally, we can look at the *compatibility problem* - given a pair of polyominoes find a figure that can be tiled with each one.

| n | k | Total |
|---|---|-------|
| 3 | 3 | 17 |
| 3 | 4 | 32 |
| 4 | 2 | 60 |
| 4 | 3 | 151 |
| 5 | 2 | 302 |

Table 19: Unexplored classes of polyominoes.

5 Conclusions

We introduced a new type of polyominoes, affectionately called the holey polyominoes. These polyominoes differ from regular ones by containing some invisible squares. We showed how these polyominoes can tile rectangles. We analysed the rectifiability of all such polyominoes up to 5 visible squares. We were able to determine the status (rectifiable or not) of all but 6 polyominoes.

6 Acknowledgements

We would like to thank Greg O’Keefe for his valuable suggestions. We are grateful to Tom Sirgedas for showing that one of our unknown polyominoes is in fact unrectifiable.

References

- [1] V. Chvatal, D. A. Klarner, and D. E. Knuth. Selected combinatorial research problems. Technical Report STAN-CS-72-292, Stanford University, 1972.
- [2] D. Coppersmith. Each four-celled animal tiles the plane. *Combinatorial Theory*, (40):444–449, 1985.
- [3] E. Friedman. Polyominoes in rectangles. <http://www2.stetson.edu/~efriedma/order/index.html>. Accessed: 17-10-2014.
- [4] E. Friedman. Tiling rectangles with one-dimensional disconnected polyominoes. <http://www2.stetson.edu/~efriedma/mathmagic/0299.html>. Accessed: 27-10-2015.
- [5] S. W. Golomb. Tiling with polyominoes. *Combinatorial Theory*, (1):280–296, 1966.
- [6] S. W. Golomb. *Polyominoes: Puzzles, Patterns, Problems, and Packings*. Princeton University Press, 1996.
- [7] B. Gordon. Tilings of lattice points in euclidean n-space. *Discrete Mathematics*, 29(2):169 – 174, 1980.
- [8] E. Harshbarger. Eric Harshbarger’s pentominoes page. http://www.ericharshbarger.org/pentominoes/article_09.html. Accessed: 28-03-2014.
- [9] D. A. Klarner. Some results concerning polyominoes. *Fibonacci Quarterly*, (3):9–20, 1965.

- [10] D. A. Klarner. Packing a rectangle with congruent n -ominoes. *Combinatorial Theory*, (7):107–115, 1969.
- [11] M. Reid. Rectifiable polyominoes. http://www.cflmath.com/Polyomino/rectifiable_data.html. Accessed: 27-10-2015.
- [12] M. Reid. References for rectifiable polyominoes. http://www.cflmath.com/Polyomino/rectifiable_bib.html. Accessed: 27-10-2015.
- [13] T. Sirgedas. Disconnected polyomino can't tile rectangle. <https://www.youtube.com/watch?v=S7-LT2Tqkmg>. Accessed: 28-10-2015.