algorithms

MDPI

Article

A Parallelizable Integer Linear Programming Approach for Tiling Finite Regions of the Plane with Polyominoes

Marcus R. Garvie [1,*] and John Burkardt [2]

[1] Department of Mathematics & Statistics, University of Guelph, Guelph, ON N1G 2W1, Canada
[2] Department of Mathematics, University of Pittsburgh, Pittsburgh, PA 15260, USA; jvb25@pitt.edu
* Correspondence: mgarvie@uoguelph.ca

Abstract: The general problem of tiling finite regions of the plane with polyominoes is $\mathcal{NP}$-complete, and so the associated computational geometry problem rapidly becomes intractable for large instances. Thus, the need to reduce algorithm complexity for tiling is important and continues as a fruitful area of research. Traditional approaches to tiling with polyominoes use backtracking, which is a refinement of the 'brute-force' solution procedure for exhaustively finding all solutions to a combinatorial search problem. In this work, we combine checkerboard colouring techniques with a recently introduced integer linear programming (ILP) technique for tiling with polyominoes. The colouring arguments often split large tiling problems into smaller subproblems, each represented as a separate ILP problem. Problems that are amenable to this approach are embarrassingly parallel, and our work provides proof of concept of a parallelizable algorithm. The main goal is to analyze when this approach yields a potential parallel speedup. The novel colouring technique shows excellent promise in yielding a parallel speedup for finding large tiling solutions with ILP, particularly when we seek a single (optimal) solution. We also classify the tiling problems that result from applying our colouring technique according to different criteria and compute representative examples using a combination of `MATLAB` and `CPLEX`, a commercial optimization package that can solve ILP problems. The collections of `MATLAB` programs `PARIOMINOES` (v3.0.0) and `POLYOMINOES` (v2.1.4) used to construct the ILP problems are freely available for download.

Keywords: tiling with polyominoes; checkerboard colouring arguments; integer linear programming; parallelizable algorithm; algorithm complexity; `MATLAB`; `CPLEX`

check for updates

Citation: Garvie, M.R.; Burkardt, J. A Parallelizable Integer Linear Programming Approach for Tiling Finite Regions of the Plane with Polyominoes. *Algorithms* **2022**, *15*, 164. https://dx.doi.org/10.3390/a15050164

Academic Editor: Yuri N. Sotskov

Received: 25 March 2022
Accepted: 7 May 2022
Published: 12 May 2022

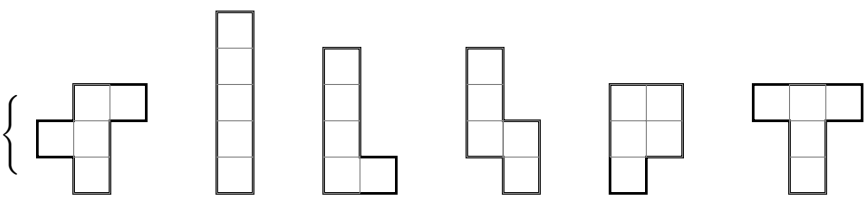
Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

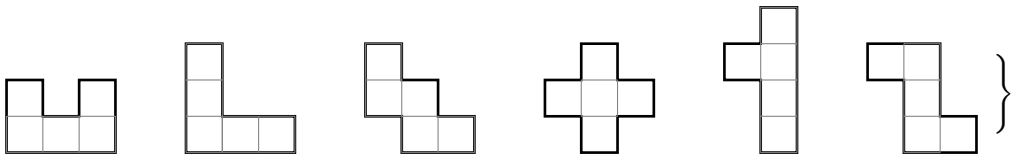
Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Background and Motivation

A polyomino P_i is a set of edge-connected unit squares in the plane, which we assume is simply connected. We refer to the polyominoes of area n as n-ominoes and for $n = 1, 2, \ldots, 8$ are called monominoes, dominoes, triminoes, tetrominoes, pentominoes, hexominoes, heptominoes, and octominoes, respectively. In this work, we focus on tiling finite regions of the plane R with copies of F *free* polyominoes $\{P_i\}_{i=1}^{F}$. Free polyominoes are the same if reflected ('flipped') or rotated, and thus correspond to a physical puzzle piece or tile. For example, there are exactly 12 free pentominoes, illustrated below:

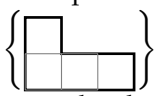




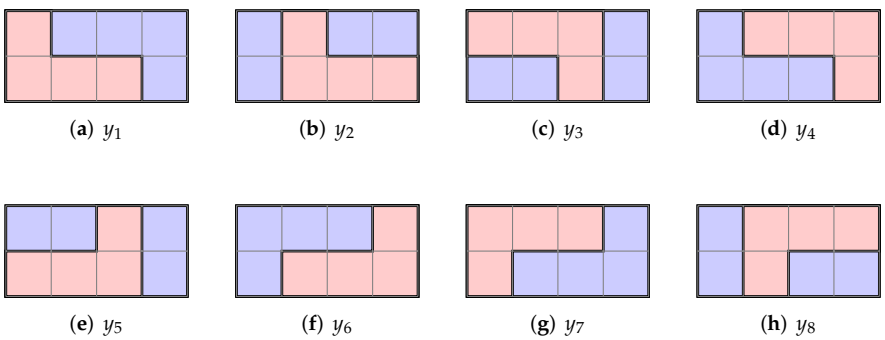
We can rotate *one-sided* polyominoes, but not reflect them, while *fixed* polyominoes cannot be rotated or reflected. There is no known closed-form formula for enumerating the number of distinct polyominoes (free, one-sided, or fixed) as a function of area or perimeter [1–12]. For tiling reasons, we can translate free, one-sided, and fixed polyominoes in a *target region* R. We assume the target region is connected, but not necessarily simply connected, i.e., we let R have 'holes'. There are two basic tiling situations: tiling with copies of a single free polyomino, or tiling with two or more distinct free polyominoes (with or without copies). We refer to these cases as *monohedral* and *multihedral*, respectively. For more background theory on polyominoes and tiling with polyominoes, we refer the reader to standard references (see e.g., [5,7,13–16] and the citations there). There is also a large specialized literature on the many computational and theoretical aspects of tiling the plane with polyominoes [17–23], or tiling finite regions of the plane with polyominoes (e.g., [24–40]). Unless stated otherwise, in the rest of this article, we assume polyominoes are free.

Tiling with polyominoes is an example of combinatorial optimization [41,42]. The key challenge is to develop algorithms that solve large tiling problems in a reasonable amount of time [43]. The general problem of tiling finite regions of the plane with polyominoes is $\mathcal{NP}$-complete [42,44,45], and so the associated computational geometry problem rapidly becomes intractable for large instances. Thus, it is important to reduce algorithm complexity for tiling, and this area continues as a fruitful area of research.

The method in this article for tiling relies on combining two different techniques, namely, integer linear programming (ILP) [46] and checkerboard colouring techniques [47]. We give a very simple tiling example to motivate our tiling strategy and leave the formal development of the method and definitions to Section 3. Although the two solutions are obvious, for the sake of introducing the colouring approach, we proceed as though the solution is unknown, to highlight the issues involved.

Consider the problem of tiling the 2×4 rectangle, denoted R, with two L-shaped tetrominoes  , all orientations permitted. Initially, we review the basic ILP approach that was developed in [46] and then solve the same problem with the addition of colouring techniques.

We introduce a variable $y_i \in \{0, 1\}$ for whether we use a particular placement of a tetromino (coloured red) to tile the region, illustrated in Figure 1.



(a) y_1 **(b)** y_2 **(c)** y_3 **(d)** y_4

(e) y_5 **(f)** y_6 **(g)** y_7 **(h)** y_8

Figure 1. Possible placements of a tetromino in R and the associated variables.

In any tiling of R, we must cover each of the eight cells exactly once, yielding the following system of eight linear equations in eight unknowns:

$$\begin{array}{rrrrrrrrr} y_1 & & + y_3 & & & & + y_7 & & = 1 \\ & y_2 & + y_3 & + y_4 & & & + y_7 & + y_8 & = 1 \\ & & y_3 & + y_4 & + y_5 & & + y_7 & + y_8 & = 1 \\ & & & y_4 & & + y_6 & & + y_8 & = 1 \\ y_1 & & & & + y_5 & & + y_7 & & = 1 \\ y_1 & + y_2 & & & + y_5 & + y_6 & & + y_8 & = 1 \\ y_1 & + y_2 & + y_3 & & + y_5 & + y_6 & & & = 1 \\ & y_2 & & + y_4 & & + y_6 & & & = 1 \end{array} \tag{1}$$

Observe that if we sum these equations, we obtain $4\sum_{i=1}^{8} y_i = 8 \implies \sum_{i=1}^{8} y_i = 2$, which implies that we must use exactly two polyominoes to tile the region. Thus, this constraint is automatically incorporated into the system. A tiling of the region R corresponds to a binary solution of this system. However, solutions of the extended system where $y_i \in \mathbb{R}$ do not necessarily correspond to a tiling as solutions may also be rational. The reduced row echelon form of this system has seven non-zero rows and eight variables, thus one free variable. Solving this system, using a high-performance optimization package, such as `CPLEX`, `Gurobi`, or `SCIP`, yields two binary solutions: $y_1 = y_4 = 1$ with all other variables equal to zero; and $y_7 = y_6 = 1$ with all other variables equal to zero, illustrated in Figure 2. These tilings are trivial variations of each other, obtained by reflecting the entire board horizontally.

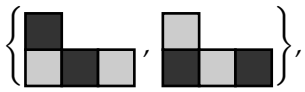
(**a**) (**b**)

Figure 2. The two possible ways to tile R: (**a**) first tiling solution; (**b**) second tiling solution.

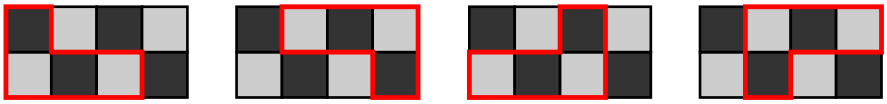
We now solve this tiling problem by including checkerboard colouring techniques. The starting point of the approach is to give the target region R a fixed checkerboard colouring, illustrated in Figure 3. There is another checkerboard colouring for this region obtained by swapping the black and white squares, but the particular choice of colouring does not matter to the solution procedure.

Figure 3. Checkerboard coloured region to be tiled with coloured tetrominoes.

We also assume that the tetrominoes used to tile R have a checkerboard colouring, and the two distinct variants are



where all orientations are permitted. Not only must the coloured polyominoes fit in the target region, but the colouring of the cells covered must correspond to the colouring of the cells in the tiles. Using two coloured tetrominoes from this set yields three subcases: tiling with two of the first coloured tetromino; tiling with one coloured tetromino of each variant; or tiling with two of the second coloured tetromino. We focus on the first subcase. We introduce a variable $x_i \in \{0, 1\}$ for whether a particular placement of a coloured tetromino of the first kind in this set is used to tile the region, illustrated in Figure 4.



(**a**) x_1 (**b**) x_2 (**c**) x_3 (**d**) x_4

Figure 4. Possible placements of a coloured tetromino in R and the associated variables. The tetrominoes are outlined in red.

In any tiling of the checkerboard coloured region in Figure 3, each of the eight cells must be covered exactly once, yielding the following system of eight linear equations in four unknowns:

$$\begin{aligned} x_1 &= 1 \\ x_2 + x_4 &= 1 \\ x_2 + x_3 + x_4 &= 1 \\ x_2 + x_4 &= 1 \\ x_1 + x_3 &= 1 \\ x_1 + x_3 + x_4 &= 1 \\ x_1 + x_3 &= 1 \\ x_2 &= 1 \end{aligned} \tag{2}$$

If we sum these equations, we obtain $4\sum_{i=1}^{4} x_i = 8 \implies \sum_{i=1}^{4} x_i = 2$, which again reflects the fact that we must use exactly two coloured polyominoes of the first variant to tile the region. Thus, this constraint is also automatically incorporated into the system. However, unlike in the previous example, this system has full rank and is thus trivially solved to yield $x_1 = x_2 = 1$ with all other variables equal to zero, i.e., we have the first tiling solution illustrated in Figure 2. The third subcase is similar, yielding the second tiling solution illustrated in Figure 2.

In the second subcase, unlike the first and third subcases, we are tiling with two different coloured polyominoes. So each coloured polyomino has an associated constraint equation that must be incorporated into the linear system (i.e., unlike the case with a single polyomino or coloured polyomino, they are not both automatically incorporated into the linear system). This is like the multihedral case for tiling with polyominoes [46], which we review in the next section. The system of equations arising from the second subcase also has full rank; however, the unique solution of the extended system with $x_i \in \mathbb{R}$ is non-binary, and thus does not correspond to a tiling solution. This tiling example, although very simple, highlights an important point. When we combine checkerboard colouring techniques with the ILP approach for tiling, the problem can split into several subproblems with a reduction in subproblem complexity. Each subproblem is independently solvable, and the solutions to the full problem were partitioned among the subproblems.

The problem we introduced above is very simple, and not only because it is a small monohedral problem. The tetromino we used to tile the region is what we call 'balanced', i.e., when we apply a checkerboard colouring to this tile, the number of black cells is equal to the number of white cells. Thus, when tiling a region with N copies of a balanced tile, there will always be $N + 1$ subcases to consider when seeking all tiling solutions: r tiles of the second coloured variant with $N - r$ tiles of the first coloured variant, where $r = 0, 1, \ldots, N$. However, we often tile with checkerboard coloured polyominoes that are not balanced, in which case the splitting of the full problem into subproblems depends on the 'parity' of the tiles and the target region, where we define parity as the number of black squares minus the number of white squares of a tile or region (see the next section and [47]).

1.2. Goals and Related Work

The traditional approach to tiling finite regions of the plane with polyominoes employs backtracking, which is the default way in computer science for exploring the search tree of a combinatorial problem [48–52]. Although backtracking is quick for some problems, it only refines a 'brute-force' solution procedure for exhaustively finding all solutions to a combinatorial search problem. Another less commonly used 'brute-force' approach to tiling with polyominoes employs either evolutionary computation [26] or genetic algorithms [53]; however, such approaches are likely considerably less efficient than backtracking. The only other general-purpose algorithmic procedure to tiling finite regions of the plane with polyominoes that we are aware of uses ILP, first introduced in [46]. There are several potential advantages of an algebraic approach to tiling over the backtracking methods. In [46], the authors note that with ILP the "the structure, combinatorial nature, and solvability of the model can be analyzed". Finally, we mention there are many pure mathematical results for proving that a set of polyominoes tiles a region, for example, using the combinatorial group theory approach of J.H. Conway [54–56]. However, these methods typically apply only to special cases and thus, we cannot make them algorithmic.

It is interesting to note that the tiling problem, which is here regarded as an ILP instance, can also be considered a satisfiability problem (SAT) [57–59], for which there are a number of powerful solvers. Examples of suitable open-source SAT solvers include `Lingeling`; see http://fmv.jku.at/lingeling/ (accessed on 6 May 2022) and `MapleSAT`, see https://sites.google.com/a/gsd.uwaterloo.ca/maplesat/maplesat (accessed on 6 May 2022). Unfortunately, however, the overhead of enforcing the conditions that we use each tile a fixed number of times is prohibitive. Further details are provided in Appendix A.

The focus of the current article is to combine checkerboard colouring techniques adapted from [47] with a recently introduced ILP method [46] for tiling with polyominoes. The checkerboard colouring method [47] was originally used to identify large impossible tiling problems, i.e., the opposite of what we aim to do here. Our checkerboard colouring techniques often splits large tiling problems into smaller tiling subproblems, where each subproblem is represented as a separate ILP problem. Problems that are amenable to this approach are embarrassingly parallel. This article provides proof of concept of a parallelizable ILP approach for tiling finite regions of the plane with polyominoes. We construct the ILP formulations of the tiling problems in `MATLAB` and compute the numerical solutions using `CPLEX`, a high-performance optimization package. The primary goal is to analyze when this approach yields a potential parallel speedup.

For tiling problems solved via a parallelizable ILP optimization technique, there are two basic aims: (i) find *a single* tiling solution (i.e., an optimal solution), and (ii) find *all* tiling solutions. In the former case, when we apply our checkerboard colouring techniques, we seek the subcase yielding an optimal solution computed in the least amount of time. In the latter case, it is the subcase that takes the longest time to compute all solutions that is of interest; if this takes less time to compute than for the full (uncoloured) problem, then we have a potential parallel speedup.

The simple tiling examples discussed above raise many questions, namely the following:

1. When do the checkerboard colouring techniques split the problem into multiple ($\geq$2) independently solvable subproblems?
2. For large problems (a large target region and/or the use of many tiles), what is an appropriate measure of problem complexity and the work done to compute tiling solutions?
3. Under what situations does this 'splitting' technique yield a potential parallel speedup with parallel computing?

We structure the remaining parts of this article as follows. After giving some preliminary details about checkerboard colouring in Section 2, we describe how to build the ILP model formulation for tiling in Section 3. In Section 4, we describe how large tiling problems often split into many smaller tiling subproblems with the application of checkerboard

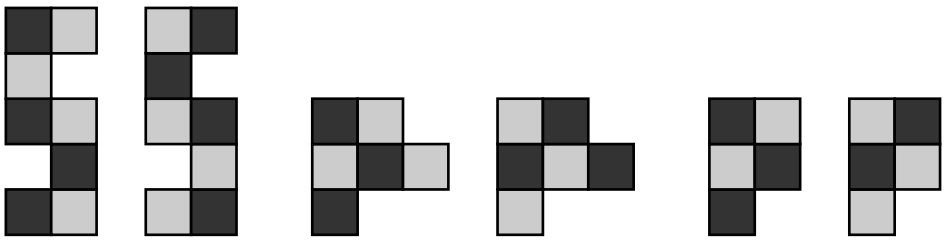
colouring techniques. Section 5 deals with the theoretical issues of complexity, measures of performance, and problem classification. Section 6 presents the numerical solution of medium to large tiling problems, and in Section 7 we make some general comments about algorithm performance. Concluding remarks are in Section 8. Finally, in Appendix A, we discuss the possible application of an SAT solver to our tiling problem.

2. Preliminaries

To begin our colouring approach to tiling, we give the target region R a *checkerboard* colouring, i.e., each black cell is only edge connected to white cells (and vice versa). We colour 'white' cells gray in all the figures of this article so that target regions with 'holes' stand out as white. We rigorously define a checkerboard colouring with the formula $f(i,j) = i + j \pmod 2$ applied to position (i,j) of each cell in the matrix representation of the region [60], where we identify the number $+1$ with black cells and the number 0 with white cells. To reverse the colouring, we instead apply the formula $f(i,j) = i + j + 1 \pmod 2$. Polyominoes placed in a checkerboard coloured region acquire the colouring of the cells they cover. Changing the placement of a polyomino in R can reverse the colouring of a polyomino, i.e., black and white cells reverse.

The *parity* of a checkerboard coloured polyomino or region is the number of black cells minus the number of white cells [47]. When the number of black cells equals the number of white cells, the polyomino or region is *balanced* [31] (p. 17) and the parity is zero. If the parity of a polyomino P_i or region R is non-zero, then there are two possible parity values denoted $\pm p_i$ or $\pm p$ respectively, where p_i and p are positive integers. If the parity of a polyomino P_i or region R is zero, then $p_i = 0$ or $p = 0$, respectively.

It is convenient to define a *pariomino* (pl. *pariominoes*) to be a particular checkerboard coloured polyomino. As the polyominoes are free, we also assume that the associated pariominoes are free. We denote the positive and negative parity pariominoes by P_i^+ and P_i^-, respectively. Clearly, if we reverse the colouring of P_i^+, we obtain P_i^- (and vice versa). If the parity of a polyomino is zero, for notational convenience, we still represent the associated pariominoes by P_i^+ and P_i^-, respectively. In the special case when the parity of a polyomino is zero *and* $P_i^+ \neq P_i^-$, then the particular choice of which variant we label P_i^+ and which P_i^- is arbitrary (see Figure 5).



(a) P_i^+ **(b)** P_i^- **(c)** P_i^+ **(d)** P_i^- **(e)** P_i^+ **(f)** P_i^-

Figure 5. Examples of pariominoes. (**a**,**b**) $P_i^+ = P_i^-$ with $p_i = 0$; (**c**,**d**) $P_i^+ \neq P_i^-$ with $p_i = 0$; (**e**,**f**) $P_i^+ \neq P_i^-$ with $p_i = \pm 1$.

Depending on the symmetry of a polyomino P_i, the number of possible orientations after applying rotations and reflections is 1, 2, 4, or 8. The associated pariomino will inherit the same number of possible orientations, except in the special situation when,

(i) P_i has 1, 2, or 4 orientations, and
(ii) $P_i^+ = P_i^-$,

in which case we double the number of orientations. If $P_i^+ = P_i^-$, this implies that there exists a non-trivial rotation or a reflection that sends P_i^+ to P_i^- (and vice-versa), which implies a balanced polyomino, i.e., the parity is zero. The converse statement is false, as there are many examples where the parity of a polyomino is zero, but the associated pariominoes are distinct (illustrated in Figure 5c,d). An example of a pariomino that has

double the orientations of the corresponding polyomino is illustrated in Figures 6 and 7. Each orientation of a polyomino or a pariomino represents a fixed polyomino or pariomino, respectively. We show below that the possible orientations of a polyomino or pariomino play a crucial role in developing our ILP colouring techniques for tiling.

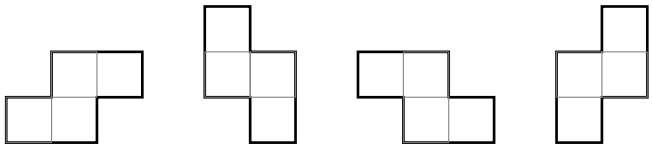


Figure 6. The 4 possible orientations of a tetromino P_i.

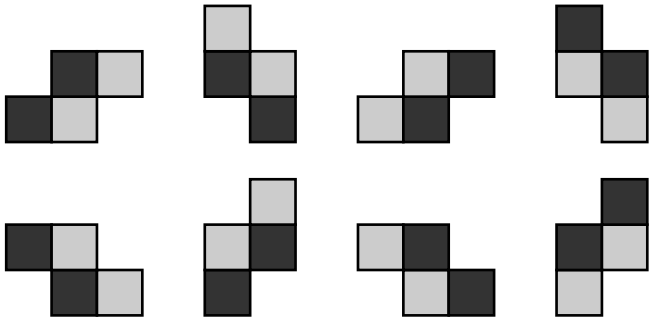


Figure 7. The 8 possible orientations of the pariomino P_i^+ $(= P_i^-)$ corresponding to the tetromino P_i of Figure 6.

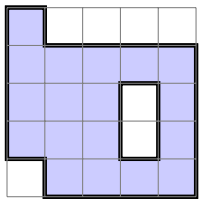
3. ILP Model Formulation

In this section, we outline the ILP method for tiling. Initially, we construct binary matrices that are the key components of our models. Then, for the reader's convenience, we review the derivation of the ILP formulations for the case of tiling with polyominoes [46] and then show how we adapt this approach to cover the case of tiling with pariominoes.

3.1. Construction of Binary Matrices

A key step in the ILP approach to tiling is to represent all placements of each polyomino in the target region R as binary matrices. The binary matrix associated with the j th placement of a polyomino P_i in R is denoted $A^{i,j}$. We also represent the target region R as a binary matrix denoted as B. The target region need not be rectangular and may have 'holes' and so, to represent it as a binary matrix, it is first positioned in the lattice of its rectangular hull. We illustrate an example in Figure 8 along with the associated binary matrix, where the (i, j) th entry of B is equal to the number '1' if there is a corresponding cell in the (i, j) th position of R; otherwise, the (i, j) th entry is equal to zero.

How we construct the binary matrices for each placement of a polyomino in R is similar. The (i, j)-th entry of the binary matrix $A^{i,j}$ is equal to the number '1' if there is a corresponding cell in the (i, j)-th position of R covered by a cell of the polyomino; otherwise, the (i, j)-th entry of $A^{i,j}$ is equal to zero. Figure 9 illustrates how we construct two binary matrices corresponding to different placements of a tetromino into R. Once we have constructed all the binary matrices, we assemble the ILP problem.

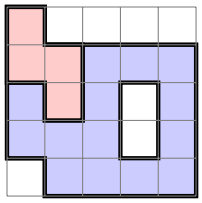


$$\longrightarrow \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

(**a**) (**b**)

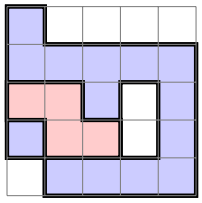
Figure 8. Constructing the binary matrix for R: (**a**) the target region R; (**b**) binary matrix B.

We must modify the procedure for constructing the binary matrices when fitting pariominoes into a checkerboard coloured region. Not only must the pariomino fit in the target region, but the colouring of cells covered must coincide with the colouring of cells in the pariomino, which generally yields a reduction in the total number of binary matrices compared to the polyomino tiling case. We construct the binary matrix B associated with a checkerboard coloured region in the same way as in the case without colouring. For example, Figure 10 illustrates how we construct B associated with a checkerboard colouring applied to the region in Figure 8. We obtain the same binary matrix if we reverse the checkerboard colouring of R, and the results of the checkerboard colouring approach in this work are independent of which colouring we use. Figure 11 illustrates the binary matrices associated with two different placements of the pariomino shown in Figure 7. To distinguish the pariominoes from the checkerboard coloured region containing them, we outline them in red. We denote the j-th placement of a pariomino P_i^+ or P_i^- in a checkerboard coloured target region R by $\widehat{A}^{i,j}$.



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

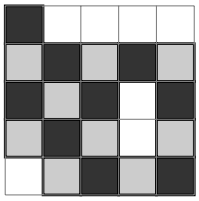
(**a**) (**b**)



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

(**c**) (**d**)

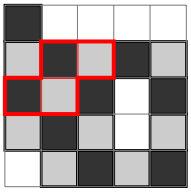
Figure 9. Constructing the binary matrices for two placements of a tetromino in R: (**a**) first placement of P_1; (**b**) binary matrix $A^{1,1}$; (**c**) second placement of P_1; (**d**) binary matrix $A^{1,2}$.



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

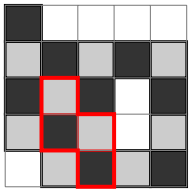
(**a**) (**b**)

Figure 10. Constructing the binary matrix for a checkerboard coloured region R: (**a**) the target region R; (**b**) binary matrix B.



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

(**a**) (**b**)



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

(**c**) (**d**)

Figure 11. Constructing the binary matrices for two placements of a pariomino in a checkerboard coloured region R: (**a**) first placement of P_1^+; (**b**) binary matrix $\widehat{A}^{1,1}$; (**c**) second placement of P_1^+; (**d**) binary matrix $\widehat{A}^{1,2}$. The pariominoes are outlined in red.

3.2. Assembly of the Linear Systems

For the reader's convenience, we briefly review how to assemble the linear system of equations that makes up the ILP formulation of the polyomino tiling problem (see [46] for further details) and then discuss the modified procedure required when tiling with pariominoes.

Assume we wish to tile a target region R with area c using F free polyominoes $\{P_i\}_{i=1}^F$ with areas $\{c_i\}_{i=1}^F$ and the numbers of copies $\{n_i\}_{i=1}^F$ ($n_i \geq 1$). The total number of polyominoes used is $N := \sum_{i=1}^F n_i$. Clearly, the total area of R must equal the sum of the areas of the polyominoes, i.e., $\sum_{i=1}^F c_i n_i = c$. Using a combination of rotations, reflections and translations, assume each polyomino P_i fits s_i ways into R. Recall the definitions of the binary matrices $A^{i,j}$ and B (see Section 3.1). We define *Series* i to be the set of all binary matrices $A^{i,j}$ associated with fitting the polyominoes P_i into R, given by

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

The total number of binary matrices $A^{i,j}$ is given by $n := \sum_{i=1}^F s_i$, which also corresponds to the total number of possible placements of all polyominoes in the region R.

We introduce a variable $\alpha_{i,j} \in \{0,1\}$, associated with $A^{i,j}$, for whether the j-th placement of P_i is used to tile R. The mathematical description of tiling the region R with copies of the F free polyominoes in P_i is a natural one: we would like to know if there is a linear combination of the binary matrices $A^{i,j}$, weighted by the respective $\alpha_{i,j}$ that is equal to B.

In this context, it is helpful to think of the binary matrix B as representing the fully tiled region R. The problem formulation is as follows.

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

$$\sum_{i=1}^{F} \sum_{j=1}^{s_i} \alpha_{i,j} A^{i,j} = B, \tag{3}$$

$$\text{subject to } \sum_{j=1}^{s_i} \alpha_{i,j} = n_i, \quad 1 \leq n_i \leq s_i, \quad i = 1, \ldots, F. \tag{4}$$

The constraints (4) enforce the conditions that we must use exactly n_i polyominoes from each series i to tile R. It is easy to verify that the problem formulation automatically enforces the condition that the polyomino areas sum to the area of the target region.

It is a standard first-year university linear algebra exercise to convert matrix equations of the type (3) into a system of linear equations. After re-arranging the left-hand side of Equation (3) into a single matrix, we equate the non-zero entries of B with the corresponding entries in this matrix yielding c linear equations. The ordering of equations corresponds to reading entries in B left-to-right and top-to-bottom. Together with the constraints (4), this yields the following binary linear system of m equations in n unknowns:

Seek $\boldsymbol{\alpha} \in \{0,1\}^n$ such that

$$M\boldsymbol{\alpha} = \widetilde{\boldsymbol{b}}, \quad M \in \{0,1\}^{m \times n}, \quad m = c + F, \quad n = \textstyle\sum_{i=1}^{F} s_i, \tag{5}$$

$$\text{and} \quad \{\widetilde{\boldsymbol{b}}\}_i = \begin{cases} 1 & \text{for } i = 1, \ldots, c \\ n_i & \text{for } i = c+1, \ldots, m \end{cases},$$

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

The vertical lines in the solution vector separate the unknowns according to their corresponding series. Notice that the constraints (4) appear here as the $c + k$-th equations, $k = 1, 2, \ldots F$, namely (4), which just enforces the condition that exactly n_k coefficients associated with each series k are equal to '1', and the remaining coefficients are equal to zero. It is also helpful to note that if we neglect the rows in M corresponding to these constraints, then the columns of this 'reduced' coefficient matrix arise from the entries in the binary matrices $A^{i,j}$ written row-wise.

We claim that every problem, monohedral or multihedral, has one extraneous constraint equation because the main linear system corresponding to the first c equations essentially specifies the total number of squares covered. Once we impose all but one of the F constraint equations, we can work out the last constraint. We verify this by summing the first c equations in (5) to give

$$c_1 \sum_{j=1}^{s_1} \alpha_{1,j} + c_2 \sum_{j=1}^{s_2} \alpha_{2,j} + \cdots + c_F \sum_{j=1}^{s_F} \alpha_{F,j} = c,$$

recalling that $c = \sum_{i=1}^{F} c_i n_i$, and imposing the $F - 1$ constraints (4). Hence, in the monohedral case, we do not have to include the single constraint in this formulation and so $m = c$. However, for simplicity in our ILP formulations, we impose all the constraints in the multihedral case and exclude the single constraint in the monohedral case.

How we construct the linear system for the pariomino tiling problem is similar to the polyomino tiling problem, but with some important differences. Similar to the polyomino tiling problem, when we use multiple ($\geq$2) free pariominoes to tile a region we refer to this problem as *pari-multihedral*; otherwise we refer to the pariomino tiling problem as *pari-monohedral*. We note it is possible for colouring techniques applied to a monohedral tiling problem to yield a pari-multihedral tiling problem. The notation we use for the linear system (5) still applies, but with some small changes. First, we denote the binary

matrices $\widehat{A}^{i,j}$ instead of $A^{i,j}$ (see Section 3.1). Second, we denote the checkerboard coloured target region $\widehat{R}$, to distinguish it from the uncoloured region R. Third, the number of free pariominoes used to tile $\widehat{R}$, denoted $\widehat{F}$, can differ from the number of free polyominoes F. This is because each polyomino P_i with non-zero parity has two pariominoes associated with it, namely P_i^+ and P_i^-, which are possibly distinct. Thus, generally we have $\widehat{F} \geq F$. As in the polyomino tiling problem, the number of unknowns n is given by the total number of ways the distinct tiles (in this case, the $\widehat{F}$ pariominoes) fit into the target region $\widehat{R}$. The number of equations m is also determined as in the polyomino tiling problem. In the pari-multihedral case, $m = c + \widehat{F}$, while in the pari-monohedral case, $m = c$.

The subtlety of the colouring approach relies on how we choose sets of pariominoes that split the full (uncoloured) problem into multiple independently solvable subproblems, covered in the next section.

4. A Splitting Method Using Parity Constraints

In [47], the authors used constraints on the parity values of the tiles to identify impossible tiling problems. We adapt these techniques here to split large tiling problems into many subproblems that admit a parallel solution strategy.

We attempt to tile a region R with parity p using F free polyominoes $\{P_i\}_{i=1}^{F}$ with copies $\{n_i\}_{i=1}^{F}$, $n_i \geq 1$. Assume the polyominoes have parity values $\{\pm p_i\}_{i=1}^{F}$ (if the parity value is zero then for ease of notation, we still represent the parity as $\pm p_i$). Denote the multiset of all possible sums of $N = \sum_{i=1}^{F} n_i$ parity values by $\{S_j\}_{j=1}^{\mathcal{T}}$, $\mathcal{T} \geq 1$, namely, all possible sums of

$$\begin{cases} n_1 \text{ elements drawn from } \{-p_1, +p_1\}, \text{ with} \\ n_2 \text{ elements drawn from } \{-p_2, +p_2\}, \text{ with} \\ \qquad \vdots \qquad\qquad\qquad \vdots \\ n_F \text{ elements drawn from } \{-p_F, +p_F\}, \end{cases} \tag{6}$$

where $n_i \in \mathbb{N}$ and $p_i, p \in \mathbb{N} \cup \{0\}$, for $i = 1 \ldots F$.

It is easy to prove the combinatorial result that [47]:

Proposition 1. *The number of sums of parity values that we can form from* (6) *is*

$$\mathcal{T} = \prod_{k=1}^{F} (1 + n_k).$$

The following simple result is key to the arguments that follow [47]:

Proposition 2. *A necessary (but not sufficient) condition for the polyominoes to tile a region R is that $S_j = p$ for at least one $j \in \{1, 2, \ldots, \mathcal{T}\}$.*

When none of the possible parity sums equals the parity of the target region, we call this a *parity violation*, which implies that it is impossible for the given polyominoes to tile the target region. In this work, we are more interested when we know ahead of time that the polyominoes *do* tile the target region R, in which case many combinations of parity values may satisfy Proposition 2. Each solution satisfying Proposition 2 corresponds to a separate tiling subproblem with pariominoes that may or may not tile $\widehat{R}$. The total number of possible tiling solutions is partitioned among all the pariomino tiling subproblems.

It is possible to compute all solutions satisfying Proposition 2 using brute-force techniques. However, this rapidly becomes intractable for large problems. The decision problem of whether there is a solution that satisfies Proposition 2 is related to the general subset sum problem, which is $\mathcal{NP}$-complete [61]. For this reason, we apply a systematic mathematical

procedure for solving this problem using linear Diophantine equations. We adapt the arguments given below from [47].

If all F polyominoes have zero parity, then we must have $p = 0$ so Proposition 2 is trivially satisfied. Now consider the case that at least one polyomino has non-zero parity. Assume, without loss of generality, the first r polyominoes, $0 \leq r \leq F-1$ have zero parity. Let the variable $a_i^+ \in \mathbb{N} \cup \{0\}$ denote how many of the n_i *non-negative* elements drawn from $\{-p_i, +p_i\}$ are $+p_i$, $0 \leq a_i^+ \leq n_i$, $i = r+1, r+2, \ldots, F$. We derive an equation in the variables a_i^+ that matches the sum of the parities of the tiles to the parity of the target region. As we have a_i^+ choices for how many of the n_i elements drawn from $\{-p_i, +p_i\}$ are $+p_i$, we must have $n_i - a_i^+$ choices of $-p_i$, $0 \leq a_i^+ \leq n_i$, $i = r+1, r+2, \ldots F$. Then the equation $s_j = p$ takes the form

$$(+p_{r+1})a_{r+1}^+ + (-p_{r+1})(n_{r+1} - a_{r+1}^+) + (+p_{r+2})a_{r+2}^+ + (-p_{r+2})(n_{r+2} - a_{r+2}^+) + \ldots$$
$$+ (+p_F)a_F^+ + (-p_F)(n_F - a_F^+) = p,$$

or after some simplification

$$2p_{r+1}a_{r+1}^+ + 2p_{r+2}a_{r+2}^+ + \cdots + 2p_F a_F^+ = p + p_{r+1}n_{r+1} + p_{r+2}n_{r+2} + \cdots + p_F n_F,$$

for $r \in \{0, 1, \ldots, F-1\}$, which is a linear Diophantine equation in $F - r$ unknowns $\{a_i^+\}_{i=r+1}^F$. We can easily verify that it is safe to divide both sides by 2, yielding the following.

Theorem 1. *The number of distinct solutions satisfying Proposition 2 is given by the number of solutions of the following linear Diophantine equation in $F - r$ unknowns $\{a_i^+\}_{i=r+1}^F$:*

$$\begin{aligned} &p_{r+1}a_{r+1}^+ + p_{r+2}a_{r+2}^+ + \cdots + p_F a_F^+ = k \in \mathbb{Z}, \\ &\quad \textit{where} \quad k := (p + p_{r+1}n_{r+1} + p_{r+2}n_{r+2} + \cdots + p_F n_F)/2, \\ &\textit{for} \quad 0 \leq a_i^+ \leq n_i, \quad a_i^+ \in \mathbb{N} \cup 0, \quad i = r+1, r+2, \ldots, F. \end{aligned} \tag{7}$$

A necessary condition for the existence of integer solutions (and hence also for non-negative integer solutions) of Equation (7) is $\gcd(p_{r+1}, p_{r+2}, \ldots, p_F) \mid k$, where $\gcd(p_{r+1}, p_{r+2}, \ldots, p_F)$ denotes the greatest common divisor of $p_{r+1}, p_{r+2}, \ldots, p_F$ [62] (p. 30). When we know there is a solution to the full (uncoloured) tiling problem, then this condition is automatically satisfied. Suppose a solution $(a_1^+, a_2^+, \ldots, a_F^+)$ exists. Consider an arbitrary polyomino P_i. Notice that if $a_i^+ = n_i$ or $a_i^+ = 0$, then we tile the coloured region R with n_i copies of the pariomino P_i^+ or P_i^- respectively. That is, there is a single pariomino associated with P_i in the pariomino tiling problem. On the other hand, if $a_i^+ \neq n_i$ and $a_i^+ \neq 0$, then we use a_i^+ copies of P_i^+ *and* $n_i - a_i^+$ copies of P_i^- in the pariomino tiling problem, i.e., we use both P_i^+ and P_i^-. Thus, the relationship between the number of free polyominoes F and the associated number of free pariominoes $\widehat{F}$ is given by

$$\widehat{F} = \sum_{i=1}^{F} \widehat{F}_i \quad \text{where} \quad \widehat{F}_i := \begin{cases} 1 & \text{if } a_i^+ = n_i \text{ or } 0 \\ 2 & \text{otherwise} \end{cases}, \quad i = 1, 2, \ldots, F.$$

As each solution satisfying Proposition 2 yields a separate pariomino tiling subproblem, Theorem 1 yields a practical means of splitting the full (uncoloured) tiling problem into separate subproblems that are independently solvable. Furthermore, the full set of solutions is partitioned among the solutions of the subproblems, and thus, the problem of finding all solutions is embarrassingly parallel.

As we saw from the simple example outlined in the introduction, there is another mechanism that splits a tiling problem into subproblems, separate from those identified using Theorem 1. If some parities are zero with $P_i^+ \neq P_i^-$, then more subproblems exist,

as each polyomino of this type yields two choices for the pariominoes, multiplying the number of subproblems by a factor of two in each instance.

For simplicity, assume for a tiling problem that none of the polyominoes satisfies $p_i = 0$ and $P_i^+ \neq P_i^-$, i.e., we exclude the alternate 'splitting' mechanism described above. Then Algorithm 1 gives a simple procedure for finding all tiling solutions (the algorithm is easily modified to account for the alternate 'splitting' mechanism if required). Our starting point is a target region R with area c and parity p, which we aim to tile using polyominoes $\{P_i\}_{i=1}^F$ with areas $\{c_i\}_{i=1}^F$ and copies $\{n_i\}_{i=1}^F$, $n_i \geq 1$. As stated earlier, we assume the parities $p_1, p_2, \ldots, p_r$ are zero ($0 \leq r \leq F-1$). We also assume that the tiling problem has at least one solution.

Algorithm 1 To find all possible solutions of a polyomino tiling problem using checkerboard colouring arguments and ILP

1: Input $\{p_i\}_{i=1}^F$, $\{c_i\}_{i=1}^F$, p, and c
2: $S \leftarrow \emptyset$
3: Identify r from the user input of $\{p_i\}_{i=1}^F$
4: $k \leftarrow (p + p_{r+1}n_{r+1} + p_{r+2}n_{r+2} + \cdots + p_F n_F)/2$
5: Compute the nt ($nt \geq 1$) non-negative integer solutions of (7) {yielding nt pariomino tiling subproblems}
6: **for** $i = 1$ to nt **do**
7: Construct the ILP file for the ith pariomino tiling subproblem and save
8: **end for**
9: **for all** subproblems $i = 1$ to nt **do in parallel**
10: Compute the solutions of the i th ILP file using an optimizer and add to S
11: **end for**
12: Output S {all tiling solutions}

A small modification of Algorithm 1 yields an algorithm for computing a single solution to one of the pariomino subproblems in the least amount of time. In line 10, instead of calculating all solutions for each subproblem, we seek a single optimal solution and break the 'parallel-for-loop' as soon as soon as we find the first optimal solution.

5. Some Theoretical Considerations

In this section, we consider the related concepts of problem complexity; measures of algorithm performance; and problem classification. Before discussing the results from the numerical examples of Section 6, we make some general comments that relate to the questions raised in the introduction.

5.1. Complexity

One way to estimate the complexity of a problem is to measure the computational work required to solve it. We can also use complexity to compare the difficulty of two problems or the effectiveness of two algorithms. For the tiling problem, some factors that might form a complexity measure on theoretical grounds include the number of unknowns, the number of constraints, and the number of free variables of the associated ILP problem.

Recall that the number of binary matrices for a tiling problem yields the number of unknowns in the resulting linear system. When we apply checkerboard colouring techniques, if we use every pariomino P_i^+ and P_i^-, for $i = 1, 2, \ldots, F$, the number of unknowns in the linear systems for the pariomino and polyomino tiling problems are the same. However, for each polyomino P_i when the corresponding pariomino used to tile R is P_i^+ or P_i^- (but not both), then the number of binary matrices (and hence the number of unknowns) in the linear system for the pariomino tiling problem is usually reduced, compared to the corresponding polyomino tiling problem. However, it is possible to construct some problems where this is not the case (see Section 6.2).

It is also worth considering in the multihedral case the role played by the number of constraints on the problem complexity. Each polyomino or pariomino contributes a constraint to the associated linear systems. Assuming the ILP problem is feasible, then it is reasonable to assume from a computational perspective that the more constraints we have the better. This is because the linear systems are usually underdetermined, and so it is helpful if the augmented matrix is as close to full rank as possible. As we often have $\widehat{F} \geq F$ we expect the rank of the augmented matrix for many pariomino tiling problems to be closer to the full rank than the augmented matrix for the polyomino tiling problem (even when the number of unknowns in the two systems is the same).

Thus to summarize, it is reasonable when we consider both the number of unknowns and the number of constraints in an ILP problem, that in general, we see an improvement in the solvability of a pariomino tiling subproblem compared to the solvability of the full (uncoloured) tiling problem. However, because of the complexity of the general tiling situation, we cannot prove the superiority either algorithmically or theoretically.

After many experiments, none of these measures seem to provide a reliable work estimate when judged by the time it takes an ILP solver to handle the problem. This may in part be because the ILP solver has a variety of techniques available and there are many features of a problem hidden from our analysis, which makes a simple complexity formula impossible. The possibly hidden structure in an ILP problem either hinders or helps the ILP solver to find solutions [63,64]. Without a complexity formula based on theoretical grounds, we use the runtime of a fixed computer system. In our experience, this produces a reliable, repeatable, and practical substitute for a complexity formula. Although runtimes differ depending on the computer architecture and even the software version, it allows us to make practical comparisons between different tiling problems, or between subproblems of the same tiling problem.

5.2. Measures of Performance

With the runtime as our basic measure of computational work, we also need an appropriate measure of improvements in runtime obtained using our checkerboard colouring technique. In parallel computing, the *speedup* of an algorithm for solving a given problem with a fixed number of processors is given by [65]

$$\text{speedup} = \frac{\text{sequential runtime}}{\text{parallel runtime}}.$$

As this article provides only proof of concept of a parallel implementation, in place of speedup for finding *all* tiling solutions of a problem, we define the *potential speedup* for a tiling problem to be

$$\text{potential speedup} = \frac{\text{runtime to find all solutions of a polyomino tiling problem}}{\text{longest runtime to find all solutions of a pariomino tiling subproblem}}.$$

When seeking a single optimal solution, we define the *potential speedup* as

$$\text{potential speedup} = \frac{\text{runtime to find an optimal solution of a polyomino tiling problem}}{\text{shortest runtime to find an optimal solution for a pariomino tiling subproblem}}.$$

5.3. Problem Classification

We classify the different situations arising when we apply checkerboard colouring arguments to a polyomino tiling problem. Either we have a monohedral polyomino tiling problem, or a multihedral polyomino tiling problem. For these two types of problems, the full tiling problem either splits into multiple pariomino tiling subproblems, or it does not (i.e., there is a single pariomino tiling problem to consider). In each of these four situations,

the resulting pariomino tiling problems may or may not yield a reduction in complexity (compared to the full uncoloured tiling problem), yielding the eight cases illustrated in Figure 12.

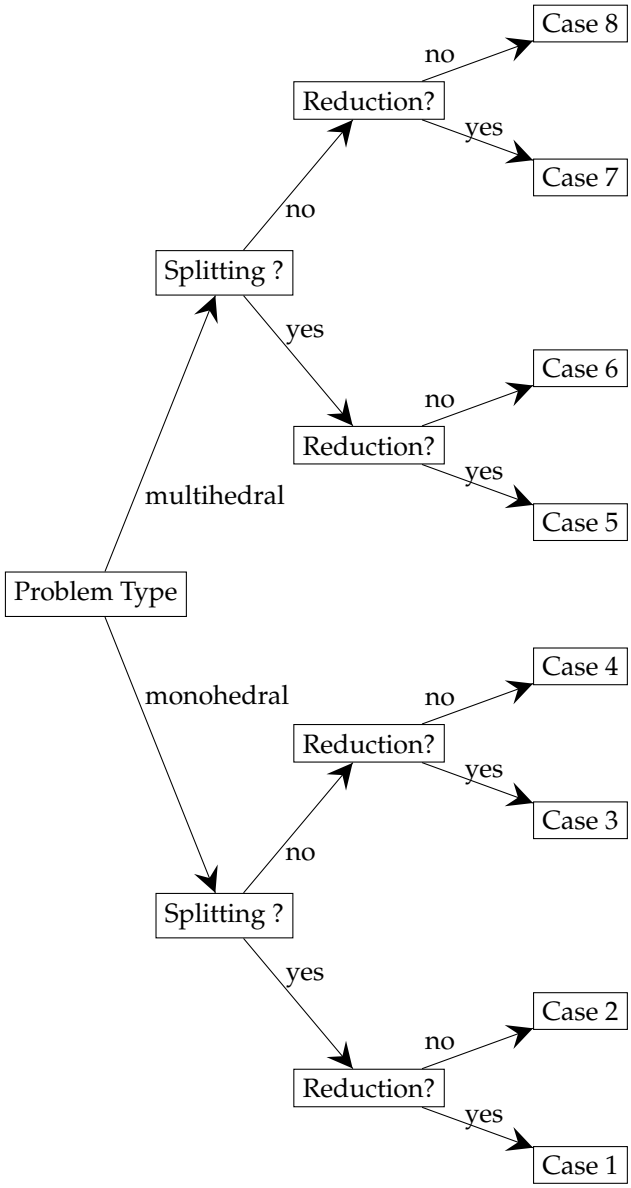


Figure 12. A decision tree for the following alternatives: monohedral polyomino tiling problem versus multihedral polyomino tiling problem; checkerboard colouring technique applied to a tiling problem splits the problem into multiple ($\geq$2) tiling pariomino subproblems versus no splitting; checkerboard colouring technique applied to a tiling problem yields a reduction in complexity of the associated pariomino tiling subproblem(s) versus no reduction in complexity.

6. Numerical Examples

We solve many tiling problems, either for all solutions, or a single (optimal) solution, using appropriate software.

We introduce some notation that helps us refer to the number of pariomino variants used in each tiling problem. Recall when we have a_i^+ copies of a pariomino P_1^+ with positive parity, then we have $n_i - a_i^+$ copies of a pariomino P_1^- with negative parity. For notational convenience in this section, we define $a_i^- := n_i - a_i^+$, so $a_i^+ + a_i^- = n_i$, for $i = 1, 2, \ldots, F$. We must be a little careful with our notation when the parity of a polyomino is zero (recall, in this situation for notational convenience, we still refer to the P_1^+ and P_1^-

variants). If $P_1^+ \neq P_1^-$ and $p_i = 0$ we allow ourselves a slight abuse of notation by still using a_i^+ to represent the number of P_1^+ variants and a_i^- to represent the number of P_1^- variants. However, we cannot use this notation when $P_1^+ = P_1^-$ (and so $p_i = 0$) as in this case, the number of copies used is always n_i.

6.1. Scientific Computing

We solved all tiling problems using `MATLAB` (R2020b) and `CPLEX` (12.8.0.0), a commercial optimization package, see https://www.ibm.com/analytics/cplex-optimizer (accessed on 6 May 2022), run on a MacBook Pro (OS X 10.15.7) with 16 GB memory and 2.7 GHz Intel Core i7. In a previous study [46] we verified that `CPLEX` was very effective at solving the ILP problems arising from tiling with polyominoes, and was generally faster than two other popular high-performance optimization packages, namely `Gurobi` (7.5.2), see http://www.gurobi.com (accessed on 6 May 2022), and `SCIP` (6.0.0), see http://scip.zib.de (accessed on 6 May 2022). The shell commands needed in `CPLEX` to find either all feasible solutions, or a single optimal solution, are given in [46].

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

(i) Construct the linear system in `MATLAB` and export the associated ILP file to `CPLEX`;
(ii) Solve the ILP file with `CPLEX` 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.

We provide all necessary files for other researchers to reproduce the experimental results in this article. The collections of `MATLAB` files needed to carry out steps (i) and (iii) above for the polyomino and pariomino tiling problems are freely available. The associated repositories are `POLYOMINOES` (v2.1.4), see https://doi.org/10.5281/zenodo.6366101 (accessed on 6 May 2022), and `PARIOMINOES` (v3.0.0), see https://doi.org/10.5281/zenodo.6366094 (accessed on 6 May 2022). For every example, we provide the appropriate `MATLAB` 'LP make' files. In the polyomino tiling case, solutions are plotted in `MATLAB` using the programs `PLOT_MONO` or `PLOT_MULTI`, depending on whether the problem is of the monohedral or multihedral type, respectively. In the pariomino tiling case, solutions are plotted in `MATLAB` using the program `PLOT_PARIOMINO`. The `MATLAB` program `DIOPHANTINE_ND_NONNEGATIVE_BOUNDED`, included in the `PARIOMINOES` repository, was used to solve the linear Diophantine Equations (7) of Theorem 1. This program is similar in construction to the related Diophantine solvers described at [47].

From a computational perspective, it would simplify the solution process if we could encode the parity constraints of Equation (7) directly into the ILP formulation. However, before the ILP formulation can be constructed we must first solve (7) yielding sets of pariominoes, where each set represents a separate pariomino tiling subproblem. Then, for each subproblem, the `MATLAB` programs in the repository `PARIOMINOES` are used to construct an ILP file. This is a non-trivial task because before we can compute the constraints we must compute all possible placements of the pariominoes in the checkerboard coloured target region. In addition to fitting the pariominoes in the region (using a combination of rotations, reflections and translations) we also ensure that the colours of squares covered match the colours of the squares of the tiles (see Section 3 for further details).

6.2. Tiling Examples Where All Solutions Are Sought

We give tiling examples for seven out of the eight cases illustrated in Figure 12. In all examples, we compare the solution process with and without using checkerboard colouring techniques. In the ILP context, we use a zero objective function and refer to the m linear equations in the resulting binary linear systems as constraints. The number of free variables f relates to the reduced row echelon form of the augmented systems $[M|\widetilde{\mathbf{b}}]$. When we apply colouring techniques, we say the full (uncoloured) tiling problem yields a 'splitting' if there are two or more associated pariomino tiling subproblems. We say there is a 'reduction in complexity' if the maximum CPLEX runtime for all pariomino tiling subproblems is less than the runtime required to solve the full (uncoloured) tiling problem,

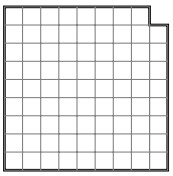
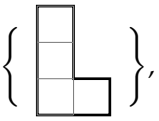
i.e., the potential speedup is greater than 1 (see Section 5.2). As a check of the correctness of our results, in each example, we checked that the total number of solutions found after solving all subproblems equaled the total number of solutions found for the uncoloured tiling problem.

According to our classification system, monohedral tiling problems naturally fall into one category on the tree in Figure 12. However, in contrast to the multihedral case, it was not always possible to find completely satisfactory representatives for each case of the monohedral tiling problems. Some cases of questionable significance yielded a small reduction in computational time. Other contrived cases, which were found with a satisfactory reduction in computational time, are not typical. Thus, some of the monohedral examples labeled with an asterisk (*) indicate that their exact placement in our classification tree is questionable.

6.2.1. Case 1

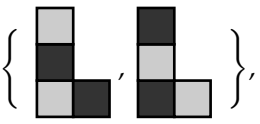
We give an example of the monohedral polyomino type, where applying colouring techniques splits the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems, with a reduction in complexity.

Example 1. *Find all ways of tiling the 9-by-9 single-notched square*

$R =$  *with* $\{P_1\} =$  ,

all orientations permitted. The area of R is 80, so we need 20 copies of the L-shaped tetromino. The associated binary ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE2`*, has n = 442 unknowns, m = 80 constraints, and f = 363 free variables.* `CPLEX` *found 1,709,594 solutions in 367.9 seconds.*

Using colouring techniques we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 13a using 20 pariominoes from the set*

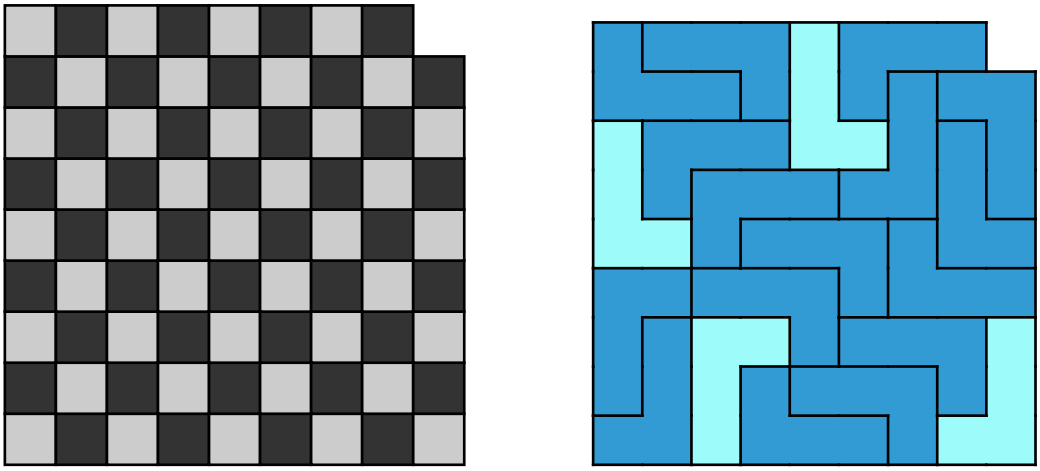
$\{P_1^+, P_1^-\} =$  ,

all orientations permitted.

As the tetromino and R have zero parity and $P_1^+ \neq P_1^-$ *we automatically obtain 21 tiling subproblems to consider with* a_1^+ *and* $a_1^- = 20 - a_1^+$ *copies of* P_1^+ *and* P_1^- *respectively, given by* (a_1^+, a_1^-) *for* $a_1^+ = 0, 1, \ldots, 20$*. Solving the subproblems in* `CPLEX` *yielded the data for the nine feasible solutions shown in Table 1, where we report the number of solutions found (# solns.), and the runtime (in seconds).*

Table 1. Number of tiling solutions (# solns.) and runtimes in seconds for `CPLEX` to find the feasible solutions of the pariomino tiling problem in Example 1.

a_1^+	20	18	16	14	12	10	8	6	4
a_1^-	0	2	4	6	8	10	12	14	16
# solns.	406	9762	72,308	252,844	475,908	503,612	296,004	88,498	10,212
runtime (s)	0.09	2.16	16.30	72.49	121.67	138.52	83.01	28.88	3.15



(**a**) (**b**)

Figure 13. Target region and a tiling solution for Example 1: (**a**) The target region $\widehat{R}$, a checkerboard coloured 9-by-9 single-notched square; (**b**) Successful tiling of the target region $\widehat{R}$ using 20 copies of an L-shaped tetromino. The P_1^+ variants are coloured light-blue and the P_1^- variants coloured dark-blue corresponding to an optimal solution of the subproblem with $(a_1^+, a_1^-) = (4, 16)$ (see Table 1).

A single (optimal) tiling solution corresponding to the subproblem with $(a_1^+, a_1^-) = (4, 16)$ *is shown in Figure 13b, i.e., using four copies of* P_1^+ *and 16 copies of* P_1^-*. The ILP file in this case was constructed using the program* `LPMAKE_PARIOMINOES_FIGURE2` *(similar programs were used to construct the ILP files for the other cases). The subproblems with either* $a_1^+ = 20$ *and* $a_1^- = 0$*, or* $a_1^+ = 0$ *and* $a_1^- = 20$*, yielded* $(m, n, f) = (80, 220, 144)$ *and* $(m, n, f) = (80, 222, 145)$ *respectively. All other subproblems yielded* $(m, n, f) = (82, 442, 362)$*. The maximum runtime for the infeasible solutions was 0.62 seconds. The subproblem with the maximum runtime of* 138.52 *seconds yielded 503,612 solutions, thus the potential speedup when using the colouring technique is about 2.7×.*

6.2.2. Case 2

We give two examples of the monohedral polyomino type, where the application of colouring techniques splits the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems, with no reduction in complexity.

Example 2 (*). *Find all ways of tiling the 18 × 24 rectangle with*

$$\{P_1\} = \left\{ \right\},$$

all orientations permitted. The area of R is 432, so we need 72 copies of the hexomino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE3`*, has n = 2816 unknowns, m = 432 constraints, and f = 2385 free variables.* `CPLEX` *found 414 solutions in* 14.22 *seconds.*

Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 14a using 72 pariominoes from the set*

$$\{P_1^+, P_1^-\} = \left\{ \quad , \quad \right\},$$

all orientations permitted.

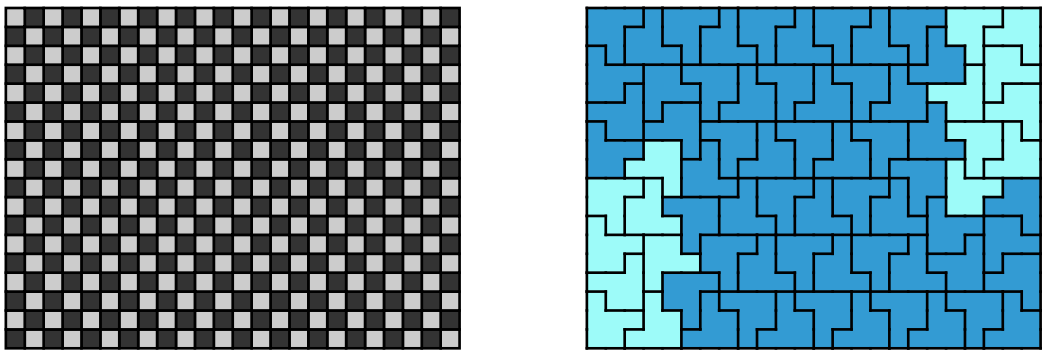
As the hexomino and R have zero parity and $P_1^+ \neq P_1^-$ *we automatically obtain 73 tiling subproblems to consider with* a_1^+ *and* $a_1^- = 72 - n_1$ *copies of* P_1^+ *and* P_1^- *respectively, given by*

(a_1^+, a_1^-) *for* $a_1^+ = 0, 1, \ldots, 72$. *Solving the subproblems in* `CPLEX` *yielded the data for the 23 feasible solutions shown in Table 2.*

A single (optimal) tiling solution corresponding to the subproblem with $(a_1^+, a_1^-) = (16, 56)$ *is shown in Figure 14b. The program* `LPMAKE_PARIOMINOES_FIGURE3` *was used to construct the ILP file in this case. (Similar programs were used to construct the ILP files for the other cases.) The subproblems with either* $a_1^+ = 0$ *and* $a_1^- = 72$, *or* $a_1^+ = 72$ *and* $a_1^- = 0$, *yielded* $(m, n, f) = (432, 1408, 978)$. *All other subproblems yielded* $(m, n, f) = (434, 2816, 2384)$. *The maximum runtime for the infeasible solutions was 6.3 seconds. The subproblem with the maximum runtime of 15.30 seconds yielded 94 solutions, which is an* increase *in the runtime of approximately 7.60% compared to the full (uncoloured) tiling problem, i.e., no potential speedup was observed.*

Table 2. Number of tiling solutions (# solns.) and runtimes in seconds for `CPLEX` to find the feasible solutions of the pariomino tiling problem in Example 2.

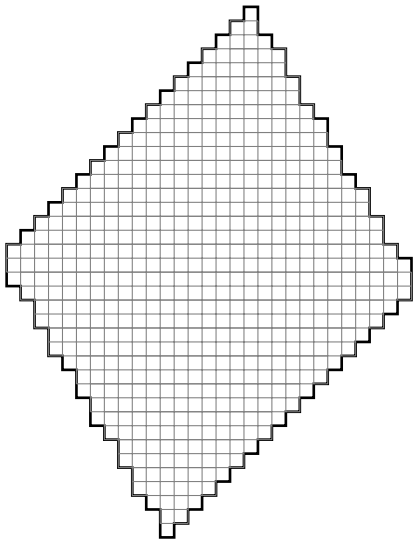
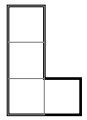
a_1^+	16	26	27	28	29	30	31	32	33	34	35	36
a_1^-	56	46	45	44	43	42	41	40	39	38	37	36
# solns.	1	4	2	4	6	9	12	31	26	51	14	94
runtime (s)	2.2	8.7	8.5	10.2	8.8	10.1	9.8	13.3	9.6	11.0	9.9	15.3
a_1^+	37	38	39	40	41	42	43	44	45	46	56	-
a_1^-	35	34	33	32	31	30	29	28	27	26	16	-
# solns.	14	51	26	31	12	9	6	4	2	4	1	-
runtime (s)	12.8	12.6	10.6	12.6	12	10.7	8	8.8	7.6	8.1	3.7	-



(**a**) (**b**)

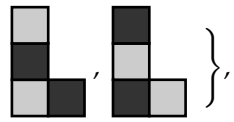
Figure 14. Target region and a tiling solution for Example 2: (**a**) the target region $\widehat{R}$, a checkerboard coloured 18×24 rectangle; (**b**) successful tiling of the target region $\widehat{R}$ using 72 copies of a hexomino. The P_1^+ variants are coloured light-blue and the P_1^- variants coloured dark-blue corresponding to the optimal solution of the subproblem with $(a_1^+, a_1^-) = (16, 56)$ (see Table 2).

Example 3 (*). *Find all ways of tiling the irregularly shaped region*

$R =$  *with* $\{P_1\} =$  ,

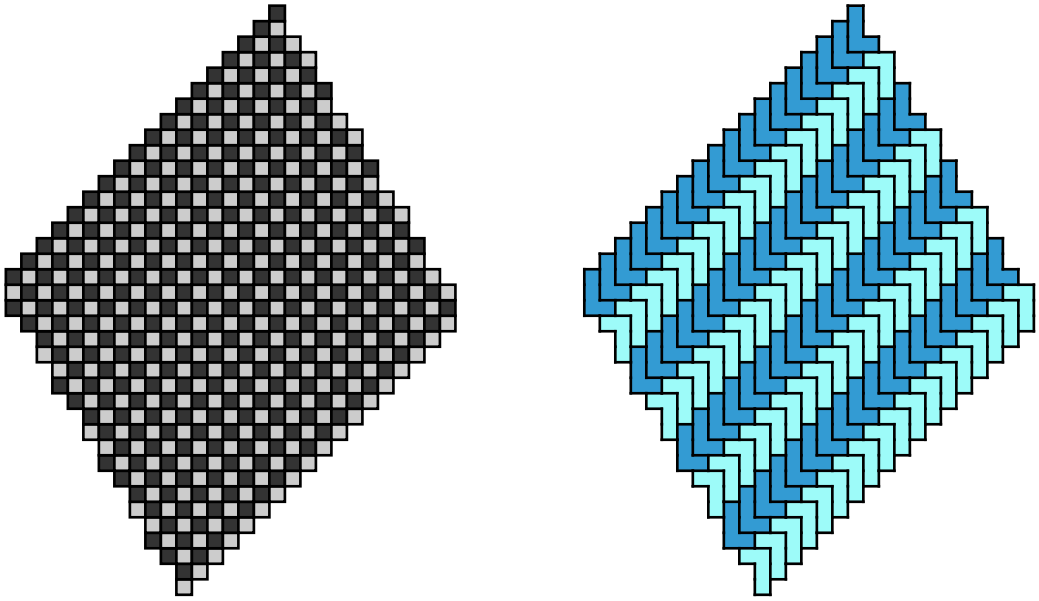
all orientations permitted. The area of R is 576, so we need 144 copies of the L-shaped tetromino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE4`*, has n = 3952 unknowns, m = 576 constraints, and f = 3377 free variables.* `CPLEX` *found a single solution in 0.15 seconds.*

Using colouring techniques we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 15a using 144 pariominoes from the set*

$\{P_1^+, P_1^-\} =$  ,

all orientations permitted.

As the tetromino and R have zero parity and $P_1^+ \neq P_1^-$ *we automatically get 145 tiling subproblems to consider with* a_1^+ *and* $a_1^- = 144 - a_1^+$ *copies of* P_1^+ *and* P_1^- *respectively, given by* (a_1^+, a_1^-) *for* $a_1^+ = 0, 1, \ldots, 144$. `CPLEX` *found a single feasible solution of the subproblems with* $(m, n, f) = (578, 3952, 3376)$*, corresponding to* $(a_1^+, a_1^-) = (72, 72)$ *in 0.21 seconds, which is an* increase *in the runtime of 40% compared to the runtime needed to solve the full (uncoloured) tiling problem, i.e., no potential speedup was observed. However, as the runtimes are so small, this is not a particularly meaningful measure. The single pariomino tiling solution is shown in Figure 15b. We constructed the ILP file in this case using the program* `LPMAKE_PARIOMINOES_FIGURE4` *(similar programs were used to construct the ILP files for the other cases).*



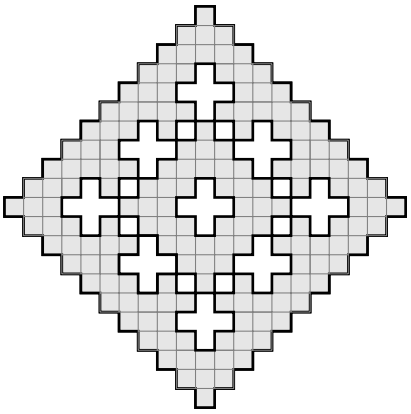
(**a**) (**b**)

Figure 15. Target region and a tiling solution for Example 3: (**a**) The target region $\widehat{R}$, a checkerboard coloured irregularly shaped region; (**b**) successful tiling of the target region $\widehat{R}$ using 72 copies of an L-shaped tetromino. The P_1^+ variants are coloured light-blue, and the P_1^- variants coloured dark-blue corresponding to the single feasible solution of the subproblems when $(a_1^+, a_1^-) = (72, 72)$.

6.2.3. Case 3

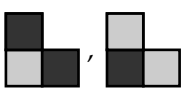
We give two examples of the monohedral polyomino type, where the application of colouring techniques does not split the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems and reduces the complexity.

Example 4 (*). *Find all ways of tiling the diamond shaped region with 'holes'*

$R =$  *with* $\{P_1\} =$  ,

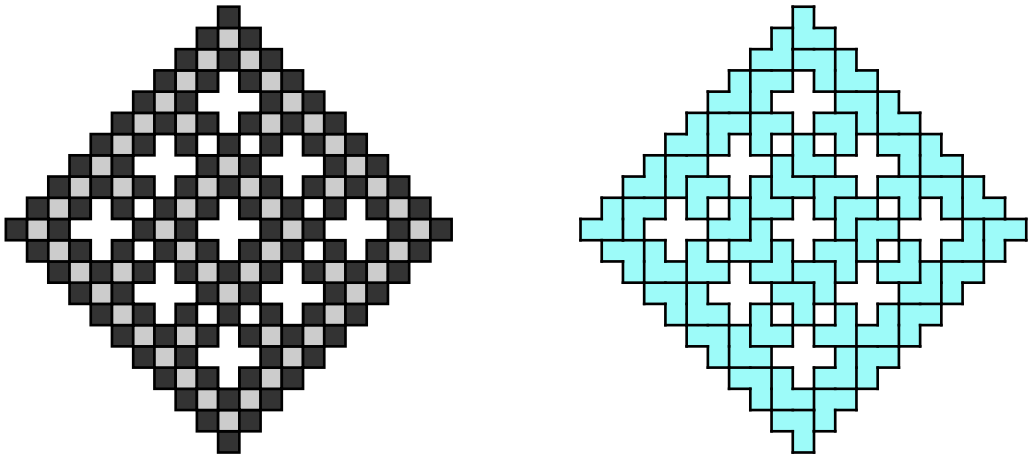
all orientations permitted. The area of R is 168, so we need 56 copies of the L-shaped trimino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE5`*, has n = 336 unknowns, m = 168 constraints, and f = 168 free variables.* `CPLEX` *found 16 solutions in 0.05 seconds.*

Using colouring techniques we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 16a using 56 pariominoes from the set*

$\{P_1^+, P_1^-\} =$  ,

with parities $\pm p_1 = \pm 1$*, all orientations permitted. The parity of* $\widehat{R}$ *is* $p = +56$*, thus the only way to satisfy the parity equation (see Theorem 1) is to choose* $a_1^+ = 56$ *and* $a_1^- = 0$*. That is, the only way to tile* $\widehat{R}$ *is to use the pariominoes* P_1^+ *and we have a single pariomino tiling subproblem. We constructed the ILP file using the program* `LPMAKE_PARIOMINOES_FIGURE5`*.* `CPLEX` *found 16 solutions with* $(m, n, f) = (168, 224, 60)$ *in 0.02 seconds, thus the potential speedup when using the colouring technique is about* $2.5\times$*. However, as the runtimes are very small, this is not a particularly meaningful measure. The optimal solution of the pariomino tiling subproblem is shown in Figure 16b.*

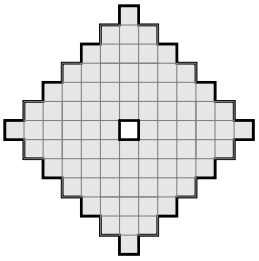
Observe that for the polyomino tiling problem, we have n = 336 and for the pariomino tiling subproblem we have n = 224, which represent the number of ways P_1 *fits in R and* P_1^+ *fits in* $\widehat{R}$*, respectively. Thus, the number of ways* P_1^- *fits in* $\widehat{R}$ *is* $336 - 224 = 112$*, even though the single tiling solution uses only pariominoes of type* P_1^+*.*



(**a**) (**b**)

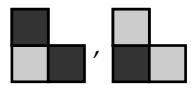
Figure 16. Target region and a tiling solution for Example 4: (**a**) the target region $\widehat{R}$, a checkerboard coloured irregularly shaped region with 'holes'; (**b**) successful tiling of the target region $\widehat{R}$ using 56 copies of an L-shaped trimino. The P_1^+ variants are coloured light-blue corresponding to an optimal solution of the single pariomino tiling subproblem.

Example 5. *Find all ways of tiling the diamond shaped region with a single 'hole' in the middle*

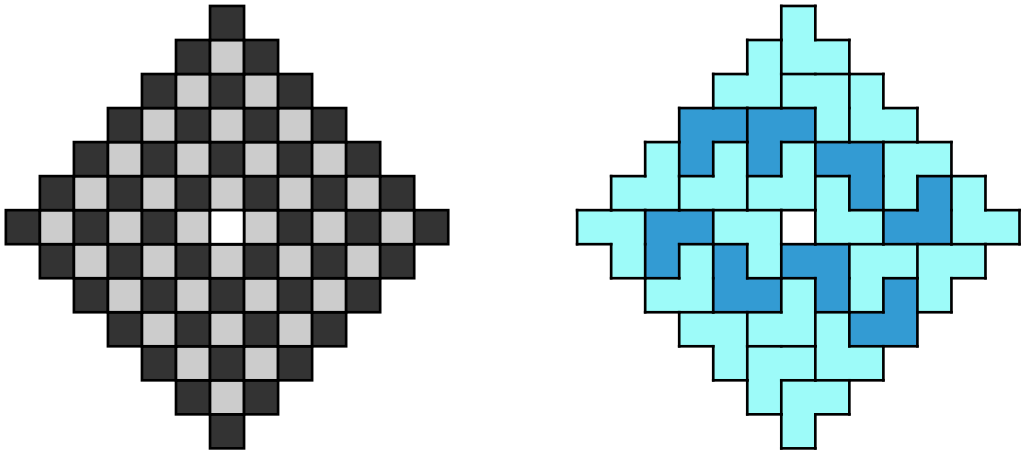
$R =$  *with* $\{P_1\} =$  ,

all orientations permitted. The area of R is 84, so we need 28 copies of the L-shaped trimino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE6`*, has n = 252 unknowns, m = 84 constraints, and f = 168 free variables.* `CPLEX` *found 731,092 solutions in 165.78 seconds.*

Using colouring techniques we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 17a using 28 pariominoes from the set*

$\{P_1^+, P_1^-\} =$  ,

with parities $\pm p_1 = \pm 1$*, all orientations permitted. The parity of* $\widehat{R}$ *is* $p = +12$*, thus, the only way to satisfy the parity equation (see Theorem 1) is to choose* $a_1^+ = 20$ *and* $a_1^- = 8$*, i.e., we have a single pariomino tiling subproblem using 20 copies of* P_1^+ *and 8 copies of* P_1^-*. The ILP file was constructed using the program* `LPMAKE_PARIOMINOES_FIGURE6`*. Solving this subproblem in* `CPLEX` *yielded 731,092 solutions with* $(m, n, f) = (86, 252, 168)$ *in 156.20 seconds, thus the potential speedup when using the colouring technique is about 1.1×. The optimal solution of the pariomino tiling subproblem is shown in Figure 17b.*



(**a**) (**b**)

Figure 17. Target region and a tiling solution for Example 5: (**a**) The target region $\widehat{R}$, a checkerboard coloured diamond shaped region with a single 'hole' in the middle; (**b**) Successful tiling of the target region $\widehat{R}$ using 28 copies of an L-shaped trimino. The 20 pariominoes P_1^+ are coloured light-blue and the 8 pariominoes P_1^- are coloured dark-blue, corresponding to an optimal solution of the single pariomino tiling subproblem.

6.2.4. Case 4

We give an example of the monohedral polyomino type, where the application of colouring techniques does not split the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems, and there is *no* reduction in complexity.

Example 6. *Find all ways of tiling the 16 × 18 rectangle with*

$$\{P_1\} = \left\{ \quad \right\},$$

all orientations permitted. The area of R is 288, so we need 48 copies of the hexomino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE7`*, has n = 892 unknowns, m = 288 constraints, and f = 609 free variables.* `CPLEX` *found 217,266 solutions in 45.55 seconds.*

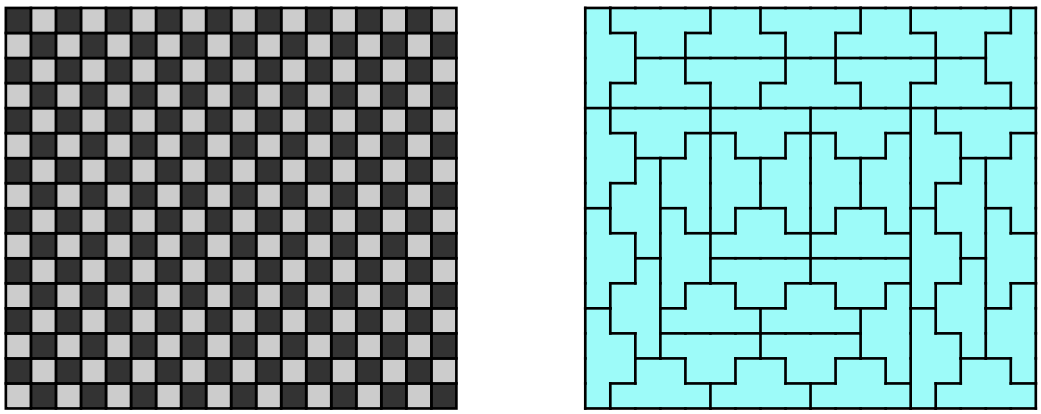
Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 18a using 48 pariominoes from the set*

$$\{P_1^+\} = \left\{ \quad \right\},$$

all orientations permitted. We note here that $P_1^+ = P_1^-$ *and so we only tile with copies of a single pariomino.*

As the hexomino and R have zero parity, the parity equation (see Theorem 1) is trivially satisfied and we obtain a single pariomino tiling subproblem. We constructed the ILP file in this case, using the program `LPMAKE_PARIOMINOES_FIGURE7`*. Solving this subproblem in* `CPLEX` *with* $(m, n, f) = (288, 892, 609)$ *yielded 217,266 solutions in 50.75 seconds, i.e., an increase in the runtime of approximately 11.42% compared to the full (uncoloured) tiling problem. Thus we do not observe a potential speedup. An optimal solution of the pariomino tiling subproblem is illustrated in Figure 18b.*

The pariomino tiling subproblem is essentially equivalent to the full polyomino tiling problem. Both formulations have the same binary matrices and no constraints of the form (4). Thus, the difference in the time taken for `CPLEX` *to solve the polyomino tiling problem and the pariomino tiling subproblem is simply because of differences in the ordering of the unknowns in the binary linear systems. This results from the different order in which the binary matrices are constructed by our* `MATLAB` *programs for the polyomino and pariomino tiling cases. This situation also applies to any monohedral tiling problem where* $P_1^+ = P_1^-$ *(so* $p_1 = 0$*), for example, any pure domino tiling problem.*



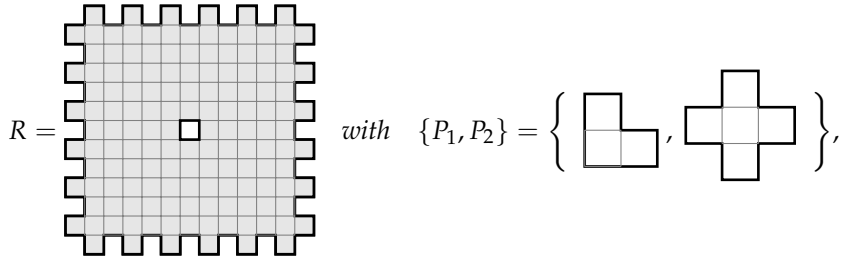
(**a**) (**b**)

Figure 18. Target region and a tiling solution for Example 6: (**a**) the target region $\widehat{R}$, a 16×18 rectangle; (**b**) successful tiling of the target region $\widehat{R}$ using 48 pariominoes P_1^+ $(= P_1^-)$, coloured light-blue, corresponding to an optimal solution of the single pariomino tiling subproblem.

6.2.5. Case 5

We give four examples of the multihedral type, where the application of colouring techniques splits the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems, with a reduction in complexity.

Example 7. *Find all ways of tiling the jagged-square shaped region with a single 'hole' in the middle*

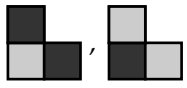


all orientations permitted. The area of R is 144, which we aim to tile using $n_1 = 33$ *copies of an L-shaped trimino and* $n_2 = 9$ *copies of the cross-shaped pentomino. The associated ILP file,*

constructed with the program `LPMAKE_POLYOMINOES_FIGURE8`*, has n = 528 unknowns, m = 146 constraints, and f = 383 free variables.* `CPLEX` *found 731,092 solutions in 165.78 seconds.*

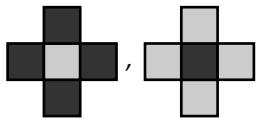
Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 19a using 33 pariominoes from the set*

$$\{P_1^+, P_1^-\} = \left\{ \quad , \quad \right\},$$



with parities $\pm p_1 = \pm 1$*, and 9 pariominoes from the set*

$$\{P_2^+, P_2^-\} = \left\{ \quad , \quad \right\},$$



with parities $\pm p_2 = \pm 3$*, all orientations permitted. The parity equation (see Theorem 1) is given by*

$$p_1 a_1^+ + p_2 a_2^+ = (p + p_1 n_1 + p_2 n_2)/2, \quad 0 \le a_i^+ \le n_i, \quad i = 1, 2.$$

After noting that $p = +24$ *we obtain the linear Diophantine equation*

$$a_1^+ + 3a_2^+ = 42, \quad 0 \le a_1^+ \le 33, \quad 0 \le a_2^+ \le 9.$$

The `MATLAB` *program* `DIOPHANTINE_ND_NONNEGATIVE_BOUNDED` *was used to solve this equation yielding seven pariomino tiling subproblems, solved in* `CPLEX` *to give the data in Table 3 for the number of pariomino tiling solutions (# solns.) and* `CPLEX` *runtimes (sec).*

Table 3. Number of tiling solutions (# solns.) and runtimes in seconds for `CPLEX` to solve the seven pariomino tiling subproblems in Example 7.

a_1^+	a_2^+	n	m	f	# Solns.	Runtime (s)
15	9	492	147	347	-	0.00
18	8	528	148	382	-	0.04
21	7	528	148	382	-	0.04
24	6	528	148	382	62,960	15.21
27	5	528	148	382	339,224	106.42
30	4	528	148	382	16,256	2.84
33	3	336	147	191	-	0.10

The fifth subproblem took the longest to solve (106.42 s), thus the potential speedup when using the colouring technique is about 1.6×. We found an optimal solution using `CPLEX` *for the fourth subproblem, i.e., using 24 copies of* P_1^+*, 9 copies of* P_1^-*, 6 copies of* P_2^+*, and 3 copies of* P_2^- *(see Figure 19b). We constructed the ILP file in this case, using the program* `LPMAKE_PARIOMINOES_FIGURE8` *(similar programs were used to construct the ILP files for the other cases).*

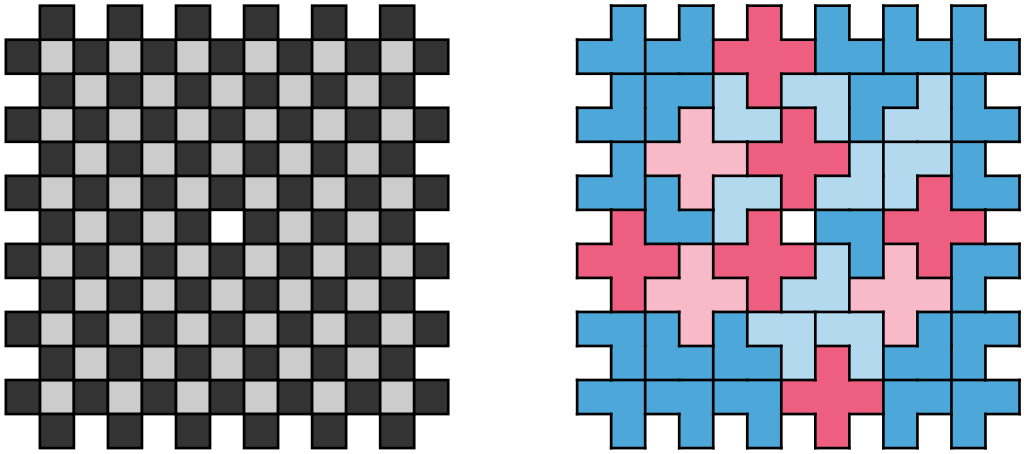
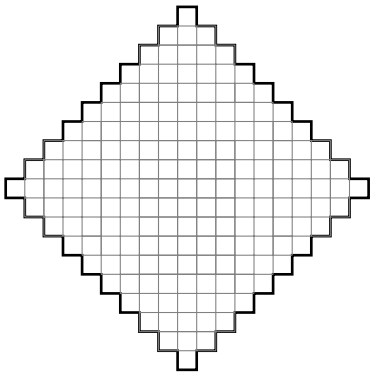


Figure 19. Target region and a tiling solution for Example 7: (**a**) The target region $\widehat{R}$, a checkerboard coloured jagged-square shaped region with a single 'hole' in the middle. (**b**) Successful tiling of the target region $\widehat{R}$ using 33 copies of an L-shaped trimino and 9 copies of the cross-shaped pentomino. The optimal solution corresponds to the fourth pariomino tiling subproblem in Table 3 where we use 24 copies of P_1^+, 9 copies of P_1^-, 6 copies of P_2^+, and 3 copies of P_2^-. The pariominoes P_1^+, P_1^-, P_2^+ and P_2^- are coloured dark blue, light blue, dark red, and light red, respectively.

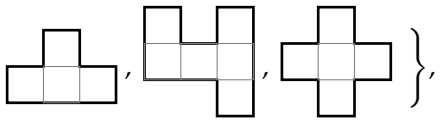
Example 8. *Find all ways of tiling the diamond-shaped region*

$R =$



with

$\{P_1, P_2, P_3\} =$



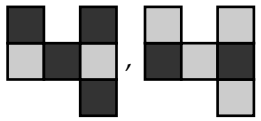
all orientations permitted. The area of R is 181, which we aim to tile using $n_1 = 34$ copies of the T-shaped trimino, $n_2 = 5$ copies of Y-shaped hexomino, and $n_3 = 3$ copies of the cross-shaped pentomino. The ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE9`*, has $n = 1685$ unknowns, $m = 184$ constraints, and $f = 1502$ free variables.* `CPLEX` *found 337,680 solutions in 225.06 seconds.*

Using colouring techniques we wish to tile the checkerboard coloured region $\widehat{R}$ shown in Figure 20a using 34 pariominoes from the set

$\{P_1^+, P_1^-\} =$

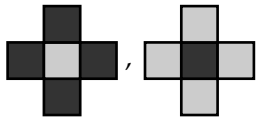
with parities $\pm p_1 = \pm 2$*, and 5 pariominoes from the set*

$$\{P_2^+, P_2^-\} = \left\{ \quad , \quad \right\},$$



with parities $\pm p_2 = \pm 2$*, and 3 pariominoes from the set*

$$\{P_3^+, P_3^-\} = \left\{ \quad , \quad \right\},$$



with parities $\pm p_2 = \pm 3$*, all orientations permitted. The parity equation (see Theorem 1) is given by*

$$p_1 a_1^+ + p_2 a_2^+ + p_3 a_3^+ = (p + p_1 n_1 + p_2 n_2 + p_3 n_3)/2, \quad 0 \le a_i^+ \le n_i, \quad i = 1, 2, 3.$$

After noting that $p = +19$ *we obtain the linear Diophantine equation*

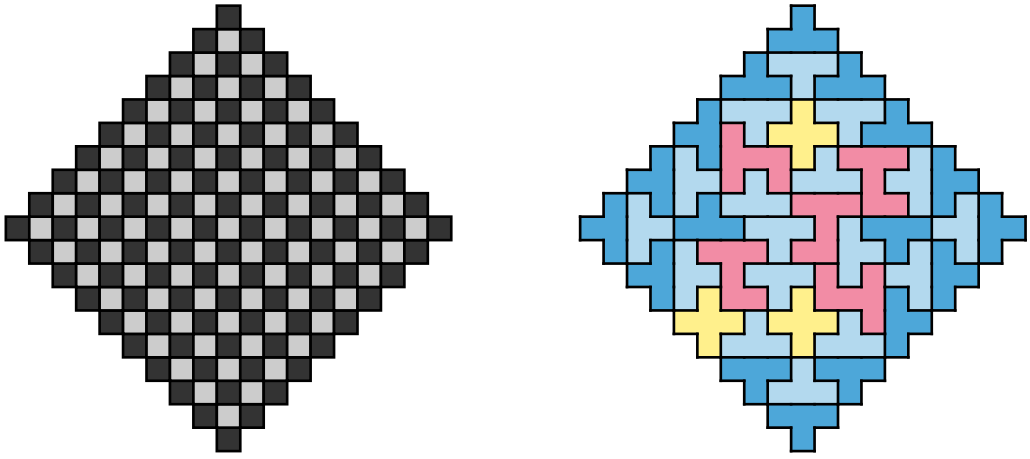
$$2a_1^+ + 2a_2^+ + 3a_3^+ = 53, \quad 0 \le a_1^+ \le 34, \quad 0 \le a_2^+ \le 5, \quad 0 \le a_3^+ \le 3.$$

The `MATLAB` *program* `DIOPHANTINE_ND_NONNEGATIVE_BOUNDED` *was used to solve this equation yielding 12 pariomino tiling subproblems, solved in* `CPLEX` *to give the data in Table 4 for the number of pariomino tiling solutions and* `CPLEX` *runtimes.*

Table 4. Number of tiling solutions (# solns.) and runtimes in seconds for `CPLEX` to solve the 12 pariomino tiling subproblems in Example 8.

a_1^+	a_2^+	a_3^+	n	m	f	# Solns.	Runtime (s)
17	5	3	1173	185	990	344	0.53
18	4	3	1621	186	1437	7528	8.32
19	3	3	1621	186	1437	53,144	43.81
20	2	3	1621	186	1437	125,104	72.01
20	5	1	1237	186	1053	-	0.39
21	1	3	1621	186	1437	117,192	72.09
21	4	1	1685	187	1500	-	1.64
22	0	3	1109	185	926	34,272	15.00
22	3	1	1685	187	1500	32	3.57
23	2	1	1685	187	1500	64	3.09
24	1	1	1685	187	1500	-	1.59
25	0	1	1173	186	989	-	0.68

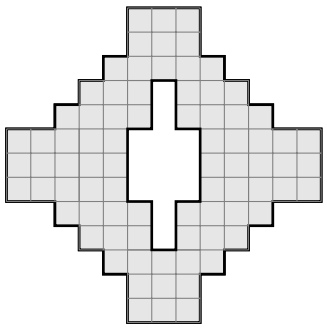
The sixth subproblem took the longest to solve (72.09 s), and thus the potential speedup when using the colouring technique is about $3.1\times$*. We found an optimal solution using* `CPLEX` *for the first subproblem, i.e., using 17 copies of* P_1^+*, 17 copies of* P_1^-*, 5 copies of* P_2^+*, and 3 copies of* P_3^+ *(see Figure 20b). We constructed the ILP file in this case using the program* `LPMAKE_PARIOMINOES_FIGURE9` *(similar programs were used to construct the ILP files for the other cases).*



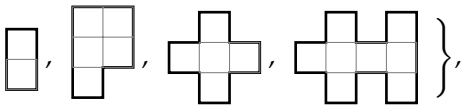
(**a**) (**b**)

Figure 20. Target region and a tiling solution for Example 8: (**a**) the target region $\widehat{R}$, a checkerboard coloured diamond-shaped region; (**b**) successful tiling of the target region $\widehat{R}$ using 34 copies of a T-shaped trimino, 5 copies of the Y-shaped hexomino, and 3 copies of the cross-shaped pentomino. The optimal solution corresponds to the first pariomino tiling subproblem in Table 4 using 17 copies of P_1^+, 17 copies of P_1^-, 5 copies of P_2^+, and 3 copies of P_3^+. The pariominoes P_1^+, P_1^-, P_2^+, and P_3^+ are coloured dark blue, light blue, red, and yellow, respectively.

Example 9. *Find all ways of tiling the region*

$R =$ 

with

$\{P_1, P_2, P_3, P_4\} =$  ,

all orientations permitted. The area of R is 80, which we aim to tile with $n_1 = 26$ *copies of the domino,* $n_2 = 3$ *copies of the P-shaped pentomino,* $n_3 = 1$ *copies of the cross-shaped pentomino, and* $n_4 = 1$ *copies of the octomino. The associated ILP file, constructed with the program* `LPMAKE_POLYOMINOES_FIGURE10`*, has* $n = 444$ *unknowns,* $m = 84$ *constraints, and* $f = 361$ *free variables.* `CPLEX` *found 105,344 solutions in 16.83 seconds.*

Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 21a using 26 pariominoes from the set*

$\{P_1^+\} =$  ,

with parity $p_1 = 0$, $(P_1^+ = P_1^-)$, *and three pariominoes from the set*

$$\{P_2^+, P_2^-\} = \left\{ \quad , \quad \right\},$$

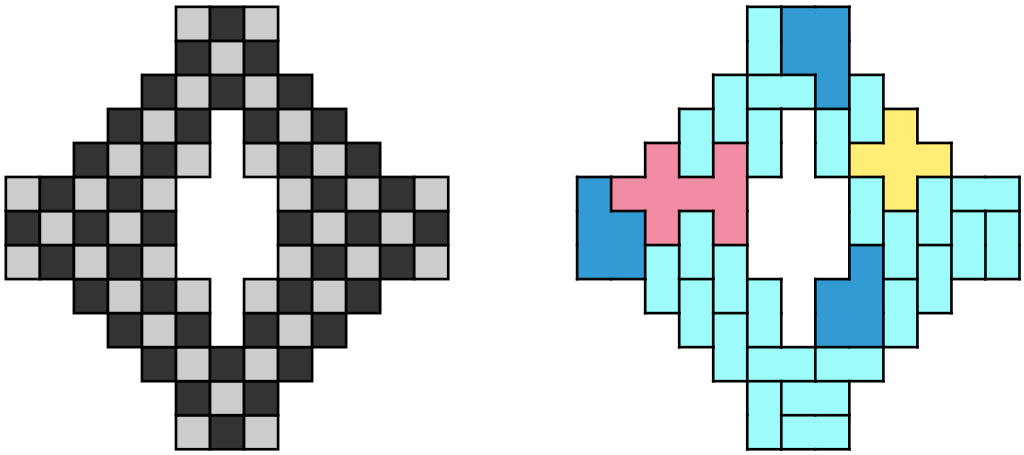
with parities $\pm p_2 = \pm 1$, *and one pariomino from the set*

$$\{P_3^+, P_3^-\} = \left\{ \quad , \quad \right\},$$

with parities $\pm p_2 = \pm 3$, *and one pariomino from the set*

$$\{P_4^+, P_4^-\} = \left\{ \quad , \quad \right\},$$

with parities $\pm p_2 = \pm 4$, *all orientations permitted.*



(**a**) (**b**)

Figure 21. Target region and a tiling solution for Example 9: (**a**) The checkerboard coloured target region $\widehat{R}$; (**b**) Successful tiling of the target region $\widehat{R}$ using 26 copies of a domino, 3 copies of the P-shaped pentomino, one copy of the cross-shaped pentomino, and one copy of an octomino. The optimal solution corresponds to the first pariomino tiling subproblem in Table 5 using 26 copies of P_1^+, three copies of P_2^-, one copy of P_3^+, and one copy of P_4^+. The pariominoes P_1^+, P_2^-, P_3^+, and P_4^+ are coloured light blue, blue, yellow and red, respectively.

The parity equation (see Theorem 1) is given by

$$p_2 a_2^+ + p_3 a_3^+ + p_4 a_4^+ = (p + p_2 n_2 + p_3 n_3 + p_4 n_4)/2, \quad 0 \le a_i^+ \le n_i, \quad i = 1, 2, 3, 4.$$

After noting that $p = +4$, *we obtain the linear Diophantine equation*

$$a_2^+ + 3a_3^+ + 4a_4^+ = 7, \quad 0 \le a_2^+ \le 3, \qquad 0 \le a_3^+, a_4^+ \le 1.$$

The `MATLAB` *program* `DIOPHANTINE_ND_NONNEGATIVE_BOUNDED` *was used to solve this equation, yielding 2 pariomino tiling subproblems, solved in* `CPLEX` *to give the data in Table 5 for the number of pariomino tiling solutions and* `CPLEX` *runtimes.*

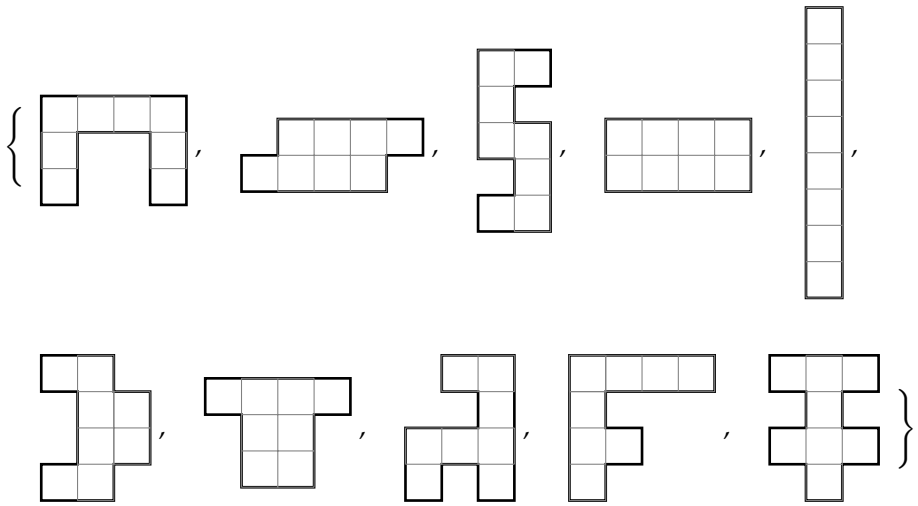
Table 5. Number of tiling solutions (# solns.) and runtimes in seconds for `CPLEX` to solve the two pariomino tiling subproblems in Example 9.

a_2^+	a_3^+	a_4^+	n	m	f	# Solns.	Runtime (s)
0	1	1	282	84	200	105,344	14.28
3	0	1	302	84	220	-	0.03

The first (and only feasible) subproblem took 14.28 seconds to solve, thus the potential speedup when using the colouring technique is about 1.2×. We constructed the ILP file, in this case, using the program `LPMAKE_PARIOMINOES_FIGURE10`*. We found an optimal solution using* `CPLEX` *for the first subproblem, i.e., using 26 copies of* P_1^+*, three copies of* P_2^-*, one copy of* P_3^+*, and one copy of* P_4^+ *(see Figure 21b).*

Example 10. *Find all ways of tiling the 8 × 16 rectangle using the following octominoes*

$$\{P_i\}_{i=1}^{10} =$$



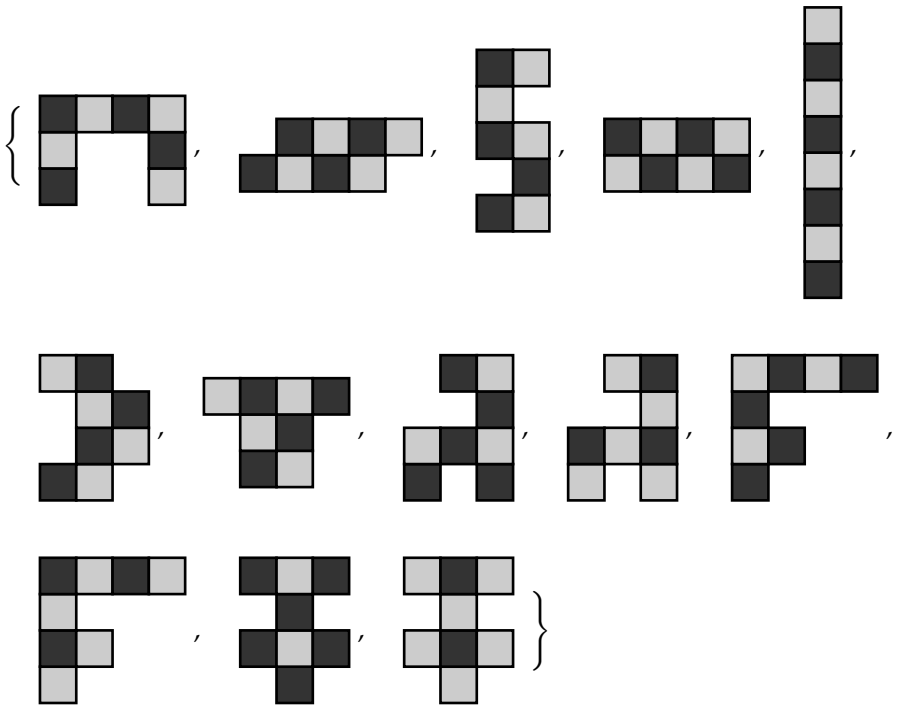
all orientations permitted, with the following numbers of copies:

$$(n_1, n_2, n_3, n_4, n_5, n_6, n_7, n_8, n_9, n_{10}) = (1, 2, 1, 2, 2, 1, 2, 2, 2, 1).$$

The sum of the areas of the tiles $8\sum_{k=1}^{8} n_k$ *equals the area of the target region as required. The associated ILP file, constructed with the program* `LPMAKE_POLYOMINOES_FIGURE11`*, has n = 3126 unknowns, m = 138 constraints, and f = 2989 free variables.* `CPLEX` *found 96 solutions in 163.95 seconds.*

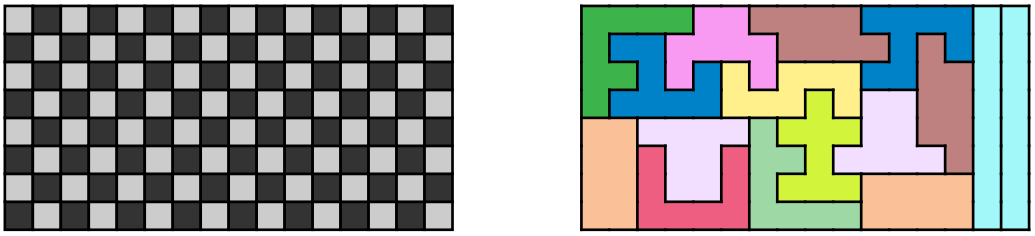
Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ *shown in Figure 22a, using* n_i *pariominoes from the sets* $\{P_i^+\}$ *for i = 1, 2, . . . , 7, and* n_j *pariominoes from the sets* $\{P_j^+, P_j^-\}$ *for j = 8, 9, 10, with*

$$\{P_1^+, P_2^+, P_3^+, P_4^+, P_5^+, P_6^+, P_7^+, P_8^+, P_8^-, P_9^+, P_9^-, P_{10}^+, P_{10}^-\} =$$



all orientations permitted, where $P_i^+ = P_i^-$ *for* $i = 1, 2, \ldots, 7$ *and* $P_j^+ \neq P_j^-$ *for* $j = 8, 9, 10$. *The parities of the parominoes are given by*

$$\{p_1, p_2, p_3, p_4, p_5, p_6, p_7, +p_8, -p_8, +p_9, -p_9, +p_{10}, -p_{10}\}$$
$$= \{0, 0, 0, 0, 0, 0, 0, +2, -2, +2, -2, +4, -4\}.$$



(**a**) (**b**)

Figure 22. Target region and a tiling solution for Example 10: (**a**) The target region $\widehat{R}$, a checkerboard coloured 8×16 rectangle; (**b**) Successful tiling of the target region $\widehat{R}$ using 16 octominoes. The optimal solution corresponds to the first pariomino tiling subproblem in Table 6 with $(a_8^+, a_8^-, a_9^+, a_9^-, a_{10}^+, a_{10}^-) = (0, 2, 1, 1, 1, 0)$. The pariomino P_9^+ is coloured dark green to distinguish it from P_9^-, which is coloured light green.

The parity equation (see Theorem 1) is given by

$$p_8 a_8^+ + p_9 a_9^+ + p_{10} a_{10}^+ = (p + p_8 n_8 + p_9 n_9 + p_{10} n + 10)/2 \quad \text{for} \quad 0 \leq a_i^+ \leq n_i, \quad i = 8, 9, 10.$$

After noting that $p = 0$ *we obtain the linear Diophantine equation*

$$2a_8^+ + 2a_9^+ + 4a_{10}^+ = 6, \quad 0 \leq a_8^+, a_9^+ \leq 2, \quad 0 \leq a_{10}^+ \leq 1.$$

The `MATLAB` *program* `DIOPHANTINE_ND_NONNEGATIVE_BOUNDED` *was used to solve this equation yielding four pariomino tiling subproblems, solved in* `CPLEX` *to give the data in Table 6 for the number of pariomino tiling solutions and* `CPLEX` *runtimes.*

Table 6. Number of tiling solutions (# solns.) and runtimes in seconds for `CPLEX` to solve the four pariomino tiling subproblems in Example 10.

a_8^+	a_9^+	a_{10}^+	n	m	f	# Solns.	Runtime (s)
0	1	1	2682	139	2545	48	47.33
1	0	1	2718	139	2581	-	33.35
1	2	0	2718	139	2581	-	49.31
2	1	0	2682	139	2545	48	54.14

The fourth subproblem took the longest to solve (54.14 s), thus the potential speedup when using the colouring technique is about 3.0×. We found an optimal solution using `CPLEX` *for the first subproblem, i.e., using n_i copies of the pariominoes P_i^+ $(= P_i^-)$ for $i = 1, 2, \ldots, 7$, two copies of P_8^-, and one copy each of P_9^+, P_9^- and P_{10}^+ (see Figure 22b). We constructed the ILP file in this case using the program* `LPMAKE_PARIOMINOES_FIGURE11` *(similar programs were used to construct the ILP files for the other cases).*

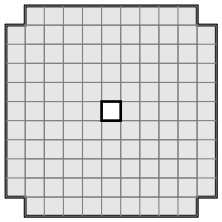
6.2.6. Case 6

We could not find an example of the multihedral type, where the application of colouring techniques splits the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems and there is *no* reduction in complexity.

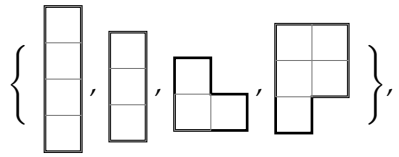
6.2.7. Case 7

We give an example of the multihedral polyomino type, where the application of colouring techniques does not split the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems and reduces the complexity.

Example 11. *Find all ways of tiling the notched square-shaped region with a single 'hole'*

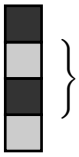
$R =$ 

with

$\{P_1, P_2, P_3, P_4\} =$  ,

all orientations permitted. The area of R is 116, which we aim to tile using $n_1 = 25$ copies of the straight tetromino, $n_2 = 1$ copy of the straight trimino, $n_3 = 1$ copy of the L-shaped trimino, and $n_4 = 2$ copies of the P-shaped pentomino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE12`*, has $n = 1376$ unknowns, $m = 120$ constraints, and $f = 1257$ free variables.* `CPLEX` *found 429,800 solutions in 768.74 seconds.*

Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ shown in Figure 23a using 25 pariominoes from the set

$\{P_1^+\} =$  ,

with parity $\pm p_1 = 0$, $(P_1^+ = P_1^-)$, *and one pariomino from the set*

$$\{P_2^+, P_2^-\} = \left\{ \quad , \quad \right\},$$

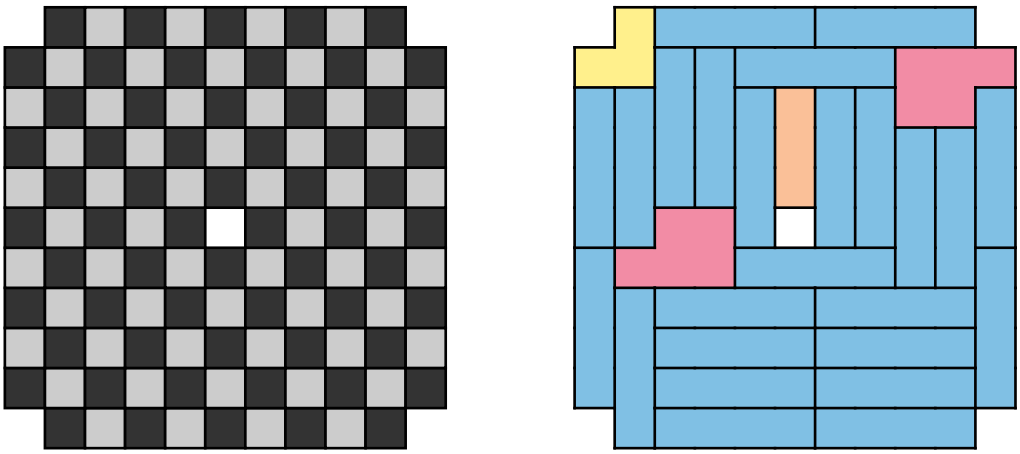
with parities $\pm p_2 = \pm 1$, *and one pariomino from the set*

$$\{P_3^+, P_3^-\} = \left\{ \quad , \quad \right\},$$

with parities $\pm p_3 = \pm 1$, *and two pariomino from the set*

$$\{P_4^+, P_4^-\} = \left\{ \quad , \quad \right\},$$

with parities $\pm p_4 = \pm 1$, *all orientations permitted. As the parity of* $\widehat{R}$ *is* $+4$, *the only way to satisfy the parity equation (see Theorem 1) is to choose* 25 *copies of* P_1^+ $(=P_1^-)$, *one copy of* P_2^+, *one copy of* P_3^+, *and two copies of* P_4^+. *We constructed the ILP file in this case using the program* `LPMAKE_PARIOMINOES_FIGURE12`. *Solving this single subproblem in* `CPLEX` *with* $(m, n, f) = (120, 784, 666)$ *yielded 429,800 solutions in 156.46 seconds, thus the potential speedup when using the colouring technique is about* $4.9\times$. *An optimal solution of this pariomino tiling subproblem is illustrated in Figure 23b.*



(**a**) (**b**)

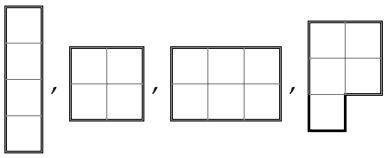
Figure 23. Target region and a tiling solution for Example 11: (**a**) the target region $\widehat{R}$, a checkerboard coloured '4-notched square shape' with a single 'hole' in the middle; (**b**) successful tiling of the target region $\widehat{R}$ using 25 copies of the straight tetromino, one copy of the straight trimino, one copy of the L-shaped trimino, and two copies of the P-shaped pentomino. The single optimal solution uses 25 copies of P_1^+ $(=P_1^-)$, one copy of P_2^+, one copy of P_3^+, and two copies of P_4^+. The pariominoes P_1^+, P_2^+, P_3^+, and P_4^+ are coloured blue, orange, yellow and red, respectively.

6.2.8. Case 8

We give an example of the multihedral polyomino type, where the application of colouring techniques does not split the full (uncoloured) polyomino tiling problem into multiple pariomino tiling subproblems, and there is *no* reduction in complexity.

Example 12. *Find all ways of tiling the 8 × 8 square with*

$$\{P_1, P_2, P_3, P_4\} = \left\{ \quad , \quad , \quad , \quad \right\},$$



all orientations permitted. We aim to tile R using $n_1 = 5$ copies of the straight tetromino, $n_2 = 7$ copies of the 2×2 square, $n_3 = 1$ copies of the 2×3 rectangle, and $n_4 = 2$ copies of the P-shaped pentomino. The ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE13`*, has $n = 549$ unknowns, $m = 68$ constraints, and $f = 482$ free variables.* `CPLEX` *found 157,288 solutions in 78 seconds.*

Using colouring techniques, we wish to tile the checkerboard coloured region $\widehat{R}$ shown in Figure 24a using five pariominoes from the set

$$\{P_1^+\} = \left\{ \quad \right\},$$

with parity $\pm p_1 = 0$, $(P_1^+ = P_1^-)$, and seven pariominoes from the set

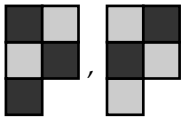
$$\{P_2^+\} = \left\{ \quad \right\},$$

with parities $\pm p_2 = 0$, $(P_2^+ = P_2^-)$, and one pariomino from the set

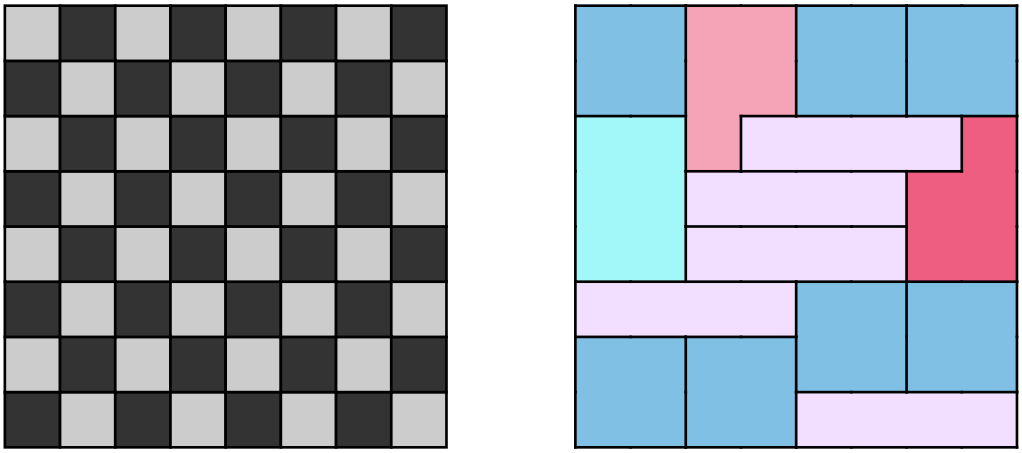
$$\{P_3^+\} = \left\{ \quad \right\},$$

with parities $\pm p_3 = 0$, $(P_3^+ = P_3^-)$, and two pariominoes from the set

$$\{P_4^+, P_4^-\} = \left\{ \quad , \quad \right\},$$



with parities $\pm p_4 = \pm 1$, all orientations permitted. As the parity of $\widehat{R}$ is zero, the only way to satisfy the parity equation (see Theorem 1) is to choose five copies of P_1^+ $(=P_1^-)$, seven copies of P_2^+ $(=P_2^-)$, one copy of P_3^+ $(=P_3^-)$, and one copy each of P_4^+ and P_4^-. We constructed the ILP file in this case using the program `LPMAKE_PARIOMINOES_FIGURE13`*. Solving this single subproblem in* `CPLEX` *with $(m, n, f) = (69, 549, 482)$ yielded 157,288 solutions in 85.3 seconds, which is an increase in the runtime of about 9.4% compared to the full (uncoloured) tiling problem. Thus, no potential speedup is observed. An optimal solution of this pariomino tiling subproblem is illustrated in Figure 24b.*



(**a**) (**b**)

Figure 24. Target region and a tiling solution for Example 12: (**a**) the target region $\widehat{R}$, a checkerboard coloured 8×8 square; (**b**) successful tiling of the target region $\widehat{R}$ using five copies of the straight tetromino, seven copies of the 2×2 square, one copy of the 2×3 rectangle, and two copies of the P-shaped pentomino. The single optimal solution uses five copies of P_1^+ ($=P_1^-$), seven copies of P_2^+ ($=P_2^-$), one copy of P_3^+ ($=P_3^-$), and one copy each of P_4^+ and P_4^-. The pariominoes P_1^+, P_2^+, P_3^+, P_4^+, and P_4^- are coloured violet, blue, light blue, red, and light red, respectively.

6.3. Tiling Examples Where We Seek a Single Solution

In this section, we are concerned with finding a single (optimal) solution to the ILP problems associated with tiling. Using checkerboard colouring techniques, we are interested in the pariomino tiling subproblem that yields a feasible solution in the least amount of time. If this is less than the time to find an optimal solution for the full (uncoloured) tiling problem, then we have reduced the complexity of this tiling problem.

We give a single example to illustrate this approach.

Example 13. *Find a single way of tiling the 60×64 rectangle with*

$$\{P_1, P_2\} = \left\{ \quad , \quad \right\},$$

all orientations permitted. We aim to tile R using $n_1 = 384$ copies of the V-shaped pentomino, and $n_2 = 384$ copies of the L-shaped pentomino. The associated ILP file, constructed with the program `LPMAKE_POLYOMINOES_FIGURE14`*, has $n = 43{,}144$ unknowns, $m = 3842$ constraints, and $f = 39{,}303$ free variables.* `CPLEX` *found an optimal solution in approximately 24 hours.*

Using colouring techniques, we wish to tile the 60×64 checkerboard coloured region $\widehat{R}$ using 384 pariominoes from the set

$$\{P_1^+, P_1^-\} = \left\{ \quad , \quad \right\},$$

with parities $\pm p_1 = \pm 1$, and 384 pariominoes from the set

$$\{P_2^+, P_2^-\} = \left\{ \quad , \quad \right\},$$

with parities $\pm p_2 = \pm 1$*, all orientations permitted. In the interests of space, we do not illustrate* $\widehat{R}$*; however, it is sufficient to note that the top left square is white. The parity equation (see Theorem 1) is given by*

$$p_1 a_1^+ + p_2 a_2^+ = (p + p_1 n_1 + p_2 n_2)/2, \quad 0 \le a_i^+ \le n_i, \quad i = 1, 2.$$

After noting that $p = 0$*, we obtain the linear Diophantine equation*

$$a_1^+ + a_2^+ = 384, \quad 0 \le a_1^+, a_2^+ \le 384,$$

with 385 solutions $\{(a_1^+, a_2^+) \ : \ a_1^+ = 384 - a_2^+, \quad a_2^+ = 0, 1, \ldots, 384\}$*. Due to time constraints, we only solved a single subproblem corresponding to* $(a_1^+, a_2^+) = (0, 384)$*, i.e., using 384 copies each of* P_1^- *and* P_2^+*. We constructed the ILP file in this case using the program* `LPMAKE_PARIOMINOES_FIGURE14`*. Using* `CPLEX` *yielded an optimal solution in 2.6 h, with* $(m, n, f) = (3842, 21572, 17732)$*. Thus the potential speedup when using the colouring technique is at least 9.2*$\times$*. The optimal solution is illustrated in Figure 25.*

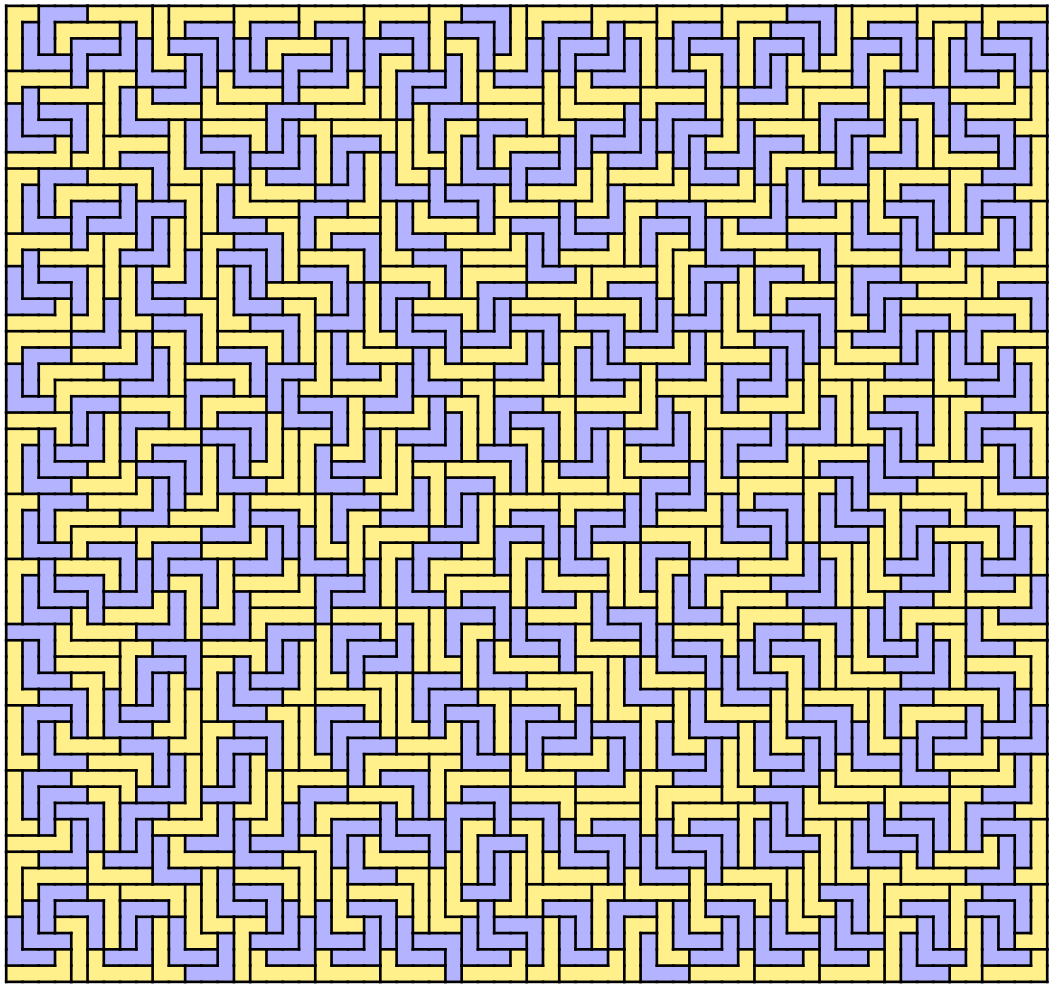


Figure 25. The tiling solution for Example 13 using 384 copies of the V-shaped pentomino, and 384 copies of the L-shaped pentomino. The optimal solution uses 384 copies of P_1^- (coloured blue) and 384 copies of P_2^+ (coloured yellow). The target region $\widehat{R}$ is a checkerboard coloured 60×64 rectangle with the top left square white.

7. Performance Analysis

We make some observations about the numerical results presented in Section 6; however, it is hard to make general conclusions because of the many situations that arise (see

Figure 12). Some problems do not split into subproblems, or do not do so usefully. We give a summary of the potential speedup results in Table 7.

Table 7. Summary of the potential speedups for Examples 1–13. Examples of questionable significance are marked with (∗).

Example	**1**	**2 (*)**	**3 (*)**	**4 (*)**	**5**	**6**	**7**
Problem Type	Case 1	Case 2	Case 2	Case 3	Case 3	Case 4	Case 5
Potential speedup	2.7×	<1.0×	<1.0×	2.5×	1.1×	<1.0×	1.6×
Example	**8**	**9**	**10**	**11**	**12**	**13**	-
Problem type	Case 5	Case 5	Case 5	Case 7	Case 8	-	-
Potential speedup	3.1×	1.2×	3.0×	4.9×	<1.0×	≥9.2×	-

Most problems do split into multiple, smaller subproblems. This has advantages if we are seeking a single solution, or all solutions because we can work in parallel. When a problem splits into multiple pariomino tiling subproblems, the experiments show that the `CPLEX` runtimes for the subproblems roughly correlate with the number of solutions found (see Example 8). Thus a subproblem with the greatest runtime is more likely to be small compared to the runtime of the full problem if the total number of solutions is uniformly distributed among the subproblems. We also saw that the runtimes for infeasible subproblems were much less than the runtimes for the feasible ones.

How a problem splits into subproblems depends on whether we are working with a multihedral or monohedral tiling problem. There are two different situations that yield a splitting of a polyomino tiling problem into multiple pariomino tiling subproblems. In the monohedral case, when tiling with n_i copies of a polyomino P_i where $P_i^+ \neq P_i^-$ and $p_i = 0$, we automatically obtain $n_i + 1$ subproblems to consider (see Example 1). The second situation that yields a splitting is in the multihedral case, where the parity equation in Theorem 1 has multiple solutions. However, we can easily create multihedral problems that do not split by using mostly tiles where $P_i^+ = P_i^-$ and hence $p_i = 0$. For example, Example 12 behaves essentially like a monohedral problem with two P-shaped pentominoes, as zero parity tiles play no part in the parity equation of Theorem 1. In the monohedral case when using tiles with non-zero parity, or where $P_i^+ = P_i^-$ (so $p_i = 0$), the parity equation is trivially solved, which implies there is no splitting of the polyomino tiling problem.

The numerical results for the multihedral problems uniformly show a reduction in complexity, as measured by the longest runtime to solve a pariomino tiling subproblem. Indeed, we could not find a tiling problem in this situation that did not yield a reduction in complexity. We obtained the greatest gains when the number of unknowns n and the number of free variables f for the pariomino tiling subproblem is small compared to the corresponding values of the polyomino tiling problem.

For monohedral problems, the results were less dramatic. When the full polyomino tiling problem is split into multiple pariomino tiling subproblems, we may (see Example 1) or may not (see Example 2) get a reduction in complexity. In the monohedral case, when we use copies of a tile with non-zero parity and apply checkerboard colouring techniques, this has little effect on the associated ILP problem. Thus, any gains in the potential speedup will probably be small. A case in point is Example 5 where both the full polyomino tiling problem and the single associated pariomino tiling subproblem had $n = 252$ unknowns and $f = 168$ free variables. We carefully constructed the monohedral Examples 3 and 4 to represent Cases 2 and 3 of the decision tree in Figure 12. Because these are contrived cases, they do not represent typical tiling problems.

Regarding tiling problems where we seek a single (optimal) solution with ILP, preliminary results (see Example 13) show excellent promise in reducing the problem complexity in the multihedral case. With many pariomino tiling subproblems, the likelihood increases that there will be a subproblem solved faster than it takes to solve the full (uncoloured) tiling problem. The advantage of this is that we might find tiling solutions to problems that are larger than we could solve without the colouring techniques.

8. Concluding Remarks

The novel parallelizable ILP approach presented in this article shows excellent promise in yielding a parallel speedup for solving large tiling problems. This is particularly true when we seek a single (optimal) solution.

A potential drawback of our approach when seeking all tiling solutions is that the number of pariomino tiling subproblems we need to solve is often very large. The current fastest supercomputers have millions of computer cores [66], although the typical user may only have access to thousands of cores. Thus, we expect that parallel computing techniques [67] can solve reasonably large tiling problems. However, the number of pariomino tiling subproblems can grow exponentially with the size of a tiling problem, and thus, for very large problems, finding *all* solutions is impractical. When we seek a few solutions, we can choose to solve only a fraction of the subproblems that we expect will yield solutions in a reasonable amount of time.

There are several possible directions for future research regarding the checkerboard colouring strategy presented in this article. For example, we could explore the computational benefits of adapting these techniques so that they apply to other types of problems with different geometries, such as polyiamonds and polyhexes [21]. Another type of problem with complex geometry that might benefit from the application of our colouring techniques is the 'Eternity puzzle' [68]. The aim of this puzzle is to tile a large dodecagon-shaped board with 209 distinct 'polydrafters', which are puzzle pieces constructed from 12 30-60-90 triangles placed edge-to-edge. Because of the fundamental intractability of this puzzle, only two solutions have been found since its introduction in 1999.

Another potential area for future research is computational methods for exact cover problems, for which Donald Knuth devised the 'Dancing Links' algorithm [50,69]. As tiling with polyominoes is an example of an exact cover problem, it might be possible to adapt our checkerboard colouring approach to that class of problems, which includes Sudoku [70], the N-queens problem [71], and edge-matching puzzles [72].

The traditional algorithmic approach to tiling with polyominoes uses backtracking, which is a recursive procedure in computer science for finding the solutions to a combinatorial search problem [50–52]. A different approach for tiling used evolutionary computation with a fitness function [26]. It would be interesting to compare the performance of our parallelizable ILP method to tiling with that of the backtracking and evolutionary computation approaches.

Author Contributions: Conceptualization, M.R.G.; methodology, M.R.G. and J.B.; software, J.B.; validation, M.R.G. and J.B.; formal analysis, M.R.G.; writing—original draft preparation, M.R.G.; writing—review and editing, M.R.G. and J.B.; funding acquisition, M.R.G. All authors have read and agreed to the published version of the manuscript.

Funding: The research of M.R.G. was supported by NSERC Discovery Grant # 400159.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Satisfiability Problem (SAT) Approach

If a tiling problem is formulated in the ILP format, assume there are m equations labeled E_1 through E_m, each corresponding to a cell in the target region to be covered, and k equations, labeled T_1 through T_k, each specifying the number of times we use a particular tile. Assume there are n variables, each associated with a placement of a particular tile in the region. This results in a (typically sparse) linear system of size $(m+k) \times n$, to which we must impose the additional constraints that the entries of the solution x are binary.

From a satisfiability problem (SAT) standpoint, the variables x are Boolean, so this constraint is automatically imposed. Each covering equation becomes a clause, joined by AND operators ($\cap$) in a single Boolean formula, whose first part is as follows:

$$(E_1 \cap E_2 \cap \cdots \cap E_m \cap \cdots).$$

For example, if a covering equation E_1 had the form:

$$x_1 + x_3 + x_7 + x_9 = 1,$$

then the corresponding Boolean clause combines these variables using the OR operator ($\cup$) is as follows:

$$(x_1 \cup x_3 \cup x_7 \cup x_9).$$

In fact, if the Boolean statement $(E_1 \cap E_2 \cap \cdots \cap E_m)$ is true, then we have guaranteed that we have covered all the cells in the region. However, in fact, SAT seeks *any and all* values of the variables that make the statement true; one such value has all the variables true, resulting in all the cells being covered, but many times. The correct formulation of the problem must impose the additional constraint that each tile can only be used a certain number of times, which guarantees that any solution will cover each cell exactly once. Thus, it is crucial to specify the number of times we use each tile.

If for a given tile, this limit is 1, then we could replace each $\cup$ in the Boolean clause by an exclusive OR operator, and we would be done. However, SAT has no exclusive OR operator. Moreover, this idea would fail if the limit was greater than 1.

Thus, for each tile, we need to impose a corresponding count constraint within the limited grammar allowed by SAT. If a tile count constraint is 1, and the tile has 100 possible placements in the region, this means we need to attach 100 clauses to the SAT formula for this single tile. Suppose we represent these tile placements by the variables x_1 through x_{100}, these clauses become:

$$(x_1 \cup \neg x_2 \cup \cdots \cup \neg x_{99} \cup \neg x_{100}) \cap (\neg x_1 \cup x_2 \cup \neg x_3 \cup \cdots \cup \neg x_{100}) \cap \cdots$$
$$\cap (\neg x_1 \cup \neg x_2 \cup \cdots \cup \neg x_{99} \cup x_{100}).$$

Of course, we will need to add such a string of clauses for every single tile. However, if a tile with 100 configurations has a larger count constraint, say 2, then we need to write $\binom{100}{2}$ such clauses and append these to our Boolean sentence. Things grow radically worse as the constraint and tile placement counts grow. In some of our problems, the number of tile placements of a single tile may run into the thousands, and the number of count constraints may be in the hundreds.

Thus, the SAT formulation of our problem would only be suitable for problems of very restricted size, and much smaller than we can handle using the ILP approach.

References

1. Castiglione, G.; Frosini, A.; Restivo, A.; Rinaldi, S. Enumeration of L-convex polyominoes by rows and columns. *Theoret. Comput. Sci.* **2005**, *347*, 336–352. http://doi.org/10.1016/j.tcs.2005.06.031.
2. Del Lungo, A.; Duchi, E.; Frosini, A.; Rinaldi, S. On the generation and enumeration of some classes of convex polyominoes. *Electron. J. Combin.* **2004**, *11*, 1–46. http://doi.org/10.37236/1813.
3. Delest, M.P.; Viennot, G. Algebraic languages and polyominoes enumeration. *Theoret. Comput. Sci.* **1984**, *34*, 169–206. http://doi.org/10.1016/0304-3975(84)90116-6.

4. Feretić, S. A perimeter enumeration of column-convex polyominoes. *Discrete Math. Theor. Comput. Sci.* **2007**, *9*, 57–83. http://doi.org/10.46298/dmtcs.390.
5. Golomb, S.; Klarner, D. Polyominoes. In *Handbook of Discrete and Computational Geometry*, 2nd ed.; Goodman, J., O'Rourke, J., Eds.; Chapman & Hall/CRC: Atlanta, GA, USA, 2004; pp. 331–352. http://doi.org/10.1201/9781420035315.ch15.
6. Goupil, A.; Cloutier, H.; Nouboud, F. Enumeration of polyominoes inscribed in a rectangle. *Discrete Appl. Math.* **2010**, *158*, 2014–2023. http://doi.org/10.1016/j.dam.2010.08.011.
7. Guttman, A. History and introduction to polygon models and polyominoes. In *Polygons, Polyominoes and Polycubes*; Lecture Notes in Physics; Guttman, A., Ed.; Springer: Dordrecht, The Netherlands, 2009; Volume 775. http://doi.org/10.1007/978-1-4020-9927-4_1.
8. Klarner, D.; Rivest, R. A procedure for improving the upper bound for the number of *n*-ominoes. *Can. J. Math.* **1973**, *25*, 585–602. http://doi.org/10.4153/CJM-1973-060-4.
9. Klarner, D.; Rivest, R. Asymptotic bounds for the number of convex *n*-ominoes. *Can. J. Math.* **1974**, *8*, 31–40. http://doi.org/10.1016/0012-365X(74)90107-1.
10. Leroux, P.; Rassart, E.; Robitaille, A. Enumeration of symmetry classes of convex polyominoes in the square lattice. *Adv. Appl. Math.* **1998**, *21*, 343–380. http://doi.org/10.1006/aama.1998.0601.
11. Bousquet-Mélou, M. Codage des polyominos convexes et équations pour l'énumération suivant l'aire. *Discrete Appl. Math.* **1994**, *48*, 21–43. http://doi.org/10.1016/0166-218X(92)00103-S.
12. Redelmeier, D. Counting polyominoes: Yet another attack. *Discrete Math.* **1981**, *36*, 191–203. http://doi.org/10.1016/0012-365X(81)90237-5.
13. Golomb, S. *Polyominoes*; Scribner: New York, NY, USA, 1965.
14. Golomb, S. *Polyominoes*, 2nd ed.; Princeton University Press: Princeton, NJ, USA, 1994. http://doi.org/10.1515/9780691215051.
15. Grünbaum, B.; Shephard, G. *Tilings and Patterns*; W.H. Freeman and Company: New York, NY, USA, 1987; Chapter 9.
16. Grünbaum, B.; Shephard, G. *Tilings and Patterns*, 2nd ed.; Dover Publications: New York, NY, USA, 2016.
17. Berger, R. The undecidability of the domino problem. *Mem. Am. Math. Soc.* **1966**, *66*, 1–72. http://doi.org/10.1090/memo/0066.
18. Fontaine, A.; Martin, G. Polymorphic Polyominoes. *Math. Mag.* **1984**, *57*, 275–283. http://doi.org/10.2307/2689601.
19. Golomb, S. Tiling with Sets of Polyominoes. *J. Comb. Theory* **1970**, *9*, 60–71. http://doi.org/10.1016/S0021-9800(70)80055-2.
20. Gruslys, V.; Leader, I.; Tan, T. Tiling with arbitrary tiles. *Proc. Lond. Math. Soc.* **2016**, *3*, 1019–1039.
21. Rhoads, G. Planar tilings by polyominoes, polyhexes, and polyiamonds. *J. Comput. Appl. Math.* **2005**, *174*, 329–353. http://doi.org/10.1016/j.cam.2004.05.002.
22. Schattschneider, D. Will It Tile? Try the Conway Criterion! *Math. Mag.* **1980**, *53*, 224–233.
23. Winslow, A. An optimal algorithm for tiling the plane with a translated polyomino. In Proceedings of the 26th International Symposium, ISAAC 2015, Nagoya, Japan, 9–11 December 2015; Volume 9472, pp. 3–13. http://doi.org/10.1007/978-3-662-48971-0_1.
24. Ardila, F.; Stanley, R. Tilings. *Math. Intell.* **2010**, *32*, 32–43. http://doi.org/10.1007/s00283-010-9160-9.
25. Ash, J.; Golomb, S. Tiling Deficient Rectangles with Trominoes. *Math. Mag.* **2004**, *77*, 46–55.
26. Ashlock, D.; Taylor, L. Evolving Polyomino Puzzles. In Proceedings of the 2016 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 327–334. http://doi.org/10.1109/CEC.2016.7743812.
27. Bodini, O. Tiling a Rectangle with Polyominoes. In Proceedings of the Conference of Discrete Models for Complex Systems, DMCS'03, Lyon, France, 16–19 June 2003; pp. 81–88. http://doi.org/10.46298/dmtcs.2313.
28. Golomb, S. Checker boards and polyominoes. *Am. Math. Mon.* **1954**, *61*, 675–682.
29. Golomb, S. Tiling with Polyominoes. *J. Comb. Theory* **1966**, *1*, 280–296. http://doi.org/10.1016/S0021-9800(66)80033-9.
30. Golomb, S. Polyominoes Which Tile Rectangles. *J. Combin. Theory Ser. A* **1989**, *51*, 117–124. http://doi.org/10.1016/0097-3165(89)90082-4.
31. Golomb, S. Tiling rectangles with polyominoes. *Math. Intell.* **1996**, *18*, 38–47.
32. Klarner, D. Packing a rectangle with congruent *n*-ominoes. *J. Comb. Theory* **1969**, *7*, 107–115. http://doi.org/10.1016/S0021-9800(69)80044-X.
33. Marshall, W. Packing Rectangles with Congruent Polyominoes. *J. Combin. Theory Ser. A* **1997**, *77*, 181–192.
34. Moore, C.; Robson, J. Hard Tiling Problems with Simple Tiles. *Discrete Comput. Geom.* **2001**, *26*, 573–590.
35. Rawsthorne, D. Tiling complexity of small *n*-ominoes ($n < 10$). *Discrete Math.* **1988**, *70*, 71–75. http://doi.org/10.1016/0012-365X(88)90081-7.
36. Reid, M. Tiling Rectangles and Half Strips with Congruent Polyominoes. *J. Combin. Theory Ser. A* **1997**, *80*, 106–123. http://doi.org/10.1006/jcta.1997.2788.
37. Reid, M. Klarner systems and tiling boxes with polyominoes. *J. Combin. Theory Ser. A* **2005**, *111*, 89–105.
38. Reid, M. Asymptotically Optimal Box Packing Theorems. *Electron. J. Combin.* **2008**, *15*, 1–19. http://doi.org/10.37236/802.
39. Reid, M. Many L-Shaped Polyominoes Have Odd Rectangular Packings. *Ann. Comb.* **2014**, *18*, 341–357.
40. Wagon, S. Fourteen Proofs of a Result About Tiling a Rectangle. *Am. Math. Mon.* **1987**, *94*, 601–617.
41. Korte, B.; Vygen, J. *Combinatorial Optimization: Theory and Algorithms*, 6th ed.; Algorithms and Combinatorics; Springer: Berlin/Heidelberg, Germany, 2018; Volume 21.

42. Takabatake, K.; Yanagisawa, K.; Akiyama, Y. Solving Generalized Polyomino Puzzles Using the Ising Model. *Entropy* **2022**, *24*, 354. http://doi.org/10.3390/e24030354.
43. Greenwood, G. Finding solutions to NP problems: philosophical differences between quantum and evolutionary search algorithms. In Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), Seoul, Korea, 27–30 May 2001; Volume 2, pp. 815–822. http://doi.org/10.1109/CEC.2001.934274.
44. Lilly, D. Complexity of solvable cases of the decision problem for predicate calculus. In Proceedings of the 19th Annual Symposium on Foundations of Computer Science, Ann Arbor, MI, USA, 16–18 October 1978; pp. 35–47. http://doi.org/10.1109/SFCS.1978.9.
45. Demaine, E.; Demaine, M. Jigsaw Puzzles, Edge Matching, and Polyomino Packing: Connections and Complexity. *Graphs Comb.* **2007**, *23*, 195–208. http://doi.org/10.1007/s00373-007-0713-4.
46. Garvie, M.R.; Burkardt, J. A new mathematical model for tiling finite regions of the plane with polyominoes. *Contrib. Discrete Math.* **2020**, *15*, 95–131. http://doi.org/10.11575/cdm.v15i2.62866.
47. Garvie, M.R.; Burkardt, J. A New Algorithm Based on Colouring Arguments for Identifying Impossible Polyomino Tiling Problems. *Algorithms* **2022**, *15*, 65. http://doi.org/10.3390/a15020065.
48. Busche, M. Solving Polyomino and Polycube Puzzles. Algorithms, Software, and Solutions. Available online: http://www.mattbusche.org/blog/article/polycube/ (accessed on 6 May 2022).
49. Fletcher, J. A program to solve the Pentomino problem by the recursive use of macros. *Commun. ACM* **1965**, *8*, 621–623. http://doi.org/10.1145/365628.365654.
50. Knuth, D. Dancing Links. In *Millennial Perspectives in Computer Science, Proceedings of the 1999 Oxford-Microsoft Symposium in honour of Professor Sir Antony Hoare, Oxford University, Oxford, UK, September 1999*; Cornerstones of Computing, Davies, J., Roscoe, B., Woodcock, J., Eds.; Red Globe Press: Oxford, UK, November 2000; p. 432.
51. Golomb, S.; Baumert, L. Backtrack Programming. *J. ACM* **1965**, *12*, 516–524. http://doi.org/10.1145/321296.321300.
52. Knuth, D. *The Art of Computer Programming, Fascicle 5: Mathematical Preliminaries Redux*; Introduction to Backtracking; Dancing Links; Addison-Wesley: Boston, MA, USA, 2020; Volume 4.
53. Gwee, B.; Lim, M. Polyominoes tiling by a genetic algorithm. *Comput. Optim. Appl.* **1996**, *6*, 273–291.
54. Conway, J.; Lagarias, J. Tiling with polyominoes and combinatorial group theory. *J. Combin. Theory Ser. A* **1990**, *53*, 183–208. http://doi.org/10.1016/0097-3165(90)90057-4.
55. Reid, M. Tile homotopy groups. *L'Enseign. Math.* **2003**, *49*, 123–155. http://doi.org/10.5169/seals-66684.
56. Thurston, W. Conway's Tiling Groups. *Am. Math. Mon.* **1990**, *97*, 757–773. http://doi.org/10.2307/2324578.
57. Aho, A.; Hopcroft, J.; Ullman, J. *The Design and Analysis of Computer Algorithms*, 1st ed.; Addison-Wesley: Reading, MA, USA, 1974.
58. Marek, V. *Introduction to Mathematics of Satisfiability*, 1st ed.; CRC Press: New York, 2009.
59. Biere, A.; Heule, M.; Maaren, H.v.; Walsh, T. *Handbook of Satisfiability*, 2nd ed.; Frontiers in Artificial Intelligence and Applications; IOS Press: Amsterdam, The Netherlands, 2021; Volume 336. http://doi.org/10.3233/FAIA336.
60. Kirillovs, J. Polyomino coloring and complex numbers. *Theoret. Comput. Sci.* **2008**, *400*, 100–112.
61. Garey, M.; Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman and Company: San Francisco, CA, USA, 1979.
62. Mordell, L. *Diophantine Equations*, 1st ed.; Pure and Applied Mathematics; Academic Press: London, UK, 1969; Volume 30.
63. Onn, S. Geometry, complexity, and combinatorics of permutation polytopes. *J. Combin. Theory Ser. A* **1993**, *64*, 31–49. http://doi.org/10.1016/0097-3165(93)90086-N.
64. Ziegler, G. *Lectures on Polytopes*, 7th ed.; Graduate Texts in Mathematics; Springer: New York, NY, USA, 1995; Volume 152. http://doi.org/10.1007/978-1-4613-8431-1.
65. Quinn, M. *Parallel Programming in C with MPI and OpenMP*, 1st ed.; McGraw-Hill: Boston, MA, USA, 2003.
66. Alsop, T. "Number of Computer Cores in the 10 Fastest Supercomputers in the World in 2020" From Statista—A Web Resource for Market and Consumer Data. Available online: https://www.statista.com/statistics/268280/number-of-computer-cores-in-selected-supercomputers-worldwide. (accessed on 6 May 2022).
67. Czech, Z.J. *Introduction to Parallel Computing*, 1st ed.; Cambridge University Press: Cambridge, UK, 2017.
68. Weisstein, E. "Eternity". From MathWorld—A Wolfram Web Resource. Available online: https://mathworld.wolfram.com/Eternity.html (accessed on 6 May 2022).
69. Knuth, D. Dancing Links. Available online: https://arxiv.org/abs/cs/0011047 (accessed on 6 May 2022).
70. Rosenhouse, J.; Taalman, L. *Taking Sudoku Seriously. The Math behind the World's Most Popular Pencil Puzzle*; Oxford University Press: Oxford, UK, 2011.
71. Bell, J.; Stevens, B. A survey of known results and research areas for n-queens. *Discrete Math.* **2009**, *309*, 1–31. http://doi.org/10.1016/j.disc.2007.12.043.
72. Kovalsky, S.; Glasner, D.; Basri, R. A Global Approach for Solving Edge-Matching Puzzles. *SIAM J. Imaging Sci.* **2015**, *8*, 916–938. http://doi.org/10.1137/140987869.