Contributions to Discrete Mathematics

Volume 15, Number 2, Pages 95–131
ISSN 1715-0868

A NEW MATHEMATICAL MODEL FOR TILING FINITE REGIONS OF THE PLANE WITH POLYOMINOES

MARCUS R. GARVIE AND JOHN BURKARDT

Abstract. We present a new mathematical model for tiling finite subsets of $\mathbb{Z}^2$ using an arbitrary, but finite, collection of polyominoes. Unlike previous approaches that employ backtracking and other refinements of 'brute-force' techniques, our method is based on a systematic algebraic approach, leading in most cases to an underdetermined system of linear equations to solve. The resulting linear system is a binary linear programming problem, which can be solved via direct solution techniques, or using well-known optimization routines. We illustrate our model with some numerical examples computed in MATLAB. Users can download, edit, and run the codes from `http://people.sc.fsu.edu/~jburkardt/m_src/polyominoes/polyominoes.html`. For larger problems we solve the resulting binary linear programming problem with an optimization package such as CPLEX, GUROBI, or SCIP, before plotting solutions in MATLAB.

1. Introduction and motivation

Consider a planar square lattice $\mathbb{Z}^2$. We refer to each unit square in the lattice, namely $[\tilde{j}-1,\tilde{j}]\times[\tilde{i}-1,\tilde{i}]$, as a *cell*. A *polyomino* is a union of a finite number of edge-connected cells in the lattice $\mathbb{Z}^2$. We assume that the polyominoes are simply-connected. The *order* (or *area*) of a polyomino is the number of cells forming it. The polyominoes of order n are called n-*ominoes* and the cases for $n = 1, 2, 3, 4, 5, 6, 7, 8$ are named monominoes, dominoes, triominoes, tetrominoes, pentominoes, hexominoes, heptominoes, and octominoes, respectively. Golomb introduced polyominoes in a 1965 book [32] (revised and reissued in 1994 [34]). An in-depth treatment of this subject area can be found in [31] and in a collection of essays edited by A. J. Guttman [39]. See also the comprehensive text by Grünbaum and Shephard [37] and the many references therein.

We broadly classify polyominoes as follows (see [31]). A *fixed polyomino* is an equivalence class of polyominoes that are equivalent under translations,

Received by the editors February 6, 2019, and in revised form July 9, 2020.

2010 *Mathematics Subject Classification.* 05B50, 52C20, 90C05, 90C10.

Key words and phrases. polyominoes, tiling, integer linear programming, optimization, MATLAB.

Corresponding author: Marcus Garvie.

so each of the eight tetrominoes illustrated in Figure 1 represent a different fixed polyomino. A *free polyomino* is an equivalence class of polyominoes that are equivalent under translations, rotations, and reflections, so the eight tetrominoes illustrated in Figure 1 represent the same free polyomino. There

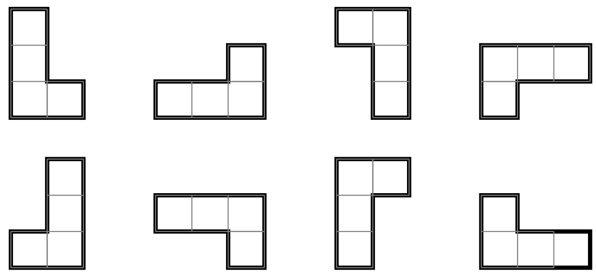


FIGURE. 1. *The* 8 *fixed L-shaped tetrominoes.*

are also *one-sided polyominoes* that are equivalence classes of polyominoes that are equivalent under rotations and translations. For example, the first four polyominoes in the above series represent the same one-sided polyomino.

Considerable effort has been devoted to the problem of enumerating polyominoes as a function of area, or perimeter. Although the general problems are still open, asymptotic formulae and formulae for special cases have been derived [12, 18, 19, 21, 31, 35, 39, 43, 44, 51, 56, 66].

Much effort has also been applied to the problem of tiling the plane with a single polyomino, or with a finite set of polyominoes [8, 23, 33, 38, 72, 75, 78]. The question of whether a finite set of polyominoes can tile the infinite plane is undecidable [8, 73], and determining whether one can tile a finite region of the plane with a given set of polyominoes is in general $\mathcal{NP}$-complete [53]. However, there are special classes of problems that can be solved efficiently. For example, the powerful methods of Combinatorial Group Theory have been successfully applied in a number of interesting cases to prove whether a region can be tiled with a given set of polyominoes, which are generally stronger than classical colouring arguments [15, 60, 61, 68, 76]. There is a large literature on the many theoretical and computational investigations of tiling finite regions of the plane with a given set of polyominoes, and the following list is far from complete: [2, 3, 4, 9, 26, 27, 28, 29, 42, 55, 57, 65, 67, 69, 70, 71, 77].

As the problem of tiling regions of the plane is in general $\mathcal{NP}$-complete, it is not surprising that existing algorithms for tiling with polyominoes use refinements of 'brute-force' techniques for exhaustively finding solutions with a computer. Tiling problems are almost always solved using *backtracking* [30, 47], which is a recursive procedure for pruning the search tree of a combinatorial problem. Backtracking is a procedure appropriate for a problem whose solution can be described as a sequence of steps, each of which involves a choice from some set of options. For example, a maze may be represented as an abstract graph whose nodes are labelled alphabetically. If we wish to search for a path from node A to node Z, a backtracking method can be employed. Starting from node A, the backtracking procedure constructs a tentative path by moving from the current node to a previously unvisited neighbour. Since there may be several choices for such a step, it randomly chooses a node to move to, storing any unchosen options for future exploration. After each move, there are three possibilities:

1. The new node is the goal Z, so we are done. Report the path.
2. The new node is not the goal, but it does connect to other nodes that have not been explored. Choose one, add it to the path, move there, and remember all other unchosen nodes for later options.
3. The new node is not the goal, and does not connect to any unexplored nodes. Remove this node from the path, 'backtracking' to the previously visited node.

Assuming the graph is finite, the backtracking procedure will methodically arrive at a path to the exit, or end up back at the starting node if there is no such path.

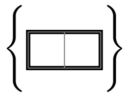
Donald Knuth [45, 46] implemented a combinatorial search algorithm called *Dancing Links* (also called 'DLX'). The Dancing Links algorithm is a recursive, nondeterministic, backtracking algorithm that finds all solutions to the exact cover problem, and has been applied to various problems in addition to tiling with polyominoes, for example: Steiner systems [10], chessboard separation problems [13], general lattice problems [52], hexomino puzzles [16], and a problem in artificial intelligence [5]. Additional backtracking solvers include an algorithm by Fletcher [22], the 'de Bruijn algorithm' [17] which is very similar, and a modern solver by Matthew Busche [11], called 'POLYCUBE', which is optimized in C++ and incorporates elements from both the DLX and de Bruijn to implement a suite of algorithms and techniques. An alternative approach using 'brute-force' for tiling with polyominoes uses evolutionary computation with fitness functions [4].

The main aim of this paper is to answer the following questions:

Question. *Is there is a general systematic mathematical alternative to the backtracking methods for tiling with polyominoes? And if the answer is 'yes', does it yield a flexible algorithmic approach for finding solutions?*

The motivation for our work is contained in an example by Michael Reid [68] for tiling a simple region with copies of a single polyomino. Reid was concerned with theoretical aspects of applying the tile homotopy method for proving the impossibility of specific tiling problems and acknowledged that his group theory method could not be made algorithmic. However, Reid's example is the only case we are aware of that turns a polyomino tiling problem into an algebraic problem, and for the purposes of introducing our method his example is reproduced below.

Consider tiling the region in Figure 2 with dominoes,



all orientations permitted:

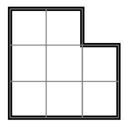
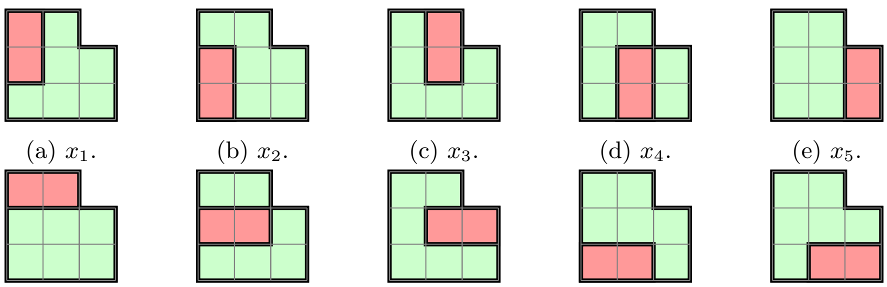


FIGURE. 2. *Region to tile with dominoes.*

Reid introduced a variable $x_i \in \{0,1\}$ for how many times each placement of a domino is used to tile the region, illustrated in Figure 3: In any tiling



(a) x_1. (b) x_2. (c) x_3. (d) x_4. (e) x_5.
(f) x_6. (g) x_7. (h) x_8. (i) x_9. (j) x_{10}.

FIGURE. 3. *Possible tile placements and associated variables.*

of Figure 2 each of the 8 cells must be covered exactly once, yielding the following system of 8 linear equations in 10 unknowns :

$$
\begin{array}{l}
x_1 + x_6 = 1 \\
x_3 + x_6 = 1 \\
x_1 + x_2 + x_7 = 1 \\
x_3 + x_4 + x_7 + x_8 = 1 \\
x_5 + x_8 = 1 \\
x_2 + x_9 = 1 \\
x_4 + x_9 + x_{10} = 1 \\
x_5 + x_{10} = 1
\end{array}
\tag{1.1}
$$

Observe that the total number of dominoes used to tile the region is given by

$$\sum_{i=1}^{10} x_i = (\text{number of cells in region})/2 = 4.$$

This is incorporated in the above system as adding the equations yields $2\sum_{i=1}^{10} x_i = 8$, which gives the same result. We have converted a tiling problem into an algebraic problem. A tiling of the region corresponds to a solution of the algebraic system, however the converse is not necessarily true. Nonbinary 'solutions' do not correspond to a tiling of the region.

We generalize the approach used in this example for tiling arbitrary finite subsets of $\mathbb{Z}^2$ using a given number of free polyominoes, with a known number of copies of each polyomino. The model formulation is easily converted into an algorithm that can be coded on a computer. The resulting linear system of equations can be solved via a direct approach, or as a binary linear programming problem using well-known optimization routines. The binary linear programming problem is another $\mathcal{NP}$-complete problem [25], but hopefully more efficient than a pure trial-and-error approach.

The remaining parts of this paper are organized as follows. In Section 2 the mathematical model for tiling with polyominoes is derived and stated and methods for solving the model are discussed in Section 3. In Section 4 we present numerical results for small problems solved in MATLAB using a direct solution method, and for larger problems solved using high-performance optimization software packages. Conclusions are made in Section 5. Finally, in Appendix A the main notation of this paper is summarized, and in Appendix B some implementation details are given for using the optimization software to find multiple feasible solutions.

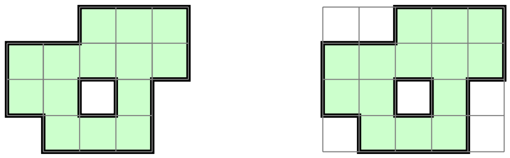
2. Constructing the mathematical model

Consider an arbitrary union of a finite number of edge-connected cells in the lattice $\mathbb{Z}^2$ denoted R with order c_R that is connected, but not necessarily simply-connected (i.e., R is allowed to have 'holes'). Using a given finite set of free polyominoes we aim to tile R. A *tiling* of a region R is an arrangement of our set of polyominoes that covers every cell of R exactly once. Our approach for representing polyominoes as unique binary matrices is similar to that in [6]. Drawing R in the positive quadrant of the plane, define the *rectangular hull* of R to be the smallest rectangle containing R. The lattice associated with any $r \times c$ rectangular hull of R can be expressed as

$$\mathfrak{B} := \{(\tilde{j}, \tilde{i}) \mid 0 \le \tilde{i} \le r; \quad 0 \le \tilde{j} \le c; \quad \tilde{i}, \tilde{j} \in \mathbb{Z}\}.$$

We then define a binary matrix $B \in \{0,1\}^{r \times c}$ such that the $(\tilde{i}, \tilde{j})$ entry of B is equal to 1 if the unit square $[\tilde{j}-1, \tilde{j}] \times [\tilde{i}-1, \tilde{i}]$ of $\mathfrak{B}$ is a cell of R, otherwise 0. If R is simply-connected and a rectangle, then all the entries in B will be equal to 1. An example is illustrated in Figure 4:

Assume we have a collection of n_s free polyominoes $\mathfrak{S} := \{P_i\}_{i=1}^{n_s}$, where the order of each P_i is denoted c_i. Allowing for a combination of rotations, reflections and translations, assume each polyomino P_i fits s_i ways into R. We tile R with d_i, $1 \le d_i \le s_i$, copies of each free polyomino $P_i \in \mathfrak{S}$,



(a) R, (b) R and $\mathfrak{B}$.

$$\begin{pmatrix} 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

(c) B.

FIGURE. 4. *Constructing the binary matrix B.*

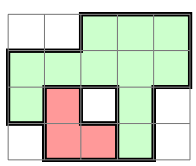
$i = 1, \dots, n_s$, thus

$$\sum_{i=1}^{n_s} c_i d_i = c_R, \tag{2.1}$$

and the total number of polyominoes used to tile R is

$$n_p := \sum_{i=1}^{n_s} d_i. \tag{2.2}$$

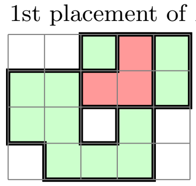
We associate the jth placement of polyomino P_i into R with a binary matrix $A^{i,j} \in \{0,1\}^{r\times c}$, $j = 1, \dots, s_i$, defined in the same way that the binary matrix B is defined. That is, the $(\tilde{i}, \tilde{j})$ entry of $A^{i,j} \in \{0,1\}^{r\times c}$ is equal to 1 if the unit square $[\tilde{j}-1, \tilde{j}] \times [\tilde{i}-1, \tilde{i}]$ of $\mathfrak{B}$ is a cell of P_i, otherwise 0. Two placements of a triomino into a region R and their associated binary matrices are illustrated in Figure 5.



(a) 1st placement of P_1.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

(b) $A^{1,1}$.



(c) 2nd placement of P_1.

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

(d) $A^{1,2}$.

FIGURE. 5. *Constructing the binary matrices $A^{i,j}$.*

For a valid configuration we must have $P_i \cap R = P_i$, i.e. each cell of P_i must not overlap with a cell outside of R or with any cells that are part of a hole in R.

For notational simplicity we avoid using a separate notation for enumerating the fixed polyominoes associated with a given free polyomino P_i. However, it is clear that the number of ways a particular free polyomino P_i fits in the region R is in general greater then the number of ways the associated fixed polyominoes for P_i fits into R[1].

The set of binary matrices $A^{i,j}$ associated with fitting the polyominoes P_i into R is called Series i, given by

$$\text{Series } i := \{A^{i,j} \in \{0,1\}^{r\times c} \mid j = 1,\ldots,s_i\}, \quad i = 1,\ldots,n_s.$$

The total number of binary matrices $A^{i,j}$ is given by

$$n := \sum_{i=1}^{n_s} s_i. \tag{2.3}$$

Before the model can be formulated and solved for a particular case, there is the practical task of finding all binary matrices $A^{i,j}$ in each series. Initially for each free polyomino P_i we find all the associated fixed polyominoes obtained by appropriate combinations of rotations and reflections. Depending on the symmetry of the polyomino this will be either 1, 2, 4, or 8. For each fixed polyomino we exhaustively find all possible ways in which the fixed polyomino fits into R, and for each configuration we calculate the associated binary matrix. The number of choices a particular fixed polyomino fits into R will depend on divisibility conditions concerning the dimensions of the rectangular hulls of P_i and R. As this is elementary (but tedious) we omit further details.

We introduce a variable $\alpha_{i,j} \in \{0,1\}$ for how many times the jth placement of P_i is used to tile R. The mathematical description of tiling the region R with polyominoes in $\mathfrak{S}$ is given by the following problem:

Problem I.

Seek parameters $\alpha_{i,j} \in \{0,1\}$, $1 \le i \le n_s$, $1 \le j \le s_i$ such that

$$\sum_{i=1}^{n_s}\sum_{j=1}^{s_i} \alpha_{i,j}A^{i,j} = B, \tag{2.4a}$$

$$\text{subject to } \sum_{j=1}^{s_i} \alpha_{i,j} = d_i, \quad 1 \le d_i \le s_i, \quad i = 1,\ldots,n_s. \tag{2.4b}$$

The constraints (2.4b) enforce the conditions that we must use exactly d_i polyominoes from each Series i to tile R.

Remark 2.1. *In general, the solutions of equation* (2.4a) *subject to the constraints* (2.4b) *(if they exist) are rational (see Theorem 2.18). If the solutions of this* relaxed problem *are not binary then we can no longer interpret each d_i as the number of copies of a polyomino P_i used to tile a region R. For example, consider $d_1 = 1$, $s_1 = 3$, with $\alpha_{1,1} = \alpha_{1,2} = \alpha_{1,3} = 1/3$, and $\sum_{j=1}^{s_1} \alpha_{1,j} = 1$, so the number of polyominoes used from Series* 1 *is* 3.

[1]This is because free polyominoes can be rotated, reflected, and translated in R, while fixed polyominoes can only be translated.

Remark 2.2. *It is natural to assume that an additional constraint is necessary in order to fully describe the tiling problem, namely, that the area of the polyominoes used in the tiling must match the area of the region to be tiled. However, it is easy to demonstrate that this constraint is already implicit in the current set of equations. If one sums over all rows and columns of the matrices in equation* (2.4a)*, the right hand side evaluates to c_R and each $A^{i,j}$ becomes c_i, yielding $\sum_{i=1}^{n_s} c_i \sum_{j=1}^{s_i} \alpha_{i,j} = c_R$. Then using equation* (2.4b) *we obtain $\sum_{i=1}^{n_s} c_i d_i = c_R$.*

We consider the mathematical description of tiling the region R with copies of a *single* polyomino in $\mathfrak{S}$ ($n_s = 1$) as a separate problem. Problem I represents the general case, in which several distinct polyominoes are to be used. Technically, this includes the case where only one polyomino shape is to be employed; however this case will be termed Problem II, since it occurs frequently in the literature, is easier to describe and visualize, and in some cases can be simpler to analyze or simulate computationally.

Problem II.

Seek parameters $\alpha_{1,j} \in \{0, 1\}$, $1 \le j \le s_1$ such that

$$\sum_{j=1}^{s_1} \alpha_{1,j} A^{1,j} = B. \tag{2.5}$$

Remark 2.3. *In general, the solutions of equation* (2.5) *(if they exist) are rational (see Theorem 2.18). If the solutions of this* relaxed problem *are not binary then d_1 may not be a positive integer (see Example 4.4).*

Remark 2.4. *If one sums over all rows and columns of the matrices in equation* (2.5)*, the right hand side evaluates to c_R, and each $A^{1,j}$ evaluates to c_1, yielding*

$$c_1 \sum_{j=1}^{s_1} \alpha_{1,j} = c_R = c_1 d_1 \implies \sum_{j=1}^{s_1} \alpha_{1,j} = d_1, \quad 1 \le d_1 \le s_1.$$

Thus the constraint on the variables of the problem when we tile with copies of a single polyomino is automatically incorporated into the problem (cf. Remark 2.2).

For a given set of free polyominoes $\mathfrak{S}$, there may be many possible ways of satisfying the area equation $\sum_{i=1}^{n_s} c_i d_i = c_R$, depending on how many copies of each polyomino P_i are used. If a single polyomino P_1 is used to tile R then it must be used $d_1 = c_R/c_1$ times. On the other hand, if all the free polyominoes are used exactly once, then $d_i = 1$ for $i = 1, \ldots, n_s$ so $\sum_{i=1}^{n_s} c_i = c_R$. If the number of copies d_i of each polyomino P_i are not specified then we have a linear Diophantine equation in n_s unknowns to solve for:

$$c_1 d_1 + c_2 d_2 + \cdots + c_{n_s} d_{n_s} = c_R. \tag{2.6}$$

Each positive integer solution may correspond to a different tiling problem. The following theorem concerns the existence of nonnegative integer solutions to equation (2.6), where $\gcd(c_1, c_2, \ldots, c_{n_s})$ denotes the greatest common divisor of the coefficients $c_1, c_2, \ldots, c_{n_s}$:

Theorem 2.5 (see [64]). *For fixed coefficients c_i, $i = 1, \ldots, n_s$ there exists an integer $\mathcal{N}$ such that*

(i) $\gcd(c_1, c_2, \ldots, c_{n_s}) \mid c_R$, *and*
(ii) $c_R \geq \mathcal{N}$

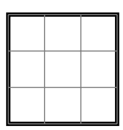
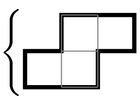
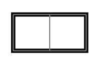
together form a sufficient (but not necessary) condition for a nonnegative integer solution of equation (2.6)*.*

The result follows immediately from Theorem 1.0.1 in [64] after dividing equation (2.6) through by $\gcd\{c_i\}_{i=1}^{n_s}$. Theorem 2.5 tells us that in general, if either $\gcd\{c_i\}_{i=1}^{n_s}$ does *not* divide c_R, or if the target region c_R is not sufficiently large, then there is no nonnegative integer solution of equation (2.6), i.e., in particular, there is no positive integer solution (recall, we assume all coefficients d_i are strictly positive).

Theorem 2.6. *If equation* (2.6) *does not possess a positive integer solution then $\mathfrak{S}$ does not tile R.*

Proof. By assumption there is at least one d_i, $1 \leq i \leq n_s$ that is *not* a positive integer. Thus from equation (2.4b) one or more of the coefficients $\alpha_{i,j}$ are not binary and so $\mathfrak{S}$ does not tile R. □

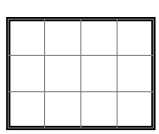
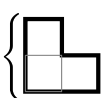
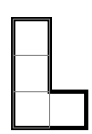
Example 2.7. *Let*

$P =$  *with* $\mathfrak{S} = \{$ ,  $\}$,

all rotations and reflections permitted. We have $c_1 = 4$, $c_2 = 2$, and $c_R = 9$, so as $\gcd(c_1, c_2)$ does not divide c_R, it follows from Theorem 2.5 that d_1 or d_2 is not a positive integer, and so from Theorem 2.6 we conclude that $\mathfrak{S}$ does not tile R.

Remark 2.8. *Theorem 2.5 implies that even when $\gcd(c_1, c_2, \ldots, c_{n_s})$ does divide c_R, if c_R is not sufficiently large we may still have no nonnegative integer solution to equation* (2.6) *(see Example 2.9). From a practical perspective this is important. If the number of polyominoes used to tile R is large, or c_R is large, it can be difficult to know ahead of time whether a nonnegative integer solution to equation* (2.6) *exists without additional information.*

Example 2.9. *Let*

$P =$  *with* $\mathfrak{S} = \{$ ,  $\}$, *all rotations and reflections permitted. We have* $c_1 = 3$, $c_2 = 4$, $c_R = 12$ *and* $\gcd(c_1, c_2)$ *divides* c_R. *However, as the linear Diophantine equation* $3d_1 + 4d_2 = 12$ *does not have a positive integer solution it follows from Theorem 2.6 that* $\mathfrak{S}$ *does not tile* R.

The problem of determining how many nonnegative integer solutions to equation (2.6) exist is well-known to be $\mathcal{NP}$-complete [62, p. 376] and some methods for computing the solutions are given in [64].

Remark 2.10. *If the constraints* (2.4b) *are omitted in Problem I then the number of copies of each polyomino used to tile the region is unspecified, which greatly expands the number of possible solutions. Indeed, some of the coefficients* d_i *may be zero, which is not possible when the constraints are imposed. (For instance, in Example 2.9 four copies of the triomino will tile the region.) For simplicity in the sequel we assume that the coefficients* d_i *are given and the constraints are incorporated in the problem.*

After multiplying the binary matrices $A^{i,j}$ on the left hand side of equation (2.4a) by the parameters $\alpha_{i,j}$ and then taking the sum of the resulting matrices, we equate the *nonzero*[2] entries of B with the corresponding entries on the left hand side of equation (2.4a) yielding c_R linear equations (see equation (2.1)). The ordering of the equations corresponds to taking entries in the matrix B row-wise, left to right, top to bottom. Consider first the case that $n_s > 1$. Adding the constraints (2.4b) yields a total of m equations in n unknowns, with the following matrix form:

Linear System I.

$$M\boldsymbol{\alpha} = \widehat{\mathbf{b}}, \quad M \in \{0,1\}^{m\times n}, \quad \boldsymbol{\alpha} \in \{0,1\}^n, \tag{2.7}$$

where $m = c_R + n_s$, $n = \sum_{i=1}^{n_s} s_i$ and

$$\{\widehat{\mathbf{b}}\}_i = \begin{cases} 1 & \text{for } i = 1, \ldots, c_R \\ d_i & \text{for } i = c_R + 1, \ldots, m \end{cases},$$

$$\boldsymbol{\alpha} = \left(\alpha_{1,1} \ldots \alpha_{1,s_1} \mid \alpha_{2,1} \ldots \alpha_{2,s_2} \mid \ \ldots \ \mid \alpha_{n_s,1} \ldots \alpha_{n_s,s_{n_s}}\right)^T.$$

The vertical lines in the solution vector indicate the partition of unknowns according to the series they are associated with. As a consequence of the constraints (2.4b) exactly d_i coefficients associated with each Series i are equal to 1, with the remaining coefficients equal to zero.

Remark 2.11. *Let* $\widehat{M}$ *be the matrix composed of the first* c_R *rows of* M, *i.e. the coefficient matrix of Linear System I without the constraint equations. Each row in* $\widehat{M}$ *corresponds to a nonzero entry in* B, *while each column corresponds to an unknown in* $\boldsymbol{\alpha}$. *The columns of* $\widehat{M}$ *are equal to the entries in the binary matrices* $A^{i,j}$ *written row-wise. (We only take the entries in*

[2] Zero entries of B correspond to cells not in R.

$A^{i,j}$ that correspond to nonzero entries in B.) Thus if we sum the first c_R equations in Linear System I we obtain

$$c_1 \sum_{j=1}^{s_1} \alpha_{1,j} + c_2 \sum_{j=1}^{s_2} \alpha_{2,j} + \cdots + c_{n_s} \sum_{j=1}^{s_{n_s}} \alpha_{n_s,j},$$

which is a linear combination of the last n_s constraint equations with weights $c_1, c_2, \ldots, c_{n_s}$[3]. Thus the reduced row echelon form of the augmented matrix associated with Linear System I will always have a row of zeros in the bottom row, which has relevance to the solvability of the system (see Section 3).

Problem II ($n_s = 1$) yields the following system of m equations in n unknowns, with the matrix form:

Linear System II.

$$M\boldsymbol{\alpha} = \widehat{\mathbf{b}}, \quad M \in \{0,1\}^{m\times n}, \quad \boldsymbol{\alpha} \in \{0,1\}^n, \tag{2.8}$$

where $m = c_R$, $n = s_1$ and

$$\{\widehat{\mathbf{b}}\}_i = 1 \quad \text{for } i = 1, \ldots, c_R \tag{2.9}$$

$$\boldsymbol{\alpha} = \left(\alpha_{1,1} \ldots \alpha_{1,s_1}\right)^T.$$

Remark 2.12. *In Linear System II if $d_1 = c_R/c_1$ is not a positive integer then the single free polyomino does not tile R.*

Remark 2.13. *By the terms* relaxed Linear System I *and* relaxed Linear System II *we mean the corresponding relaxed linear systems with (if they exist) rational solutions (see Remark 2.1 and Remark 2.3).*

Tiling with sets of polyominoes typically yields underdetermined linear systems of equations with multiple (binary) solutions. Linear System I is underdetermined if $c_R + n_s < n$, while Linear System II is underdetermined if $c_R < s_1$.

There appears to be little we can say about the number of binary solutions, denoted N, of Linear System I or II without some additional information. If we have a consistent underdetermined linear system then either we have no binary solutions ($N = 0$), a unique binary solution ($N = 1$), or a finite number of binary solutions ($N > 1$). For example, the problem of tiling the 6×10 rectangle with the full set of 12 free pentominoes has $N = 2339$ solutions, excluding trivial variations obtained by reflecting or rotating the whole rectangle [40], and the problem of tiling any rectangle with 20 squares using the full set of 5 free tetrominoes has no solution [49].

Regarding the solvability of Linear Systems I and II the usual condition involving rank can be computed for determining the consistency of the system (see Section 3.1 for more details). We may also be able to determine ahead of time if positive rational solutions exist. If positive solutions do

[3] This is equal to c_R, cf. Remark 2.2.

not exist then certainly there are no binary solutions. To this end we state one of the many (equivalent) results concerning the existence of solutions to linear inequalities, which is a classic result in linear programming and optimization:

Theorem 2.14 (Farkas' Lemma, see [20],[24, Lemma 1, p. 318])**.** *Let $A \in \mathbb{R}^{m\times n}$ and $\mathbf{b} \in \mathbb{R}^m$, then exactly one of the following alternatives hold:*

(i) There exists $\mathbf{x} \in \mathbb{R}^n$ satisfying $A\mathbf{x} = \mathbf{b}$ and $\mathbf{x} \geq \mathbf{0}$.
(ii) There exists $\mathbf{y} \in \mathbb{R}^m$ satisfying $\mathbf{y}^T A \geq \mathbf{0}^T$ and $\mathbf{y}^T\mathbf{b} < 0$.

Remark 2.15. *Inequalities involving vectors are understood component-wise, i.e. $\mathbf{x} \geq \mathbf{0}$ means each component of $\mathbf{x}$ is greater than or equal to 0. A similar interpretation is given to the statement $\mathbf{y}^T A \geq \mathbf{0}^T$.*

Remark 2.16. *The statement $\mathbf{y}^T A \geq \mathbf{0}^T$ is equivalent to the statement $A^T\mathbf{y} \geq \mathbf{0}$. (Here, $\mathbf{0}^T$ is the $1 \times n$ zero vector.)*

Remark 2.17. *In Linear System I when we use only a single polyomino from each series then the vector $\mathbf{b}$ in Theorem 2.14 is a vector of all '1's and so the statement $\mathbf{y}^T\mathbf{b} < 0$ becomes $\sum_{i=1}^{m} y_i < 0$, where $y_i := \{\mathbf{y}\}_i$.*

If A is a binary matrix and the components of $\mathbf{b}$ are positive integers, then the above result can be strengthened to the rational solution case:

Theorem 2.18. *If $A \in \{0,1\}^{m\times n}$ and $\mathbf{b} \in \mathbb{N}^m$ then Theorem 2.14 holds with $\mathbf{x} \in \mathbb{Q}^n$.*

Proof. Suppose the system of equations $A\mathbf{x} = \mathbf{b}$ is consistent. If we have a unique solution then the Gaussian elimination algorithm with exact arithmetic and partial pivoting applies a finite number of operations ($+$, $-$, $\times$, $\div$) to the entries of $\mathbb{N} \cup \{0\}$ in the augmented matrix $[A|\mathbf{b}]$. Thus the solution must be rational. If we have an infinite number of solutions then the solutions are guaranteed to be rational if we choose the free variables to be rational. $\square$

In the next section we illustrate the applicability of the Farkas' Lemma (Theorem 2.14) to our mathematical model with two problems (Example 4.3 and Example 4.4) that are simple enough to be done without the need for optimization software. However, for larger problems, standard linear programming techniques applied to the problem

$$\mathbf{ILP1}: \quad \begin{cases} \text{minimize } \mathbf{y}^T\mathbf{b} \\ \text{subject to } \mathbf{y}^T A \geq \mathbf{0}^T \\ \mathbf{y} \text{ unrestricted in sign} \end{cases}$$

guarantees the existence of a positive rational solution for nonnegative optimal values, i.e. for $\mathbf{y}^T\mathbf{b} \geq 0$, which is a necessary condition for the existence of a binary solution to $A\mathbf{x} = \mathbf{b}$.

3. Solving the model

Numerous MATLAB codes, written in support of this research work, are available at the website `http://people.sc.fsu.edu/~jburkardt/m_src/polyominoes/polyominoes.html`. Because of some efficiency issues, there are separate codes for the cases where we tile with copies of a single polyomino ('monohedral case'), or tile with copies of several different polyominoes ('multihedral case'). Small cases ($n < 30$) can be solved entirely with these codes, but for even moderate-sized problems ($30 < n < 200$), it is best to write out the linear system as an LP file, and then rely on high-performance integer linear programming packages such as CPLEX, GUROBI, or SCIP for the solution. Once that is obtained, there are MATLAB codes to read solutions and produce plots or printouts (see Section 3.2 for more details).

3.1. **Direct solution algorithm with MATLAB.** We employed a direct solution method for solving small problems ($n < 30$) in MATLAB (R2018A), run on a MacBook Pro (OS X 10.13.6) with 16 GB of memory and 2.7 GHz Intel Core i7.

The method of computing the solution to a linear system $Ax = b$ is called a *direct method* if the solution can be computed using exact arithmetic with a finite number of operations. Binary linear programming problems of this type can trivially be solved with $\mathcal{O}(2^n mn)$ floating point operations via exhaustive search [59], but this is prohibitive for large problems. We describe below how we reduce the computational cost of finding binary solutions for our problem using a direct method, which is then illustrated for some small problems in Section 4.1.

The first step in our approach for solving Linear System I or II via a direct method is to reduce the associated augmented matrix $[M|\widehat{\mathbf{b}}]$ to reduced row echelon form (RREF), denoted $[A|\mathbf{b}]$. Let $r := \operatorname{rank}(M)$. If $r < \operatorname{rank}([M|\widehat{\mathbf{b}}])$ the system is inconsistent, which implies there does not exist a tiling of the region R. Assume the system is consistent. If $r = n$, then we have the unique (rational) solution $\boldsymbol{\alpha} = \mathbf{b}$. If this is a binary solution then there is a unique tiling of the region R using the polyominoes in $\mathfrak{S}$. If $r < n$ we must identify the $f := n - r$ free variables and the series to which they belong. Then to find any binary solutions we choose all possible *allowable* binary choices of the free variables, and for each choice check via a back-substitution procedure (described below) if the solution of the linear system is binary. The number of binary choices for the free variables depends on the series they belong to and the number of copies of each polyomino used in each series. In any particular Series i if we have f_i free variables then we need to consider all ways for at most d_i of the free variables to be equal to '1', with the remaining free variables in that series equal to '0'. This leads to the following theorem for the number of binary choices of the free variables we must consider in the direct solution method:

Theorem 3.1. *Consider $f \geq 1$ free variables with f_i free variables in Series i, $0 \leq f_i \leq s_i$, $i = 1, \ldots, n_s$, where $f = \sum_{i=1}^{n_s} f_i$. Then the number of possible binary choices for the free variables to check in the direct solution method is given by*

$$b_f := \prod_{i=1}^{n_s} b_{f_i} \qquad \text{where} \qquad b_{f_i} := \begin{cases} 2^{f_i} & \text{if } f_i \leq d_i \\ \displaystyle\sum_{k=0}^{d_i} \binom{f_i}{k} & \text{if } f_i > d_i \end{cases}. \tag{3.1}$$

Proof. In any given series if $f_i \leq d_i$ then we need all ways of assigning '1's to the free variables up to a maximum of f_i. That is, we need all ways of choosing $0, 1, \ldots, f_i$ of the f_i free variables equal to '1', yielding $\sum_{k=0}^{f_i} \binom{f_i}{k} = 2^{f_i}$ possibilities. If $f_i > d_i$ then we need all ways of assigning '1's to at most d_i free variables, as we use exactly d_i copies of P_i in Series i. Thus we need all combinations of choosing $0, 1, \ldots, d_{i-1}, d_i$ of the f_i free variables equal to '1', yielding $\sum_{k=0}^{d_i} \binom{f_i}{k}$ possibilities. The result then follows after applying the Fundamental Counting Principle over all series. □

Corollary 3.2. *We have two special cases:*

(i) If exactly one polyomino is used from each series, that is $d_i = 1$ for all i, $n_s > 1$, then equation (3.1) *reduces to*

$$b_f = \prod_{i=1}^{n_s} (1 + f_i).$$

(ii) If copies of a single polyomino are used to tile R, i.e. $n_s = 1$, $d_1 = c_R/c_1$ then equation (3.1) *becomes*

$$b_f = \begin{cases} 2^{f} & \text{if } f \leq d_1 \\ \displaystyle\sum_{k=0}^{d_1} \binom{f}{k} & \text{if } f > d_1 \end{cases}.$$

We employ a simple procedure that facilitates the coding of the back-substitution process for a given binary choice of free variables. Each binary choice of a free variable is enforced by inserting a new equation back into the system. For example, for the ith free variable $\alpha_i = c \in \{0, 1\}$, we insert a new row in the system $A\boldsymbol{\alpha} = \mathbf{b}$ between rows $i-1$ and i with a '1' at the (i,i) position, illustrated below:

$$\begin{array}{cc} & \begin{matrix} & & & i & & & \end{matrix} \\ i & \begin{pmatrix} 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \end{pmatrix} \end{array} \begin{pmatrix} \alpha_i \end{pmatrix} = \begin{pmatrix} c \end{pmatrix},$$

After we have inserted all equations corresponding to the current binary choices of free variables we have an $n \times n$ upper triangular system which we solve via back-substitution. If the 'solution' is binary we have found a solution to the tiling problem. All nonbinary 'solutions' are discarded. For the most efficient implementation of back-substitution, as soon as a nonbinary variable is computed we halt the procedure, and start the back-substitution procedure with the next set of free variables. The above procedure is repeated for all binary choices of free variables. The direct solution method is summarized in Algorithm 1 (Figure 6).

Algorithm 1

$N \leftarrow 0$
Read M and $\widehat{\mathbf{b}}$ from file
Compute $[A|\mathbf{b}]$ (the RREF of $[M|\widehat{\mathbf{b}}]$)
Compute $r := \text{rank}(M)$
if $r < \text{rank}([M|\widehat{\mathbf{b}}])$ **then**
    print `"System is inconsistent"`
else if $r = n$ **then**
    if $\mathbf{b}$ is binary **then**
        $\boldsymbol{\alpha} \leftarrow \mathbf{b}$
        Save binary solution $\boldsymbol{\alpha}$ to file
        print `"There is a unique binary solution"`
    else
        print `"No binary solution exists"`
    end if
else {system has an infinite number of rational solutions}
    $f \leftarrow n - r$
    Identify the f_k free variables in each Series S_k and compute b_{f_k}, for $k = 1, \ldots, n_s$, using equation (3.1)
    Compute $b_f := \prod_{k=1}^{n_s} b_{f_k}$
    for $i = 1$ to b_f **do**
        Assign binary values to free variables $\{f_k\}_{k=1}^{n_s}$
        Solve $A\boldsymbol{\alpha} = \mathbf{b}$ via back-substitution for the variables $\{\alpha_n, \alpha_{n-1}, \ldots, \alpha_1\}$
        if any α_k is not binary **then**
            Break for loop
        end if
        $N \leftarrow N + 1$
        Write binary solution $\boldsymbol{\alpha}$ to file
    end for
    if $N = 0$ **then**
        print `"No binary solution exists"`
    else
        print `"There are" N "binary solutions"`
    end if
end if

FIGURE. 6. *Direct solution method for solving polyomino tiling problem. See Section 3.1 for further details.*

3.2. **Solution using integer linear programming packages.** Excellent noncommercial and commercial software for solving large, real-world integer programming problems have been developed. Among them, we investigated two commercial optimization solvers, namely

- IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer 12.8.0.0 (CPLEX for short), see `https://www.ibm.com/analytics/cplex-optimizer`,
- GUROBI Optimization 7.5.2. (GUROBI for short), see `http://www.gurobi.com`

and one noncommercial solver

- SCIP (Solving Constraint Integer Programs) version 6.0.0., see `http://scip.zib.de`.

These optimization packages carry out the optimization process by branch-and-bound algorithms, including some general purpose heuristics. For a survey of modern advances in the theory of branch-and-bound algorithms see [58].

Linear System I and II can be expressed as integer linear programming problems in the LP format with no objective function, which for medium-sized or large problems ($200 < n < 70,000$) we solved using CPLEX, GUROBI, or SCIP. We found CPLEX to be considerably faster than either GUROBI or SCIP for solving the problems in this paper. The interactive shell commands used for calculating all feasible solutions are given in Appendix B. For further details and for the standard shell commands needed to find a single optimal solution see the links given above. All optimizers were run on a MacBook Pro (OS X 10.13.6) with 16 GB of memory and 2.7 GHz Intel Core i7.

To solve a particular instance of a tiling problem we employed three steps:

(i) Construct the linear system in MATLAB and export the associated LP file to an optimizer [4]
(ii) Solve the LP file with an optimizer and export the solution file back to MATLAB
(iii) Extract the solution(s) from the file produced in (ii) in a form that MATLAB can read and plot the solution(s)

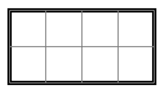
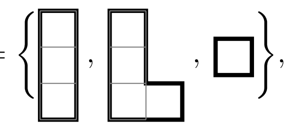
To help other researchers who wish to reproduce our results, or pursue investigations of their own, the MATLAB 'LPmake' files needed to perform step (i) above for the medium to large problems are available from
`http://people.sc.fsu.edu/~jburkardt/m_src/polyominoes.html`
(called `SCRIPTS`) and the MATLAB codes needed to perform step (iii) (called `PLOT_MONO` and `PLOT_MULTI`) for the monohedral and multihedral cases, respectively).

[4]For GUROBI to solve these problems we found it necessary to make the objective function in the LP file blank, instead of the default: 'Obj: 0'.

4. Numerical experiments

4.1. **Small problems solved in MATLAB.** We demonstrate the direct solution method with some examples that illustrate the types of solutions that are possible. Although these examples are very simple, they encapsulate all the features of larger problems.

Example 4.1 (Tiling with 3 polyominoes)**.** *We find all possible ways of tiling the* 2×4 *rectangle*

$P =$  *with* $\mathfrak{S} =$ ,

all rotations and reflections permitted. Considering all possible placements of the three polyominoes in the rectangle yields the series:

$$\textit{Series 1} = \left\{ \underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A^{1,1}}, \underbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A^{1,2}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}}_{A^{1,3}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}}_{A^{1,4}} \right\},$$

$$\textit{Series 2} = \left\{ \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}}_{A^{2,1}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{pmatrix}}_{A^{2,2}}, \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 \end{pmatrix}}_{A^{2,3}}, \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}}_{A^{2,4}}, \right.$$

$$\left. \underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}}_{A^{2,5}}, \underbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{A^{2,6}}, \underbrace{\begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{A^{2,7}}, \underbrace{\begin{pmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A^{2,8}} \right\},$$

$$\textit{Series 3} = \left\{ \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A^{3,1}}, \underbrace{\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A^{3,2}}, \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A^{3,3}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{A^{3,4}}, \right.$$

$$\left. \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}}_{A^{3,5}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}}_{A^{3,6}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{A^{3,7}}, \underbrace{\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A^{3,8}} \right\}.$$

The binary matrix associated with the region R is

$$B = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}.$$

We have $m = c_R + n_s = 8 + 3 = 11$ and $n = s_1 + s_2 + s_3 = 4 + 8 + 8 = 20$. So Problem I becomes

> *Seek parameters $\alpha_{i,j} \in \{0, 1\}$, $1 \le i \le 3$, $1 \le j \le s_i$ such that*
>
> $$\sum_{i=1}^{3} \sum_{j=1}^{s_i} \alpha_{i,j} A^{i,j} = B, \quad \text{subject to} \ \sum_{j=1}^{s_i} \alpha_{i,j} = 1, \quad i = 1, 2, 3,$$

which yields an underdetermined linear system $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$ with 11 equations in 20 unknowns corresponding to Linear System I. If we neglect the last 3 constraint equations, then the columns of M are simply the elements of the binary matrices $A^{i,j}$ written row-wise, taken in the order in which they are given above. The right-hand-side vector is a vector of all ones. The reduced row echelon form of the associated augmented matrix is given by:

$$[A|\mathbf{b}] =$$

$$\begin{array}{c}
\begin{array}{cccc|cccccccc|cccccccc}
\alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & \alpha_{2,4} & \alpha_{2,5} & \alpha_{2,6} & \alpha_{2,7} & \alpha_{2,8} & \alpha_{3,1} & \alpha_{3,2} & \alpha_{3,3} & \alpha_{3,4} & \alpha_{3,5} & \alpha_{3,6} & \alpha_{3,7} & \alpha_{3,8}
\end{array} \\
\left(\begin{array}{cccccccccccccccccccc|c}
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 2 & -1 & 0 & -1 & -1 & -1 & -2 & -1 & 0 & -2 & -1 \\
0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & -2 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & -1 & 2 & 1 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & -1 & -1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right)
\end{array}$$

The variables corresponding to each column are indicated with vertical lines separating the series they are associated with and pivots are circled. The system is clearly consistent with $10 = r < n = 20$, so there are $f = n - r = 10$ free variables:

Series 1:	$-$
Series 2:	$\alpha_{2,3}, \alpha_{2,7}, \alpha_{2,8}$
Series 3:	$\alpha_{3,2}, \alpha_{3,3}, \alpha_{3,4}, \alpha_{3,5}, \alpha_{3,6}, \alpha_{3,7}, \alpha_{3,8}$

In each series only a single free variable can be equal to 1 at a time, or all members of that series can be zero, yielding (see Corollary 3.2, part (i)) $b_f = (1 + 0)(1 + 3)(1 + 7) = 32$ possible binary choices of free variables to check. Solving the linear systems via back-substitution for all possible choices of free variables as described in Section 3.1 yields the following four binary solutions:

$\alpha_{1,1}$	$\alpha_{1,2}$	$\alpha_{1,3}$	$\alpha_{1,4}$	$\alpha_{2,1}$	$\alpha_{2,2}$	$\alpha_{2,3}$	$\alpha_{2,4}$	$\alpha_{2,5}$	$\alpha_{2,6}$	$\alpha_{2,7}$	$\alpha_{2,8}$	$\alpha_{3,1}$	$\alpha_{3,2}$	$\alpha_{3,3}$	$\alpha_{3,4}$	$\alpha_{3,5}$	$\alpha_{3,6}$	$\alpha_{3,7}$	$\alpha_{3,8}$
0	0	0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	**1**	0	0	0	0
0	0	**1**	0	0	0	0	0	0	0	0	**1**	**1**	0	0	0	0	0	0	0
1	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	**1**	0	0	0
0	**1**	0	0	0	0	**1**	0	0	0	0	0	0	0	0	0	0	0	0	**1**

The four corresponding tilings are shown below:

(a) Tile 1. (b) Tile 2. (c) Tile 3. (d) Tile 4.

FIGURE. 7. *The* 4 *possible ways to tile* R.

These tilings are trivial variations of each other obtained by rotating or reflecting the whole rectangle. Thus ignoring these variations yields just one way to tile the region.

Example 4.2 (Tiling with a single polyomino)**.** *We solve the simple example given by Reid* [68] *introduced at the start of this paper. Initially we calculate all binary matrices corresponding to the placements of a domino in the region* R *(illustrated in Figure 3). We have* $m = c_R = 8$, $n = s_1 = 10$, $(n_s = 1)$*. So Problem II becomes*

Seek parameters $\alpha_{1,j} \in \{0,1\}$, $1 \le j \le 10$ *such that*

$$\sum_{j=1}^{10} \alpha_{1,j} A^{1,j} = B,$$

which yields an underdetermined linear system $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$ *with* 8 *equations in* 10 *unknowns corresponding to Linear System II. The columns of* M *are simply the elements of the binary matrices* $A^{i,j}$ *written row-wise, taken in the order in which they are given above, but neglecting the entries in the* $(1,3)$ *positions. The right-hand-side vector is a vector of all ones. The reduced row echelon form of the associated augmented matrix is given by:*

$$[A|\mathbf{b}] = \begin{array}{c} \begin{array}{cccccccccc} \alpha_{1,1} & \alpha_{1,2} & \alpha_{1,3} & \alpha_{1,4} & \alpha_{1,5} & \alpha_{1,6} & \alpha_{1,7} & \alpha_{1,8} & \alpha_{1,9} & \alpha_{1,10} \end{array} \\ \left(\begin{array}{cccccccccc|c} \boxed{1} & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & \boxed{1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & \boxed{1} & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & \boxed{1} & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & \boxed{1} & -1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \boxed{1} & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right) \end{array}$$

The variables associated with each column are indicated and the pivots have been circled. The system is clearly consistent with $7 = r < n = 10$, so there are $f = n - r = 3$ free variables: $\alpha_{1,7}$, $\alpha_{1,9}$, and $\alpha_{1,10}$. As $d_1 = c_R/c_1 = 4$ we have $b_f = 2^3$ binary choices for the free variables to check for (see Corollary 3.2, part (ii)). Solving the linear systems via back-substitution for all possible choices of free variables as described in Section 3.1 yields the following four binary solutions:

$\alpha_{1,1}$	$\alpha_{1,2}$	$\alpha_{1,3}$	$\alpha_{1,4}$	$\alpha_{1,5}$	$\alpha_{1,6}$	$\alpha_{1,7}$	$\alpha_{1,8}$	$\alpha_{1,9}$	$\alpha_{1,10}$
0	**1**	0	**1**	**1**	**1**	0	0	0	0
0	**1**	0	0	0	**1**	0	**1**	0	**1**
1	0	**1**	0	**1**	0	0	0	**1**	0
0	0	0	0	**1**	**1**	**1**	0	**1**	0

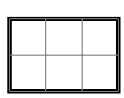
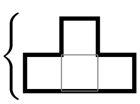
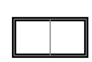
The four corresponding tilings are shown below:

(a) Tile 1. (b) Tile 2. (c) Tile 3. (d) Tile 4.

FIGURE. 8. *The 4 possible ways to tile R.*

Ignoring trivial differences due to rotating and reflecting, the whole tiled region yields just two different tilings.

Example 4.3 (Rational solutions of the relaxed system)**.** *Consider tiling*

$P =$  *with* $\mathfrak{S} = \Big\{$  ,  $\Big\}$,

all rotations and reflections permitted. There is obviously no tiling. We have $c_1 = 4$, $c_2 = 2$, $c_R = 6$, and with $d_1 = d_2 = 1$ there is a solution to the linear Diophantine equation $c_1 d_1 + c_2 d_2 = c_R$, thus a solution to the relaxed Linear System I may exist. We prove there exists a positive rational solution. Initially we calculate all possible binary matrices corresponding to the placements of the two polyominoes in the rectangle.

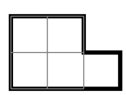
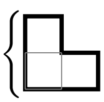
We have $m = c_R + n_s = 6 + 2 = 8$ and $n = s_1 + s_2 = 2 + 7 = 9$, which leads to the underdetermined linear system $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$ with 8 equations in 9 unknowns corresponding to Linear System I. For any vector $\mathbf{y} \in \mathbb{R}^8$, the *linear inequalities* $\mathbf{y}^T M \geq \mathbf{0}^T$ *become*

$$
\begin{aligned}
y_2 \qquad + y_4 + y_5 + y_6 + y_7 \qquad &\geq 0 && (1)\\
y_1 + y_2 + y_3 \qquad + y_5 \qquad + y_7 \qquad &\geq 0 && (2)\\
y_1 + y_2 \qquad\qquad\qquad\qquad + y_8 &\geq 0 && (3)\\
y_2 + y_3 \qquad\qquad\qquad + y_8 &\geq 0 && (4)\\
y_4 + y_5 \qquad\quad + y_8 &\geq 0 && (5)\\
y_5 + y_6 \qquad + y_8 &\geq 0 && (6)\\
y_1 \qquad\quad + y_4 \qquad\qquad\quad + y_8 &\geq 0 && (7)\\
y_2 \qquad\quad + y_5 \qquad\quad + y_8 &\geq 0 && (8)\\
y_3 \qquad\quad + y_6 \qquad + y_8 &\geq 0 && (9)
\end{aligned}
$$

Adding inequalities (1)*,* (2)*,* (7)*, and* (9) *yields* $2\sum_{i=1}^{8} y_i \geq 0$*, which is a contradiction to the statement* $\mathbf{y}^T\widehat{\mathbf{b}} = \sum_{i=1}^{8} y_i < 0$*. Thus from Theorem 2.18 and Theorem 2.14 we conclude there exists a nonnegative rational solution to the system* $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$*. Indeed, it is easy to verify that* $\alpha_{1,1} = \alpha_{1,2} = \alpha_{2,5} = \alpha_{2,7} = 1/2$ *is a solution as equation* (2.4a) *of Problem I becomes*

$$\frac{1}{2}\begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}.$$

Example 4.4 (Unique rational solution of the relaxed system)**.** *Consider tiling*

$P =$  *with* $\mathfrak{S} = \{$  $\}$,

all rotations and reflections permitted. There is obviously no tiling. We show there exists a unique rational solution. After calculating all binary matrices, we have $m = c_R = 5$*,* $n = s_1 = 5$*,* $(n_s = 1)$*, which leads to the linear system of equations* $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$ *with* 5 *equations in* 5 *unknowns corresponding to Linear System II. For any vector* $\mathbf{y} \in \mathbb{R}^5$*, the linear inequalities* $\mathbf{y}^T M \geq \mathbf{0}^T$ *become*

$$
\begin{aligned}
y_1 \qquad + y_3 + y_4 \qquad &\geq 0\\
y_2 \qquad + y_4 + y_5 &\geq 0\\
y_1 + y_2 + y_3 \qquad\qquad &\geq 0\\
y_1 + y_2 \qquad + y_4 \qquad &\geq 0\\
y_2 + y_3 + y_4 \qquad &\geq 0
\end{aligned}
$$

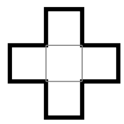
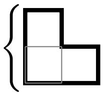
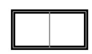
which together with $\mathbf{y}^T\widehat{\mathbf{b}} = \sum_{i=1}^{5} y_i < 0$ *is satisfied by* $\mathbf{y} = (1\ 1\ -2\ 1\ -2)^T$*. Thus from Theorem 2.14 and Theorem 2.18 it follows there does not exist a positive rational solution to* $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$*. However, it is easy to check that this system is consistent (*$\operatorname{rank}(M) = 5 = \operatorname{rank}([M|\widehat{\mathbf{b}}])$*). Indeed, Gaussian elimination yields the unique solution:*

$$\alpha_{1,1} = 2/3, \quad \alpha_{1,2} = 1, \quad \alpha_{1,3} = 2/3, \quad \alpha_{1,4} = -1/3, \quad \alpha_{1,5} = -1/3,$$

and equation (2.5) *becomes*

$$\frac{2}{3}\begin{pmatrix}1&0&0\\1&1&0\end{pmatrix}+\begin{pmatrix}0&1&0\\0&1&1\end{pmatrix}+\frac{2}{3}\begin{pmatrix}1&1&0\\1&0&0\end{pmatrix}-\frac{1}{3}\begin{pmatrix}1&1&0\\0&1&0\end{pmatrix}-\frac{1}{3}\begin{pmatrix}0&1&0\\1&1&0\end{pmatrix}$$
$$=\begin{pmatrix}1&1&0\\1&1&1\end{pmatrix}.$$

Example 4.5 (No solution of any kind)**.** *Let*

$P =$  *with* $\mathfrak{S} = \Big\{$  ,  $\Big\}$,

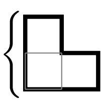
all rotations and reflections permitted. There is obviously no tiling. However, we have $c_1 = 3$*,* $c_2 = 2$*, and* $c_R = 5$*, so with* $d_1 = d_2 = 1$ *we have a nonnegative integer solution to* $c_1d_1 + c_2d_2 = c_R$*. After calculating the binary matrices, we have* $m = c_R + n_s = 7$*,* $n = s_1 = 8$*, which leads to the linear system of equations* $M\boldsymbol{\alpha} = \widehat{\mathbf{b}}$ *with* 7 *equations in* 8 *unknowns corresponding to Linear System I. After reducing the associated augmented matrix to reduced row echelon form we find*

$$\operatorname{rank}(M) = 5 < \operatorname{rank}([M|\widehat{\mathbf{b}}]) = 6,$$

thus the system of equations is inconsistent.

4.2. **Larger problems solved with optimization packages.** We present some medium to large ($200 < n < 70,000$) tiling examples that illustrate the range of problems that can be solved with the methodology outlined in Section 3.2.

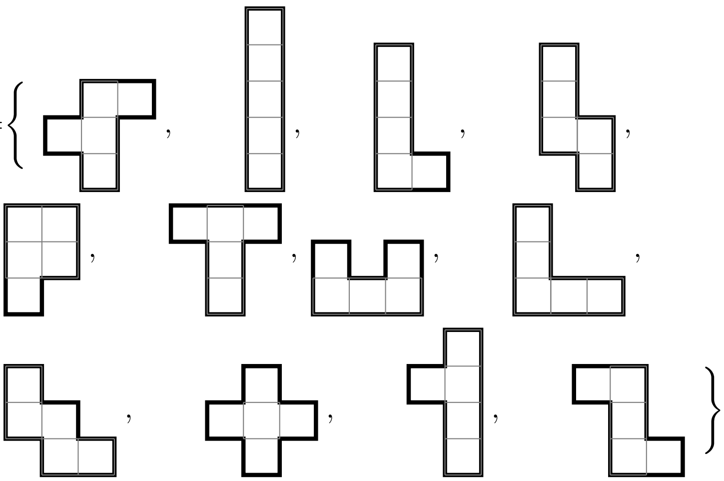
Example 4.6. *For our first example we investigated the problem of enumerating all possible solutions to tiling the* 5×18 *rectangle with* 30 *copies of a single L-triomino:*

$\mathfrak{S} = \Big\{$  $\Big\}$.

All rotations and reflections are permitted. The binary linear programming problem has 272 *variables with* 91 *constraints. A particular solution computed using CPLEX is shown in Figure 9a. CPLEX, GUROBI and SCIP successfully found* 1,168,512 *solutions in* 3.8 *minutes,* 2.8 *hours, and* 8.6 *minutes, respectively, which is in agreement with results reported at* [36]*. If we neglect trivial variations due to rotating and reflecting the whole board this yields a total of* $1,168,512/4 = 292,128$ *solutions.*

Example 4.7. *For our second example we enumerated the total number of solutions to tiling the* 10×6 *rectangle with each of the* 12 *free pentominoes (often labelled with the letters - F, I, L, N, P, T, U, V, W, X, Y, and Z.*

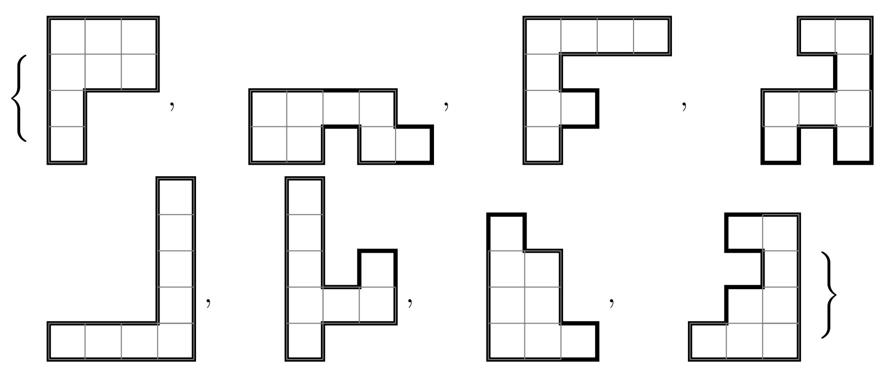
which are indicative of the polyomino shapes):

$\mathfrak{S} =$ 

All rotations and reflections are permitted. The binary linear programming problem has $2,056$ *variables with* 73 *constraints. This example is a classic problem of tiling with polyominoes. A particular solution computed using CPLEX is shown in Figure 9b. CPLEX, GUROBI and SCIP successfully found* $9,356$ *solutions in* 7.3 *minutes,* 77.4 *minutes, and* 89.9 *minutes, respectively, which is in agreement with the first reporting of this result in 1960* [40]*, and at numerous websites (e.g.* [36]*). If we neglect trivial variations due to rotating and reflecting the whole board this yields a total of* $9,356/4 = 2,339$ *solutions.*

The next two examples demonstrate the capability of our method to tile nonrectangular regions, or regions containing 'holes'.

Example 4.8. *We tile an L-shaped region with* 4 *copies of each of the following* 8 *octominoes*[5]*:*

$\mathfrak{S} =$ 

[5]There are 369 free octominoes [66].

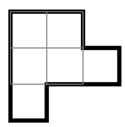
All rotations and reflections are permitted. The binary linear programming problem has 9,878 *variables with* 265 *constraints. CPLEX and SCIP successfully found a single (optimal) solution in* 12.8 *minutes and* 13.1 *hours, respectively, however GUROBI failed to find a solution in over* 14 *hours of computation. The solution shown in Figure 10a was computed in CPLEX.*

Example 4.9. *We tiled an* 11×11 *square with four* 2×2 *holes using the* 12 *free pentominoes (see Example 4.7) with copies*[6]. *All rotations and reflections are permitted. The binary linear programming problem has* 2,619 *variables with* 118 *constraints. CPLEX, GUROBI, and SCIP successfully found a single (optimal) solution in* 0.4, 1.5, *and* 15.9 *seconds, respectively. The solution shown in Figure 10b was computed in CPLEX.*

The last two examples illustrate the capability of our method to tile large problems, either with copies of a single polyomino, or with copies of a set of pieces.

Example 4.10. *We tile a* 60×60 *square with* 600 *copies of a single hexomino:*

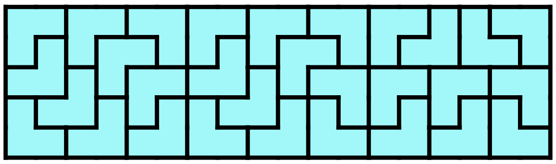
$$\mathfrak{S} = \Bigg\{ \quad \Bigg\}.$$



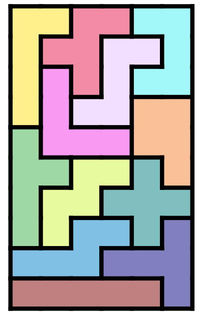
All rotations and reflections are permitted. The binary linear programming problem has 26,912 *variables with* 3,601 *constraints. CPLEX found a single (optimal) solution (see Figure 11) in* 1.9 *hours, however SCIP and GUROBI failed to find a solution in over* 14 *hours of computation.*

Example 4.11. *For our final example we tile the* 40×30 *rectangle using* 20 *copies each of the* 12 *free pentominoes (see Example 4.7). All rotations and reflections permitted. The binary linear programming problem has* 67,396 *variables with* 1,213 *constraints. CPLEX found a single (optimal) solution (see Figure 12) in* 9.5 *minutes, however SCIP and GUROBI failed to find a solution in over* 14 *hours of computation.*

[6]With the naming convention of Example 4.7 we used the following numbers of copies of the free pentominoes: V, W, and Z - a single copy each; all other pieces utilized two copies each.

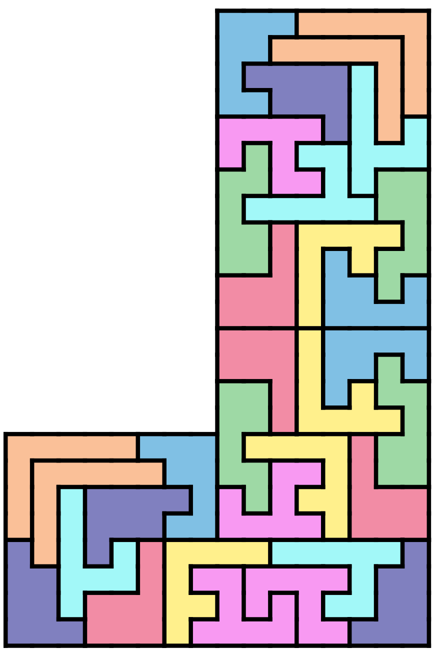


(a) Tiling the 5×18 rectangle.

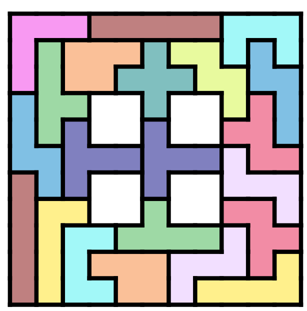


(b) Tiling the 10×6 rectangle.

FIGURE. 9. *Tiling rectangular regions. (a) See Example 4.6. We tile with* 30 *copies of a single L-triomino. (b) See Example 4.7. We tile with a full set of the* 12 *free pentominoes.*



(a) Tiling an L-shaped region.



(b) Tiling a square with 4 holes.

FIGURE. 10. *Two medium-sized tiling problems. (a) See Example 4.8. We tile with* 4 *copies of* 8 *octominoes. (b) See Example 4.9. We tile with copies of the* 12 *free pentominoes. See the text for further details concerning the tiles used.*

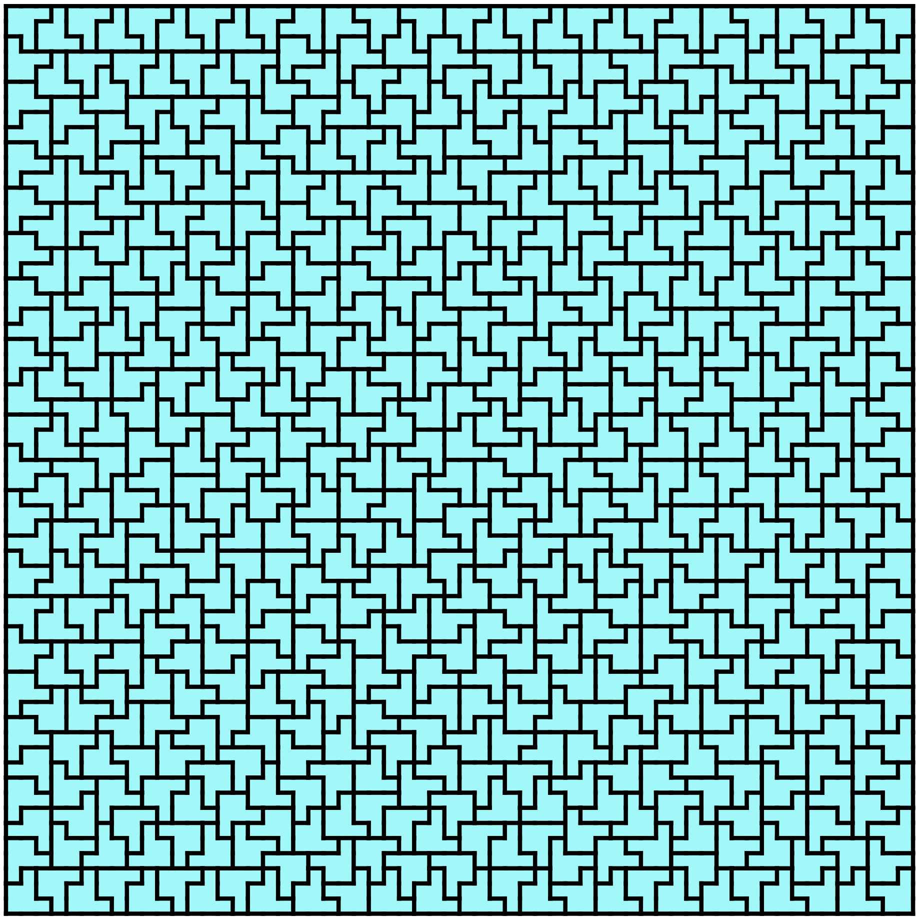


FIGURE. 11. *See Example 4.10.* 600 *copies of a single hexomino tile a* 60×60 *square.*

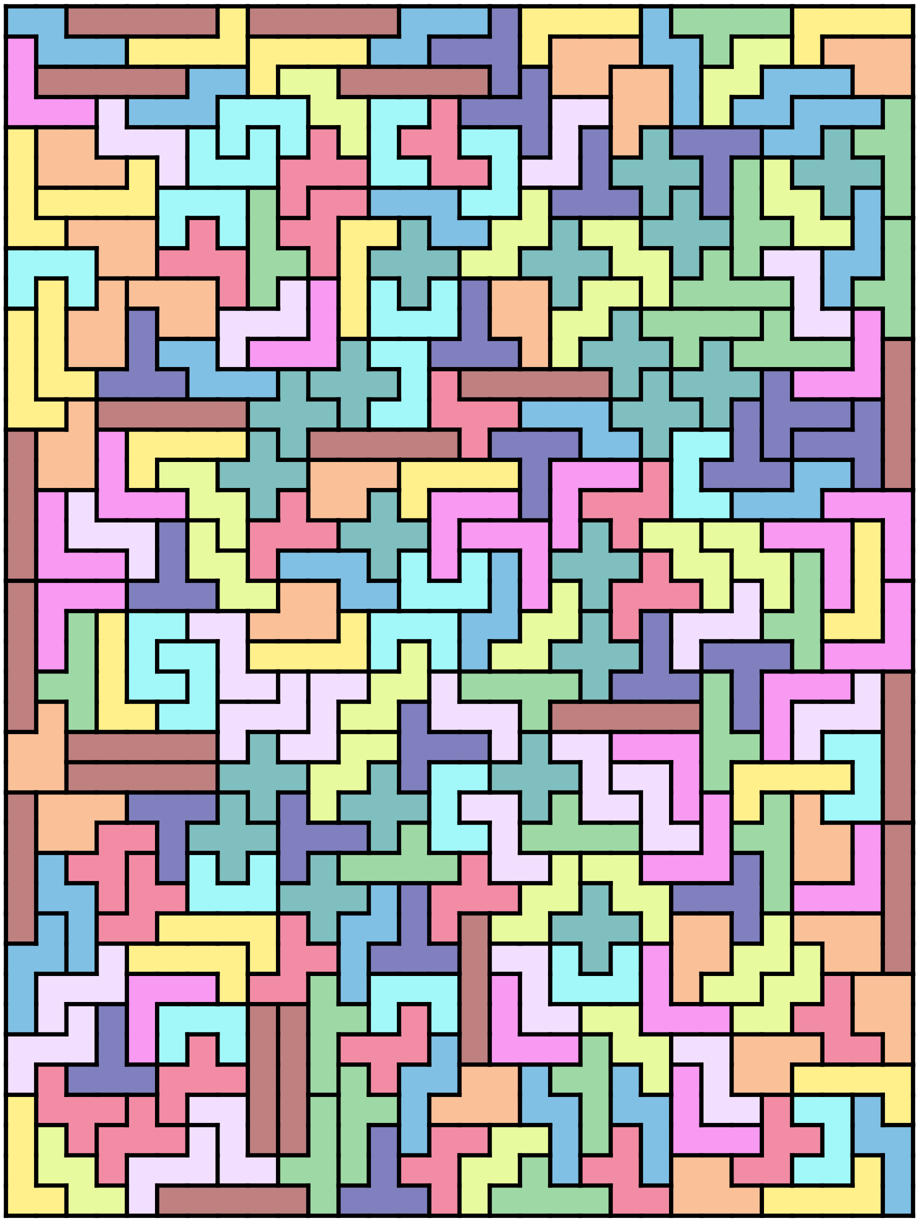


FIGURE. 12. *See Example 4.11. We tile the* 40×30 *rectangle with* 20 *copies of each of the* 12 *free pentominoes.*

Experiment 4.12. *We investigated the relationship between the time it takes for CPLEX to find an optimal solution to a tiling problem ('runtime') and the size of the problem (measured by the area of the region to be tiled). For simplicity and to avoid confounding results with different geometries we used copies of a single L-triomino (see Example 4.6) to tile a sequence of squares of increasing size:*

$$P := [45 + 3k] \times [45 + 3k], \quad k = 0, 1, \ldots, 29.$$

The runtimes (in seconds) versus the areas of the square regions tiled are shown in Figure 13 (dashed line with triangular markers). The exponential regression curve for the data is also plotted with a solid line and round markers, and has the approximate equation

$$runtime = 2.15e^{3.87\times 10^{-4}\cdot area},$$

with a correlation coefficient of $r = 0.865$. The runtimes double after each increase in area of about $1,791$ cells.

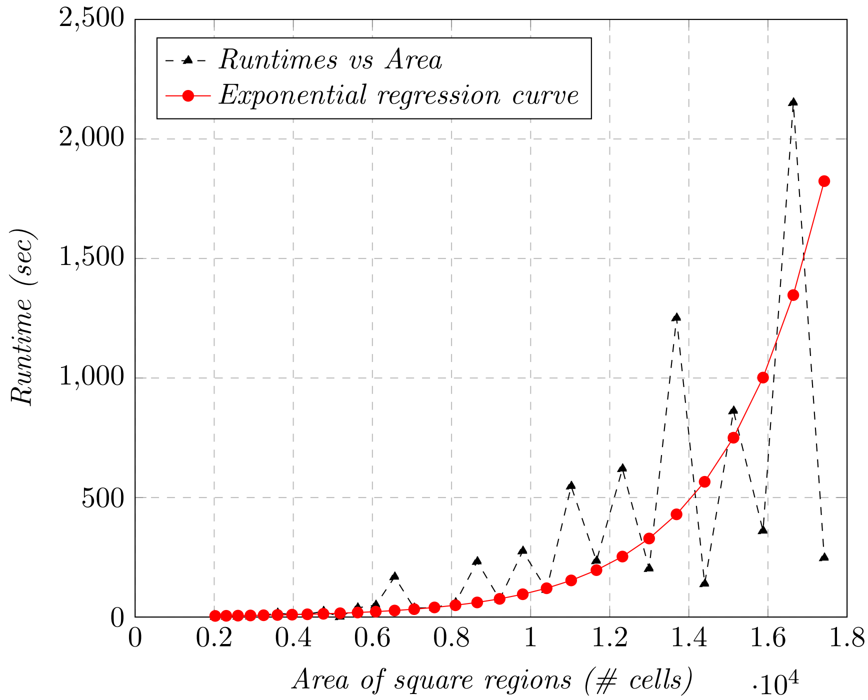


Figure. 13. *CPLEX runtimes versus area of square regions tiled with L-triominoes.*

The 'saw-tooth' shape of the data merits some discussion. The lower data points in the 'saw-tooth' pattern correspond to areas divisible by 6 while the upper data points correspond to areas divisible by 3, but not 6. As the L-triomino tiles a 6×6 square it is plausible that optimal solutions are found *more readily when the area is divisible by* 6*. If this were the case we might expect the optimal tiling of areas divisible by* 6 *to be predominantly filled with* 6×6 *blocks, where each block is tiled by* 12 *L-triominoes. We checked if this was the case for the* 66×66 *square and it was not (details omitted). Further investigations are needed to explain this behaviour.*

5. Conclusions

The main contribution of this paper is to present the first general systematic algebraic approach for tiling finite subsets of $\mathbb{Z}^2$ with polyominoes. The resulting mathematical model can be solved via direct solution techniques (see Algorithm 1), or expressed as a binary linear programming problem in MATLAB and solved using high-performance optimization packages, for example, CPLEX, SCIP, or GUROBI. To the best of our knowledge, all current general purpose algorithms for tiling regions of the plane with polyominoes employ backtracking techniques for exhaustively finding solutions with a computer. We illustrated our methodology for small problems ($n < 30$) in MATLAB, and for medium to large problems ($200 < n < 70,000$) we used a combination of MATLAB and high-performance optimization packages. We found the CPLEX optimizer to be much faster than either SCIP or GUROBI for solving the problems in this paper.

We make no claims regarding the efficiency of our methodology. Indeed, preliminary investigations using the optimized C++ backtracking software (POLYCUBE), freely available at [11], indicate this package to be considerably faster than our approach (details omitted). There are a couple of reasons that may explain this. Firstly, the backtracking algorithms for tiling remain entirely in the discrete realm, however, with our optimization approach we must sift through the rational solutions to find the binary solutions. Another reason that can make backtracking fast is that the code can be tailored to the specific problem being solved. For example, one can choose to place pieces in the target region first that are most constrained. Regardless of the method employed, as the problem of tiling finite regions of the plane with polyominoes is in general $\mathcal{NP}$-complete (see the discussion in Section 1), there will always be an upper bound on the size of the problems that can be solved. This was illustrated by the results of Experiment 4.12 in Section 4.2, which confirmed numerically that tiling finite regions of the plane with the L-triomino is $\mathcal{NP}$-complete [57]. With our approach that upper bound is determined in large part by the efficiency of the particular optimization package employed. There have been huge advances made in recent decades for solving mixed-integer and integer linear programming problems [48, 54]. As advances are made (in software and hardware) for solving integer linear programming problems, the efficiency of our methodology for tiling with polyominoes will also improve.

There are several advantages of the methodology employed in this paper for tiling with polyominoes compared to the backtracking methods. Firstly, as the tiling problem has been converted into an algebraic model, the structure, combinatorial nature, and solvability of the model can be analyzed (see for example Theorem 2.14, Theorem 2.18, and Theorem 3.1). For example, the standard condition involving rank can be efficiently applied to determine if the associated linear system of equations is consistent. In principle, it should also be possible to construct a solver that is tailored to the properties of the linear systems. It is the subject of a follow-up paper to investigate the efficiency of an optimized C++ version of Algorithm 1. A second advantage of our mathematical model is its flexibility. Ignoring memory and efficiency considerations, there are no limitations on: the geometry of the region to be tiled; the number of different pieces used; or the total number of pieces (including copies) that are used to tile a specific region. If the constraints in Problem I are neglected, then we do not have to specify beforehand how many copies of each polyomino are to be used[7]. A third advantage of our methodology is that the underlying code for constructing the mathematical model is freely available as a suite of MATLAB programs (see `http://people.sc.fsu.edu/~jburkardt/m_src/polyominoes/polyominoes.html`). Thus other researchers can reproduce our numerical results, or use it as a starting point for their own investigations.

The techniques developed in this paper may have more general applicability. As the problem of tiling regions of the plane with polyominoes is an example of an exact cover problem [45], it would be interesting to investigate if our methodology can be adapted to tackle other exact cover problems. For example, the n-queens problem [7], Sudoku [74], and edge-matching puzzles [50] are examples of mathematical puzzles and games that can be represented as exact cover problems. Applications that can be modelled as exact cover problems include Integrated Circuit Design [41], 3D Printing [79], and Cloud Computing [14].

There are some additional lines of enquiry we can pursue with our mathematical model. For example, it would be interesting to investigate least squares approximations of Linear System I and II. When the linear system is inconsistent, and cells in a tiling are allowed to overlap, the least squares solution corresponds to a weighted sum of pieces that minimizes the sum of the squared deviations. In this context, a deviation corresponds to the mismatch between the 'height' of a cell in the target region and unity. Thus even when a given set of polyominoes does not tile the target region, we can seek an approximate solution in the least squares sense. This is an approach that is not possible with backtracking methods as backtracking methods remain entirely in the discrete realm.

In addition to investigating the efficiency of an optimized C++ implementation of Algorithm 1 (see the comments above), it might be advantageous

[7]Although such an approach would lead to a much more computationally expensive optimization problem.

to apply a parallel computing approach to the direct solution method for solving our model. This is because the binary choices for each of the b_f sets of free variables can be tested independently (see Section 3.1), which would significantly reduce runtimes of a solver.

Finally, we mention that our methodology can be smoothly generalized to tiling problems in higher dimensional spaces $\mathbb{Z}^d$, $d \geq 3$, using polycubes [1]. An example for the case $d = 3$ is the 'somacube' puzzle, where seven pieces made from unit cubes are used to tile a $3 \times 3 \times 3$ cube. There are exactly 240 distinct configurations [63]. We leave this task, and other investigations, for future work.

Appendix A. Notation

For the convenience of the reader we summarize the main notation used in Table 1.

Table 1
Main notation used.

Symbol	Definition
R	The polyomino to be tiled
c_R	The order of R
$\mathfrak{B}$	Lattice for the rectangular $r \times c$ hull of R
$B \in \{0,1\}^{r \times c}$	Binary matrix associated with $\mathfrak{B}$
n_s	Number of series (= number of free polyominoes)
$\mathfrak{S}$	Set of free polyominoes P_i
P_i	ith free polyomino
c_i	Order of P_i
s_i	Number of ways each P_i fits in R (= number of binary matrices $A^{i,j}$ in Series i)
d_i	Number of copies of P_i used to tile R
$A^{i,j} \in \{0,1\}^{r \times c}$	Binary matrix associated with j ways of fitting P_i in R
n	Total number of binary matrices $A^{i,j}$ (= number of unknowns in linear systems)
$\alpha_{i,j} \in \{0,1\}$	Number of times the jth placement of P_i is used to tile R
n_p	Number of polyominoes used to tile R
m	Number of equations in Linear System I, or II
$M \in \{0,1\}^{m \times n}$	Coefficient matrix of Linear System I or II
$\boldsymbol{\alpha} \in \{0,1\}^n$	Solution vector for Linear System I or II
$\widehat{\mathbf{b}}$	Right-hand-side vector for Linear System I or II
N	Number of binary solutions of Linear System I or II
$[A\|\mathbf{b}]$	Reduced row echelon form of $[M\|\widehat{\mathbf{b}}]$
r	Rank of M
f	$n - r$ (= number of free variables in $[A\|\mathbf{b}]$)
f_i	Number of free variables in Series i
b_{f_i}	Number of binary choices for the free variables in Series i
b_f	Number of binary choices for all free variables

Appendix B. Shell commands for calculating all feasible solutions

B.1. **Shell commands for CPLEX.**

1. Set the absolute gap for the solution pool to zero:
   `CPLEX>  set mip pool absgap 0.0`
2. Aggressively seek all solutions:
   `CPLEX>  set mip pool intensity 4`
3. Set the upper bound on the number of solutions sought to N:
   `CPLEX>  set mip limits populate N`
4. Set the upper bound on the number of solutions stored to be N:
   `CPLEX>  set mip pool capacity N`
5. Specify that all solutions found will be written out to file:
   `CPLEX>  set output writelevel 1`
6. Read the LP file:
   `CPLEX>  read test.lp`
7. Calculate all solutions and store them:
   `CPLEX>  populate`
8. Write all solutions to a file:
   `CPLEX>  write test.sol all`

B.2. **Shell commands for SCIP.**

1. Read the LP file:
   `SCIP>  read test.lp`
2. Set some parameters needed for collecting all feasible solutions:
   `SCIP>  set emphasis counter`
   `SCIP>  set constraints countsols collect TRUE`
3. Calculate all feasible solutions:
   `SCIP>  count`
4. Write all solutions to file:
   `SCIP>   write allsolutions test.txt`

B.3. **Shell commands for GUROBI.** The following line of code: reads the LP file, sets the upper bound on the number of solutions sought to N, calculates all feasible solutions, and saves the solutions to file:

```
  >  gurobi_cl PoolSearchMode=2 PoolSolutions=N PoolGap=0
ResultFile=test.sol test.lp
```

Acknowledgments

We gratefully acknowledge the helpful comments of: Matthew Busche, Laurence Garvie, Steven Gismondi, Herb Kunze, Rajesh Pereira, and Allan Willms. We are also very grateful to an anonymous referee for their careful corrections and valuable suggestions that improved this paper.

References

1. G. Aleksandrowicz and G. Barequet, *Counting polycubes without the dimensionality curse*, Discrete Math. **309** (2009), 4576–4583.
2. F. Ardila and R. Stanley, *Tilings*, Math. Intelligencer **32** (2010), no. 4, 32–43.
3. J. M. Ash and S. W. Golomb, *Tiling deficient rectangles with trominoes*, Math. Mag. **77** (2003), no. 1, 46–55.
4. D. Ashlock and L. Taylor, *Evolving polyomino puzzles*, Proceedings of the IEEE 2016 Congress on Evolutionary Computation, IEEE Congress on Evolutionary Computation, IEEE, Vancouver, 2016, pp. 327–334.
5. B. Banerjee, L. Kraemer, and J. Lyle, *Multi-agent plan recognition: Formalization and algorithms*, Proceedings of the twenty-fourth AAAI conference on artificial intelligence (AAAI-10), Book Series: Lecture Notes in Artificial Intelligence, Conference: 24th AAAI Conference on Artificial Intelligence (AAAI), Atlanta, GA, 2010, pp. 1059–1064.
6. D. Battalion, M. Bouvel, A. Frosini, and S. Rinaldi, *Permutation classes and polyomino classes with excluded submatrices*, Math. Struct. in Comp. Science **27** (2017), 157–183.
7. J. Bell and B. Stevens, *A survey of known results and research areas for n-queens*, Discrete Math. **309** (2009), no. 1, 1–31.
8. R. Berger, *The undecidability of the domino problem*, Mem. Amer. Math. Soc. **66** (1966), 1–72.
9. O. Bodini, *Tiling a rectangle with polyominoes*, Discrete Math. Theor. Comput. Sci. AB(DMCS) (2003), 81–88.
10. M. Braun, T. Etzion, P. R. J. Östergård, A. Vardy, and A. Wassermann, *Existence of q-analogs of steiner systems*, Forum Math. Pi **e7, 14 pp.** (2016).
11. M. Busche, *Solving polyomino and polycube puzzles. algorithms, software, and solutions*, `http://www.mattbusche.org/blog/article/polycube/` (2 July 2020).
12. G. Castiglione, A. Frosini, A. Restivo, and S. Rinaldi, *Enumeration of L-convex polyominoes by rows and columns*, Theoret. Comput. Sci. **347** (2005), 336–352.
13. R. D. Chatham, M. Doyle, J. J. Miller, A. M. Rogers, R. D. Skaggs, and J. A. Ward, *Algorithm performance for chessboard separation problems*, J. Combin. Math. Combin. Comput. **70** (2009), 127–142.
14. B. A. Cheikh, *An exact cover-based approach for service composition*, 2016 IEEE International Conference on Web Services (ICWS), IEEE Computer Society, 2006, pp. 631–636.
15. J. L. Conway and J. C. Lagarias, *Tiling with polyominoes and combinatorial group theory*, J. Combin. Theory Ser. A **53** (1990), no. 2, 183–208.
16. M. Cwiek and J. Nalepa, *Spatial planning as a hexomino puzzle*, Intelligent Information and Database Systems, ACIIDS 2017, Pt 1, Book Series: Lecture Notes in Artificial Intelligence, vol. 10191, Conference: 9th Asian Conference on Intelligent Information and Database Systems (ACIIDS), Kanazawa, JAPAN, 2017, pp. 410–420.
17. N. G. de Bruijn, *Programmeren van de pentomino puzzle*, Euclides **47** (1971/72), 90–104.
18. A. Del Lungo, E. Duchi, A. Frosini, and S. Rinaldi, *On the generation and enumeration of some classes of convex polyominoes*, Electron. J. Combin. **11** (2004), no. 1, 1–46.
19. M.-P. Delest and G. Viennot, *Algebraic languages and polyominoes enumeration*, Theoret. Comput. Sci. **34** (1984), no. 1-2, 169–206.
20. J. Farkas, *Über die Theorie der einfachen Ungleichungen*, Journal für riene und angewandte Mathematik **121** (1902), 1–27.
21. S. Feretić, *A perimeter enumeration of column-convex polyominoes*, Discrete Math. Theor. Comput. Sci. **9** (2007), no. 1, 57–83.

22. J. G. Fletcher, *A program to solve the pentomino problem by the recursive use of macros*, Communications of the ACM **8** (1965), no. 10, 621–623.
23. A. Fontaine and G. E. Martin, *Polymorphic polyominoes*, Math. Mag. **57** (1984), no. 5, 275–283.
24. D. Gale, H. W. Kuhn, and A. W. Tucker, *Linear programming and the theory of games*, Activity Analysis of Production and Allocation (T.C. Koopmans, ed.), John Wiley & Sons, New York, 1951, pp. 317–329.
25. M. R. Garey and D. S. Johnson, *Computers and intractability: A guide to the theory of NP-Completeness*, W. H. Freeman and Company, San Francisco, 1979.
26. S. W. Golomb, *Checker boards and polyominoes*, Amer. Math. Monthly **61** (1954), no. 10, 675–682.
27. ______, *Tiling with polyominoes*, J. Combinatorial Theory **1** (1966), 280–296.
28. ______, *Polyominoes which tile rectangles*, J. Combin. Theory Ser. A **51** (1987), 117–124.
29. ______, *Tiling rectangles with polyominoes*, Math. Intelligencer **18** (1996), no. 2, 38–47.
30. S. W. Golomb and L. D. Baumert, *Backtrack programming*, J. Ass. Comp. Machinery **12** (1965), no. 4, 516–524.
31. S. W. Golomb and D. A. Klarner, *Polyominoes*, Handbook of discrete and computational geometry (J.E. Goodman and J. O'Rourke, eds.), Chapman & Hall / CRC, Atlanta, GA, 2nd ed., 2004, pp. 331–352.
32. S.W. Golomb, *Polyominoes*, Scribner, New York, 1965.
33. ______, *Tiling with sets of polyominoes*, J. Combinatorial Theory **9** (1970), 60–71.
34. ______, *Polyominoes*, second ed., Princeton University Press, Princeton, NJ, 1994.
35. A. Goupil, H. Cloutier, and F. Nouboud, *Enumeration of polyominoes inscribed in a rectangle*, Discrete Appl. Math. **158** (2010), no. 18, 2014–2023.
36. D. O. Grekov, *Polyomino tilings*, `http://polyominoes.org/data/` (3 August 2018).
37. B. Grünbaum and G.C. Shephard, *Tilings and patterns*, W. H. Freeman and Company, New York, 1987.
38. V. Gruslys, I. Leader, and T. S. Tan, *Tiling with arbitrary tiles*, Proc. London Math. Soc. **3** (2016), no. 112, 1019–1039.
39. A. J. Guttman, *History and introduction to polygon models and polyominoes*, Polygons, Polyominoes and Polycubes (A.J. Guttman, ed.), Lecture Notes in Physics, vol. 775, Springer, Dordrecht, 2009.
40. C. B. Haselgrove and J. Haselgrove, *A computer program for pentominoes*, Eureka **23** (1960), no. 2, 16–18.
41. I. Hui-Ru and H.-Y. Chang, *Multiple patterning layout decomposition considering complex coloring rules and density balancing*, IEEE transactions on computer-aided design of integrated circuits and systems **36** (2017), no. 12, 2080–2092.
42. D. A. Klarner, *Packing a rectangle with congruent n-ominoes*, J. Combinatorial Theory **7** (1969), 107–115.
43. D. A. Klarner and R. L. Rivest, *A procedure for improving the upper bound for the number of n-ominoes*, Can. J. Math. **25** (1973), no. 3, 585–602.
44. ______, *Asymptotic bounds for the number of convex n-ominoes*, Can. J. Math. **25** (1974), no. 8, 31–40.
45. D. Knuth, *Dancing links*, Millennial Perspectives in Computer Science, proceedings of the 1999 Oxford-Microsoft Symposium in honour of Professor Sir Antony Hoare (J. Davies, B. Roscoe, and J. Woodcock, eds.), Cornerstones of Computing, Palgrave Macmillan, 2000, p. 432.
46. ______, *Selected papers on fun & games*, CSLI Lecture Notes; no. 192, CSLI Publications, Stanford, CA, 2011.

47. ———, *The art of computer programming, fascicle 5: Mathematical preliminaries redux; introduction to backtracking; dancing links*, vol. 4, Addison-Wesley, Boston, 2020.
48. T. Koch, T. Achterberg, E. Andersen, O. Bastert, T. Berthold, and R. E. Bixby, *Mixed integer programming library version 5*, Math. Prog. Comp. **3** (2011), 103–163.
49. M. Korn and I. Pak, *Tilings of rectangles with T-tetrominoes*, Theoret. Comput. Sci. **319** (2004), no. 1-3, 3–27.
50. S. Z. Kovalsky, D. Glasner, and R. Basri, *A global approach for solving edge-matching puzzles*, SIAM J. Imaging Sciences **8** (2015), no. 2, 916–938.
51. P. Leroux, E. Rassart, and A. Robitaille, *Enumeration of symmetry classes of convex polyominoes in the square lattice*, Adv. in Appl. Math. **21** (1998), no. 3, 343–380.
52. D. K. Lilly, *Solving general lattice puzzles*, Frontiers in Algorithmics, Book Series: Lecture Notes in Computer Science, vol. 6213, Conference: 4th International Frontiers of Algorithmics, Wuhan Univ, Wuhan, China, 2010, pp. 124 –.
53. D.K. Lilly, *Complexity of solvable cases of the decision problem for predicate calculus*, 19th Annual Symposium on Foundations of Computer Science, IEEE, Long Beach, Calif., 1978, pp. 35–47.
54. A. Lodi, *Mixed integer programming computation*, 50 Years of Integer Programming 1958-2008 (M. Jünger, T.M. Liebling, D. Naddef, G.L. Nemhauser, W.R. Pulleyblank, G. Reinelt, G. Rinaldi, and L.A. Wolsey, eds.), Springer-Verlag, Berlin, 2010, pp. 410–420.
55. W. R. Marshall, *Packing rectangles with congruent polyominoes*, J. Combin. Theory Ser. A **77** (1997), 181–192.
56. B.-M. Mireille, *Codage des polyominos convexes et équations pour l'énumération suivant l'aire*, Discrete Appl. Math. **48** (1994), no. 1, 21–43.
57. C. Moore and J. M. Robson, *Hard tiling problems with simple tiles*, Discrete Comput. Geom. **26** (2001), 573–590.
58. D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, *Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning*, Discrete Optim. **19** (2016), 79–102.
59. G. L. Nemhauser and L. A. Wolsey, *Integer and combinatorial optimization*, John Wiley & Sons, New York, 1988.
60. I. Pak, *Ribbon tile invariants*, Trans. Amer. Math. **352** (2000), no. 2, 5525–5561.
61. ———, *Tile invariants: new horizons*, Theoret. Comput. Sci. **303** (2003), 303–331.
62. C. H. Papadimitriou and K. Stieglitz, *Combinatorial optimization: algorithms and complexity*, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1982.
63. C. Peter-Orth, *All solutions of the soma cube puzzle*, Discrete Math. **57** (1985), no. 1-2, 105–121.
64. J. L. Ramírez Alfonsin, *The Diophantine Frobenius problem*, Oxford Lecture Series in Mathematics and its Applications, vol. 30, Oxford University Press, Oxford, 2005.
65. D. A. Rawsthorne, *Tiling complexity of small n-ominoes ($n < 10$)*, Discrete Math. **70** (1988), no. 1, 71–75.
66. D. H. Redelmeier, *Counting polyominoes: yet another attack*, Discrete Math. **36** (1981), no. 2, 191–203.
67. M. Reid, *Tiling rectangles and half strips with congruent polyominoes*, J. Combin. Theory Ser. A **80** (1997), no. 1, 106–123.
68. ———, *The homotopy groups*, L' Enseignement Mathématique **49** (2003), 123–155.
69. ———, *Klarner systems and tiling boxes with polyominoes*, J. Combin. Theory Ser. A **111** (2005), 89–105.
70. ———, *Asymptotically optimal box packing theorems*, Electron. J. Combin. **15** (2008), no. 1, 1–19.
71. ———, *Many L-shaped polyominoes have odd rectangular packings*, Ann. Comb. **18** (2014), 341–357.

72. G. C. Rhoads, *Planar tilings by polyominoes, polyhexes, and polyiamonds*, J. Comput. Appl. Math. **174** (2005), 329–353.
73. R. M. Robinson, *Undecidability and nonperiodicity of tilings of the plane*, Invent. Math. **12** (1971), 177–209.
74. J. Rosenhouse and L. Taalman, *Taking Sudoku seriously. the math behind the world's most popular pencil puzzle*, Oxford University Press, Oxford, 2011.
75. D. Schattschneider, *Will it tile? Try the conway criterion!*, Math. Mag. **53** (1980), no. 4, 224–233.
76. W. P. Thurston, *Conway's tiling groups*, Amer. Math. Monthly **97** (1990), no. 8, 757–773.
77. S. Wagon, *Fourteen proofs of a result about tiling a rectangle*, Amer. Math. Monthly **94** (1987), no. 7, 601–617.
78. A. Winslow, *An optimal algorithm for tiling the plane with a translated polyomino*, Algorithms and Computation: 26th International Symposium, ISAAC 2015 Nagoya, Japan, December 9-11, 2015 Proceedings (K. Elbassioni and K. Makino, eds.), Lecture Notes in Comput. Sci., vol. 9472, Springer, Heidelberg, 2015, pp. 3–13.
79. W. Xiao-Ran, Z. Yu-He, and G. Guo-Hua, *No-infill 3D printing*, 3D Research **7** (2016), no. 3, 7–24.

Department of Mathematics & Statistics, University of Guelph, ON Canada N1G 2W1
E-mail address: `mgarvie@uoguelph.ca`

Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260
E-mail address: `jvb25@pitt.edu`