# Unfolding Boxes with Local Constraints

Long Qian ✉, Eric Wang ✉,
Bernardo Subercaseaux ✉, and Marijn J. H. Heule ✉

Carnegie Mellon University, Pittsburgh PA 15213, USA
{longq, ebwang, bsuberca, mheule}@andrew.cmu.edu

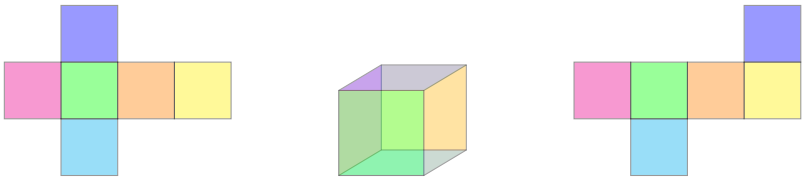
**Abstract.** We consider the problem of finding and enumerating polyominos that can be folded into multiple non-isomorphic boxes. While several computational approaches have been proposed, including SAT, randomized algorithms, and decision diagrams, none has been able to perform at scale. We argue that existing SAT encodings are hindered by the presence of global constraints (e.g., graph connectivity or acyclicity), which are generally hard to encode effectively and hard for solvers to reason about. In this work, we propose a new SAT-based approach that replaces these global constraints with simple local constraints that have substantially better propagation properties. Our approach dramatically improves the scalability of both computing and enumerating common box unfoldings: (i) while previous approaches could only find common unfoldings of two boxes up to area 88, ours easily scales beyond 150, and (ii) while previous approaches were only able to enumerate common unfoldings up to area 30, ours scales up to 60. This allows us to rule out 46, 54, and 58 as the smallest areas allowing a common unfolding of three boxes, thereby refuting a conjecture of Xu et al. (2017).

**Keywords:** Box folding · SAT encodings · Graph connectivity · Graph cyclicity · Local constraints.

## 1 Introduction

Folding two-dimensional surfaces into a three-dimensional structure is a fundamental problem in computational geometry, with many applications ranging from the arts (e.g., origami) to diverse fields of engineering (e.g., packaging, protein folding) [4]. Perhaps the simplest example, usually introduced to children, is that of folding a $1\times1\times1$ box (Figure 1b) from a *net* corresponding to a polyomino of area 6 (Figure 1a). Interestingly, as depicted in Figure 1c, multiple nets can fold into the same box, and the problem of enumerating all the nets that fold into a given $a\times b\times c$ box is already non-trivial (see Table 5).

An even more surprising fact, depicted in Figure 2, is that sometimes multiple non-isomorphic boxes can be obtained from the same net, simply by folding along different edges. These *common unfoldings* (also known as *common developments* [12]) allow for interesting engineering applications: a single two-dimensional piece of cardboard can be used for different types of boxes, depending on the dimensions of the object to be packed.



(a) Standard net. (b) 1×1×1 box. (c) Non-standard net.

Fig. 1: Illustration of the non-uniqueness of nets that fold into a box.

A body of work has focused in the particular case in which the nets are *polyominos*, and one can only fold along the edges of the unit squares forming the polyomino (cf. Figure 2) [6,8,10,11,12]. Nonetheless, several important questions remain open. To state them, let us define some notation.

For a positive integer $s$ representing a surface area, let $P(s)$ be the set of all triples $(a, b, c)$ with $a \leq b \leq c$ such that $s = 2(ab + ac + bc)$. In other words, $P(s)$ is the set of all possible integer dimensions of a box with surface area $s$. For example, $P(22) = \{(1, 1, 5), (1, 2, 3)\}$, and as shown in Figure 2, it turns out that both boxes in $P(22)$ can be folded from the same net of area 22. For a positive integer $k$, let $\Psi(k)$ be the smallest integer $s$ such that there is a subset $P' \subseteq P(s)$ with $|P'| = k$ and all boxes in $P'$ have a common unfolding. The example in Figure 2 shows that $\Psi(2) \leq 22$, and it can be easily checked that no smaller value of $s$ works. An impressive result by Shirakawa and Uehara is that $\Psi(3) \leq 532$, as they showed a common unfolding for boxes 7×8×14, 2×4×43, and 2×13×16. Xu et al. [12] conjectured $\Psi(3) = 46$, which is the smallest value for which $|P(s)| \geq 3$. For $k \geq 4$, it is not even known whether $\Psi(k)$ is finite. We define as well the opposite quantity, $\Delta(k)$, as the smallest integer $s$ such that there is a subset $P' \subseteq P(s)$ with $|P'| = k$ where **no** common unfolding for $P'$ exists. The work of Mitani and Uehara [6] shows that $\Delta(2) > 38$, and suggests $\Delta(2) > 88$. On the other hand, no upper bounds for $\Delta(2)$ are known.

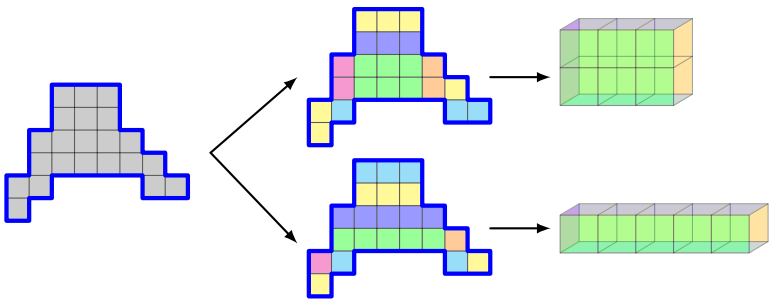
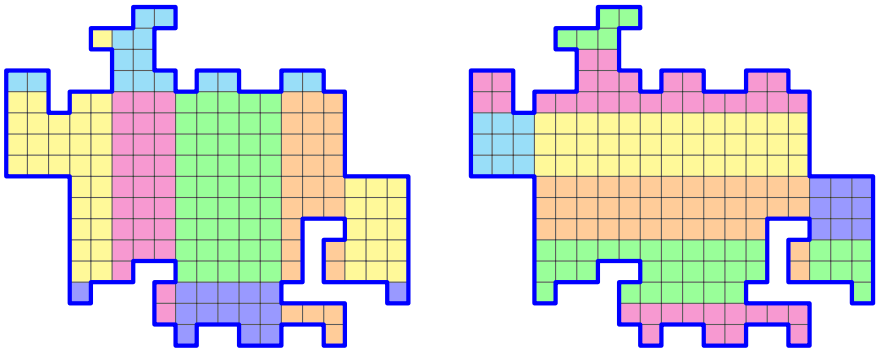


Fig. 2: Two non-isomorphic boxes that can be folded from the same net.



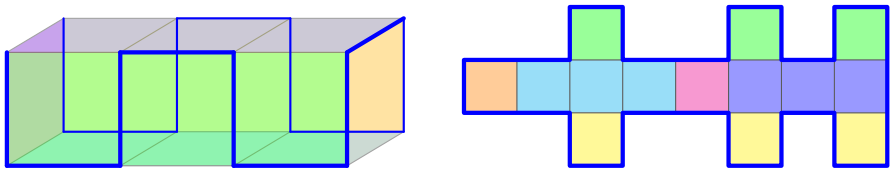
(a) 3×5×9 (b) 3×3×13

Fig. 3: A common unfolding of area 174.

In this work, we show that $\Psi(3) > 58$, that $\Delta(3) = 46$. We show as well that $\Delta(2) > 86$, and using heuristics we manage to compute solutions for certain very large areas (see Figure 3). More importantly, the SAT-based approach we developed to prove these results showcases a more general idea that might be applicable in a wide variety of contexts: global constraints, for which conflicts can be detected only after long propagation chains, can be approximated by local constraints for which conflicts are detected much earlier, leading to substantially better performance. To illustrate the power of our approach, let us present some brief elements in comparison with previous research. In 2011, Abel et al. [1] enumerated all common unfoldings for boxes 1×1×5 and 1×2×3 in about 10 hours [12], whereas our approach allows a complete enumeration in 2 minutes on a personal computer. Xu et al. [12] remarked that using the same approach for area 30 would have taken *"too huge memory even on a supercomputer"*, and their more efficient ZDD-based approach took only 10 days for area 30; ours takes 10 minutes. Moreover, Xu et al. conjectured that $\Psi(3) = 46$, saying *"However, the number of polygons of area 46 seems to be too huge to search"*. On a supercomputer [3], our approach took 3 hours to refute this conjecture.

**Code.** Our code and the instructions to reproduce our results are publicly available at https://github.com/LongQianQL/CADE30-BoxUnfoldings.

**Organization.** In Section 2, we give a high-level overview of our approach. Then, Section 3 introduces necessary properties that box unfoldings must satisfy, and that guide our encoding. In Section 4, we detail the differences with previous SAT encodings for a spanning tree (a natural subproblem for finding unfoldings) as we use local constraints that approximate both connectivity and acyclicty. Then, in Section 5, we show how to encode that a folding net, which our encoding keeps implicit, maps to two different boxes. In Section 6 we show how we break rotational symmetries of the problem. Section 8 discusses related work, and in particular, the limitations of previous SAT encodings. In Section 7 we present our experimental results.



(a) A $1 \times 1 \times 3$ box with cut edges. (b) The corresponding folding net.

Fig. 4: The correspondence between cuts in a box and its folding nets.

## 2 Overview of our approach

The first step toward finding common unfoldings of multiple boxes is finding unfoldings of a single box. As in previous work (e.g., [4,10,12]), we frame the search for unfoldings of a given box $B$ in terms of a search for *"cut edges"* in $B$ (see Figure 4) which after being physically cut, would allow to unfold the box into a flat net without overlaps. For instance, if one were to cut the blue edges of the $1\times1\times3$ box depicted in Figure 4a, and then proceed to unfold[1] the box, the result would be the net depicted in Figure 4b. More precisely, any net $N$ that folds into a box $B$ through a sequence $\gamma_1, \ldots, \gamma_k$ of folding motions in space can be obtained by cutting a subset of the edges of the unit-squares that compose the different faces of $B$, and then reversing the folding motions as $\gamma_k^{-1}, \ldots, \gamma_1^{-1}$ to obtain the net $N$.

Note, however, that not all sets of cut edges are *"valid"*, in the sense of allowing for an unfolding of the box into a flat net. For example, one can easily see that cutting a single edge of a box never allows for an unfolding, and furthermore in Section 3 we will describe a simple argument showing one needs at least 4 cut edges to unfold a box. Similarly, another requirement for a set of cut edges to be valid is not contain cycles; cutting along a cycle would separate the box into disconnected pieces!

Nonetheless, provided an efficient method to find valid sets of cut edges for a box $B$, we can search for common unfoldings of multiple boxes. Indeed, to find a net $N$ that folds into different boxes $B_1, \ldots, B_m$, we need to find a valid set $C_i$ of cut edges in each box $B_i$ such that the different sets $C_i$ are *"compatible"*, which intuitively means that for each pair $B_i, B_j$, the unit squares of $B_i$ can be mapped to those of $B_j$ so that two adjacent squares in $B_i$ without their common edge cut map to two adjacent squares in $B_j$ without their common edge cut. The details of this mapping presented in Section 5.

Now, to illustrate our general methodology, let us present a concrete result and a high-level sketch of how we obtain it.

**Theorem 1.** *No set of three non-isomorphic boxes of area* 58 *or less has a common unfolding. In other words,* $\Psi(3) > 58$.

[1] A precise mathematical definition of folding/unfolding turns out to be pretty intricate, using the entirety of Chapter 11 in the book of Demaine and O'Rourke [4]. We will thus mostly stick to intuition.

*Proof (Methodology).* Let us consider the particular case of ruling out area 46, since the other areas are analogous. The set of non-isomorphic box-dimensions for area 46 is $P(46) = \{(1,1,11),(1,2,7),(1,3,5)\}$. Any common unfolding of the corresponding boxes is, in particular, a common unfolding of boxes $B_1 := 1{\times}1{\times}11$ and $B_2 := 1{\times}2{\times}7$, so we focus first on constructing the set $S$ of all common unfoldings of these two boxes, where each common unfolding can be represented as a pair $(C_1, C_2)$, where $C_1$ (resp. $C_2$) is the set of edges to cut in $B_1$ (resp. $B_2$). However, instead of computing $S$ exactly, we compute a superset $S' \supseteq S$, that contains all common unfoldings of $B_1$ and $B_2$, but also potentially pairs $(C_1, C_2)$ where the sets of cut edges do not necessarily allow to unfold the boxes into a common net. The set $S'$ is obtained by enumerating all satisfying assignments of a CNF formula $\Phi(B_1, B_2)$, whose constraints will be detailed throughout the paper. Then, for each pair $(C_1, C_2) \in S'$, we do another SAT call to check whether it is possible to unfold box $B_3 := 1{\times}3{\times}5$ in a way that is compatible with $(C_1, C_2)$. Since none of the $|S'|$ calls is satisfiable, we can conclude that no common unfolding of $B_1, B_2, B_3$ exists. □

While a SAT encoding for the problem of finding common unfoldings of two (or more) boxes was already presented by Tadaki and Amano [10], our approach represents a significant improvement in allowing to search and enumerate common unfoldings for significantly larger dimensions. At a high-level, the key improvements of our encoding are:

1. When encoding the unfolding of a single box, that is, whether a set of cut edges is *"valid"*, we do not explicitly encode the net as [10], but rather properties that the set of cut edges must satisfy, and moreover, our encoding of these properties is a very efficient under-approximation, that replaces global constraints (i.e., the connectivity constraint of [10]) with local constraints.
2. When encoding a 2-box common unfolding, we do not use a net as intermediary, and instead encode the existence of a direct mapping between the unit-squares of box $B_1$ and those of $B_2$, that *"preserves"* the cut edges.
3. We exploit the symmetry of the boxes to reduce the search space. For example, in a $1{\times}1{\times}11$ there are several rotational symmetries that we break.

Before we detail the improvements, let us describe what it means for a set of cut edges to be *"valid"*, or more precisely, some necessary conditions for it.

## 3 Valid sets of cut edges

Let us first briefly describe 4 well-known (cf. [4, Ch. 21], [6]) necessary properties for a set of cut edges to be valid for a box $B$:

P1. **Connectivity:** The graph induced by the cut edges (taking the set of their endpoints as vertices) must be connected.
P2. **Cut corners:** The graph induced by the cut edges must touch all the 8 corners of the box $B$.
P3. **Acyclicity:** The graph induced by the cut edges must be acyclic.

P4. **Necessity:** For every set of four unit-squares $\{s_1, s_2, s_3, s_4\}$ that forms a $2\times2$ square on $B$, it cannot be the case that *exactly one* edge between these unit-squares is cut, as such cuts are not necessary [6, Lemma 1].

We suggest the reader inspects Figure 4a to check that these constraints are satisfied, and to try to obtain some insight into their necessity. Intuitively, P1 is justified by the fact that the cut edges unfold into the boundary of the net, as exemplified in Figure 4, and that boundary is connected. P2, on the other hand, can be justified by noting that every non-cut edge connecting adjacent unit-squares $s_1, s_2$ in a box $B$ will remain an edge connecting two adjacent squares in the net $N$ that folds into $B$. Thus, if none of the three edges incident to a corner are cut, then the three squares incident to that corner will remain adjacent in the net $N$, which is a contradiction since in a polyomino there cannot be three pairwise adjacent squares, as illustrated in Figure 5. P3 is justified by the fact that if the graph induced by the cut edges contained a cycle $C$, then the unit squares inside $C$ would be disconnected from the rest when unfolding the box $B$. Finally, P4 is intuitively justified by noticing that if exactly one such edge is cut, then this edge can always be "glued" back without affecting the underlying unfolding, a rigorous proof of P4 can also be found in earlier works [6, Lemma 1].

To represent whether an edge $\{s_1, s_2\}$ is cut or not, we can simply use a boolean variable $e_{s_1,s_2}$ that is true if and only if $\{s_1, s_2\}$ is *not cut*. Then, P2 can be trivially encoded by 8 clauses of the form $(\overline{e}_{s_1,s_2} \vee \overline{e}_{s_1,s_3} \vee \overline{e}_{s_2,s_3})$, where $s_1, s_2, s_3$ are the three adjacent squares on a corner of the box. Similarly, for each $2\times2$ square $\{s_1, s_2, s_3, s_4\}$ on $B$ (with $\{s_1, s_4\}$ non-adjacent), $P_4$ can be encoded by 4 clauses of the form $(e_{s_1,s_2} \wedge e_{s_1,s_3} \wedge e_{s_2,s_4} \rightarrow e_{s_3,s_4})$ The difficulty, however, arises when encoding properties P1 and P3, which are *"non-local"* properties, and despite a body of research, remain challenging to encode without resulting in either a large number of clauses or poor propagation properties [5,13].

Our approach replaces these constraints by a set of local constraints that intuitively pursue a similar goal as P1 and P3: ensuring that the graph of cut edges has sufficiently many edges (P1) without having too many (P3). Concretely:

1. To force cutting a significant number of edges, we leverage the work of Tadaki and Amano [10], and assign orientations to each square of the box, which then allows for enforcing consistency constraints between adjacent squares whose common edge is not cut. To satisfy those constraints, a significant number of edges must be cut.

Fig. 5: Illustration of the necessity of the cut-corners property.

2. In contrast to the encoding in [10], which forbid cutting too many edges by enforcing the connectivity of the resulting 2D net, explicit in their encoding, we use a fully novel idea: assigning orientations to the edges of the box, and then forbidding a small number of local directed patterns that every valid unfolding can avoid, but most disconnected nets contain. Intuitively, since disconnected nets correspond to cycles of cut edges, our encoding attempts to prevent such cycles.

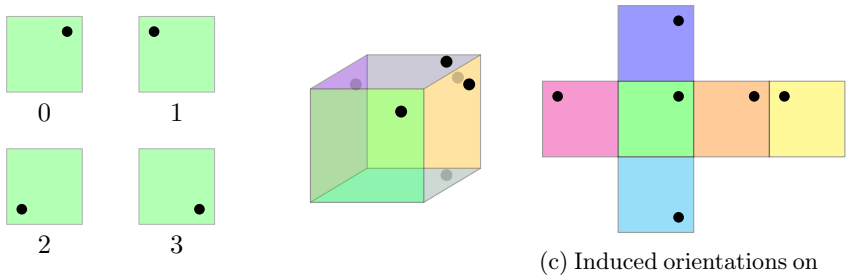
## 4 Local constraints

As described in Section 3, our goal is to impose constraints that ensure that sufficiently many, but not too many, variables $e_{s_1,s_2}$ (representing that the edge $\{s_1, s_2\}$ is not cut) are set to true. We achieve these goals independently: we enforce cutting edges using *"square-orientation constraints"* in a similar (albeit with important differences) way to Tadaki and Amano [10], and use a fully novel approach, based on *"edge directions"* to forbid too many edges from being cut.

### 4.1 Square orientations

Inspired by [10], we think of each square in a box $B$ as having an *"orientation"*, that intuitively represents whether, on an unfolding of the net, the square would be rotated by 0°, 90°, 180°, or 270°. As Figure 6a indicates, each square is labeled with a dot, which then induces an orientation value on the square once it is unfolded onto the 2D plane. Naturally, this requires a labeling of squares on the box with such dots. Any consistent labeling will work. We adopt the convention that the dot is labeled at the corner that is diagonally the furthest away from the origin when the box is placed in the positive octant. Equivalently, one can directly extend the labeling shown in Figure 6b to general boxes. Once such a labeling has been fixed, it is clear that any unfolding of the box will induce an orientation assignment $o : B \to \{1, 2, 3, 4\}$.

Importantly, any orientation assignment $o\colon B \to \{1, 2, 3, 4\}$ induced by a valid unfolding will necessarily preserve *"relative orientations"* between connected




(a) Square orientations (b) Box labels. (c) Induced orientations on net, $r_{\text{green}}(\text{blue}) = 3$.

Fig. 6: Square orientations, labels, and relative orientations.

squares. For a pair of connected squares $s_1, s_2$, define the relative orientation of $s_2$ with respect to $s_1$, $r_{s_1}(s_2)$, to be the orientation of $s_2$ if $s_1$ was rotated to have orientation 0 (Figure 6c). Note that $r_{s_1}(s_2)$ *only* depends on the canonical labeling chosen for the box, and in particular does not depend on potential unfoldings. Since unfolding is an orientation-preserving geometric transformation, for any edge $e = \{s_1, s_2\}$ that is *not cut*, the relative orientations between $s_1, s_2$ must remain invariant. Thereby necessarily implying $o(s_2) = o(s_1) + r_{s_1}(s_2)$ (and vice versa), where addition is carried out in $\mathbb{Z}_4$. In fact, these are the only constraints that we enforce in our encoding for square orientations. The constraints are:

- Variables $o_{s,d}$ for $s \in B, d \in \{1, 2, 3, 4\}$ encoding a function $o : B \to \{1, 2, 3, 4\}$ representing the orientation values. For this to be a well-defined function, we have the following constraints for all $s \in B$.

$$\sum_{d=1}^{4} o_{s,d} = 1$$

- For each edge $e = \{s_1, s_2\}$ in $B$ that is not cut (i.e. $e_{s_1,s_2}$ is true), it must be the case that $o(s_2) = o(s_1) + r_{s_1}(s_2)$ (and vice versa). This is encoded as follows for every $d \in \{1, 2, 3, 4\}$.

$$\left(e_{s_1,s_2} \wedge o_{s_1,d} \to o_{s_2,r_{s_1}(s_2)+d}\right) \wedge \left(e_{s_1,s_2} \wedge o_{s_2,d} \to o_{s_1,r_{s_2}(s_1)+d}\right)$$

### 4.2 Edge directions

Let $G_B$ be the graph with the squares of box $B$ as vertices, and where neighboring squares have a graph edge if and only if their common geometrical edge is not cut. Recall now that forbidding cycles of cut edges is equivalent to making the graph $G_B$ connected, which is our goal. The SAT encoding in [10] encodes graph connectivity by choosing one vertex $s^\star \in V(G_B)$ as the source of a *Breadth First Search* (BFS), and then encoding that every vertex is reached by that BFS.

Concretely, variables $t_{v,k}$ represent that vertex $v$ is reached on step $k$ or earlier of the BFS, with $k$ ranging up to $|V(G_B)| - 1$ in the worst case. Then, the encoding consists of:

- The source vertex $s^\star$ is reached at step 0, enforced by unit clause $t_{s^\star,0}$.
- Each vertex is reached at some step, enforced by the formula

$$\bigwedge_{v \in V(G_B)} \bigvee_{k=0}^{|V(G_B)|-1} t_{v,k}.$$

- Let $N(v)$ denote the set of four neighbors of $v \in V(G_B)$. A vertex $v$ is reached at step at most $k$ if and only if one of its neighbors (or itself) was reached at step at most $k - 1$:

$$\bigwedge_{k=1}^{|V(G_B)|-1} \bigwedge_{v \in V(G_B)} \left( t_{v,k} \leftrightarrow \bigvee_{u \in N(v)} (t_{u,k-1} \wedge e_{u,v}) \right).$$

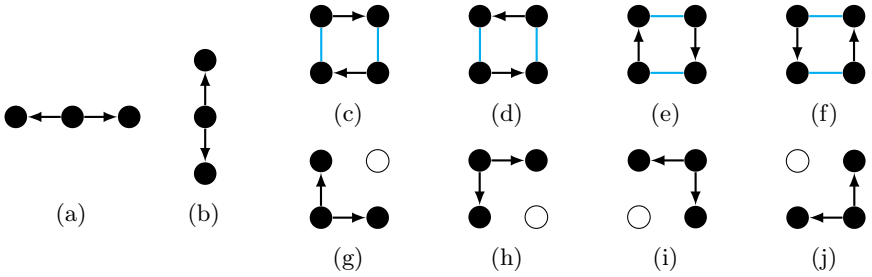



Fig. 7: The set $\mathcal{F}$ of forbidden subgraphs. The blue edges represent that the graph belongs to $\mathcal{F}$ regardless of the orientation of its blue edges. For the forbidden patterns 7g to 7j, the white node must not be present in the graph. Blocking the patterns 7a to 7b requires one binary clause per pattern. Blocking the patterns 7c to 7f requires two ternary clauses. Finally, blocking the patterns 7g to 7j requires two ternary clauses. The ternary clauses make use of the observation that it is impossible to preserve exactly 3 of the 4 edges.

As a result, the number of variables and clauses in their encoding is quadratic in $|V(G_B)|$, the number of squares of the box. It is possible to upper-bound $k \leq T$ for some $T \leq |V(G_B)| - 1$ to improve performance at the cost of potentially missing solutions, thus such bounds cannot be used if one wants to enumerate all solutions. Table 1 illustrates this for a typical sub-problem (Section 6) between boxes $1{\times}1{\times}7, 1{\times}3{\times}3$.

We approximate connectivity using a constant number of clauses per square. First, for each neighboring pair of squares $(s_1, s_2)$, we create two variables, $d_{s_1,s_2}$ and $d_{s_2,s_1}$ representing that the preserved edge $\{s_1, s_2\}$ will be directed from $s_1$ toward $s_2$ (or from $s_2$ toward $s_1$, respectively). We have clauses $(\overline{d}_{s_1,s_2} \vee \overline{d}_{s_2,s_1})$ to prevent both directions per preserved edge. If an edge $\{s_1, s_2\}$ is preserved, it has at least one direction, this is enforced by $(e_{s_1,s_2} \to d_{s_1,s_2} \vee d_{s_2,s_1})$. Finally, if the edge $\{s_1, s_2\}$ has a direction, it is necessarily preserved. This is enforced by $d_{s_1,s_2} \to e_{s_1,s_2}$ and $d_{s_2,s_1} \to e_{s_1,s_2}$.

Now, we encode that a special vertex $s^\star$ is a *unique sink* by enforcing that (i) it has outdegree 0 according to the edge directions, so $\bigwedge_{u \in N(s^\star)} \overline{d}_{s^\star,u}$, and (ii) every other vertex has outdegree at least 1:

$$\bigwedge_{v \in V(G_B) \setminus \{s^\star\}} \bigvee_{u \in N(v)} d_{v,u}$$

Note that for us, $s^\star$ is a *"sink"*, instead of a *"source"*. To approximate further that our edge directions correspond to a *reverse BFS* from $s^\star$, we forbid all local patterns depicted in Figure 7 as subgraphs. Since this forbids a constant number of possibilities around each vertex, it totals $O(|V(G_B)|)$ clauses. While this is insufficient in theory to guarantee connectivity, it almost always results in connected nets in practice. We prove in the extended arXiv version of this

paper [7] that these constraints are sound, meaning that every valid unfolding must satisfy them.

Table 1: Runtimes with BFS/forbidding local patterns at $|V(G_B)| = 30$.

| $T = 15$ | $T = 16$ | $T = 17$ | $T = 18$ | $T = 19$ | $T = 20$ | **Local patterns** |
|---|---|---|---|---|---|---|
| 79.66 s | 240.53 s | 385.56 s | 504.43 s | 947.83 s | 1823.88 s | **36.51 s** |

## 5 Common unfoldings of two boxes

An important feature of our approach is the *implicit* representation of nets via their corresponding cut edges, in contrast to *explicitly* representing them as subsets of the 2D plane. Utilizing such implicit representations, the search for a net $N$ that folds into different boxes $B_1, \cdots, B_m$ naturally reduces down to the search for cut edges $C_1, \cdots, C_m$ that are "compatible". Importantly, such constraints are *intrinsic* to the boxes and cut edges $(B_i, C_i)$ and do not concern the explicit 2D representation of nets. This allows for much more compact and efficient encodings compared to earlier work [10], which uses explicit representations.

Figure 8 shows how the search for a common net can be done explicitly by enforcing equivalence on the resulting 2D unfoldings, which is costly and introduces additional auxiliary variables. In contrast, we achieve this by encoding an equivalence mapping $M\colon B_2 \to B_1$ directly on the level of boxes, and thereby alleviating the need to consider the nets explicitly.

Figure 9 depicts the constraints imposed on $M\colon B_2 \to B_1$. If a square $s_1' \in B_2$ is mapped to $s_1 \in B_1$ *and* the edge $e = \{s_1, s_2\}$ is not cut, then it must necessarily be the case that the corresponding neighbor of $s_1'$, in this case $s_2'$, maps to $s_2$

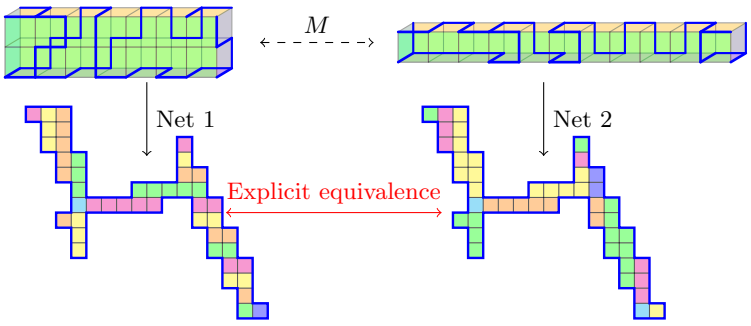



Fig. 8: Equivalence of nets of boxes 1×2×7 and 1×1×11, explicitly via nets (red) and implicitly via cut edges ($M$).

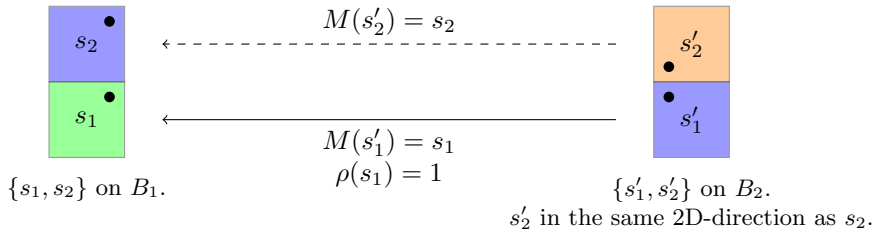



Fig. 9: Constraints on $M$ with relative change in orientation $\rho$.

and the edge $e = \{s'_1, s'_2\}$ is *not cut* in box 2. Furthermore, notice that $s'_2$ is the unique neighbor of $s'_1$ in the same direction as $s_2$, and can be computed from $o(s'_1) - o(s_1)$, the relative change in orientations. These constraints are encoded as follows.

- Mapping variables $m_{s,s'}$ for $s' \in B_2, s \in B_1$ that encode a bijection $M\colon B_2 \to B_1$. For each $s' \in B_2$ we have the constraint

$$\sum_{s \in B_1} m_{s,s'} = 1$$

- Auxiliary variables $\rho_{s,d}$ for $s \in B_1, d \in \{1,2,3,4\}$ that encode a function $\rho\colon B_1 \to [4]$ indicating the change in orientation between $s$ and the unique $s' \in B_2$ where $M(s') = s$. I.e. $o(s') = o(s) + \rho(s)$ holds. This is encoded via the following where $s' \in B_2, s \in B_1$ and $d, r \in \{1,2,3,4\}$.

$$m_{s,s'} \wedge o_{s',d+r} \wedge o_{s,d} \to \rho_{s,r}$$

- Constraints that enforce the preservation of relative positions as shown in Figure 9. Let $\{s_1, s_2\}$ be an edge in $B_1$, $s'_1 \in B_2$ and $d, r \in \{1,2,3,4\}$ be such that $\rho(s_1) = r$ and $d$ indicates the 2D-direction of $s_2$. Let $s'_2$ be the unique neighbor of $s'_1$ in the same direction as $s_2$. The constraints are:

$$m_{s_1,s'_1} \wedge \rho_{s_1,r} \wedge e_{s_1,s_2} \to m_{s_2,s'_2}$$
$$m_{s_1,s'_1} \wedge \rho_{s_1,r} \wedge e_{s_1,s_2} \to e_{s'_1,s'_2}$$
$$m_{s_1,s'_1} \wedge \rho_{s_1,r} \wedge e_{s'_1,s'_2} \to e_{s_1,s_2}$$

  where the latter two constraints encode that the edge $e = \{s_1, s_2\}$ is cut if and only if $e' = \{s'_1, s'_2\}$ is cut. Only the first type of constraint is necessary. The other two types help with performance.
- Finally, we write $\text{EQUIV}_{B_1,B_2}$ to denote the overall encoding for equivalence obtained by taking the conjunction of all constraints above. Equivalence of more than 2 boxes can easily be encoded by enforcing $\text{EQUIV}_{B_1,B_n}$ for all $n \geq 2$.

## 6 Symmetry breaking

Geometrically, if a box $B$ unfolds into a net $N$, then the symmetries of $B$ can certainly be unfolded into the same net $N$. To efficiently enumerate all unfoldings, it is important to avoid the computation of repeated solutions by breaking these symmetries. In this section, we describe how symmetry breaking is carried out and show experimentally that this greatly improves the performance.

As a first step, notice that a geometric (2D) rotation of a net only affects the orientations of squares and preserves the edges. Indeed, constraints on the orientation of squares introduced in Section 4 only enforce relative equality, and therefore if $o\colon B \to \{1,2,3,4\}$ gives a satisfying assignment of orientations, then a constant shift (e.g. $o'(s) = o(s) + 1$ for all $s$) will also be satisfying and corresponds to a 2D rotation of the underlying net. Thus, we may without loss of generality pick a distinguished square $\hat{s} \in B$ and enforce $o(\hat{s}) = 0$ with a unit clause.

To further break symmetries when searching for common unfoldings (of two boxes $B_1, B_2$), consider the image of pairs of squares under the equivalence mapping $M$ (Section 5) of a common unfolding. First, fix a pair of squares $(s_1, s_2)$ on $B_1$. Since $M$ is a bijection, some pair of squares $(s_1', s_2')$ on $B_2$ satisfies $(M(s_1'), M(s_2')) = (s_1, s_2)$, as illustrated in Figure 10. If $Q \in \mathrm{Sym}(B_2)$ is a symmetry of $B_2$, then certainly the same unfolding can be carried out on $Q(B_2)$ (as $Q$ is a symmetry), inducing a common unfolding between $Q(B_2)$ and $B_1$ where the pair $(Q^{-1}(s_1'), Q^{-1}(s_2'))$ now maps to $(s_1, s_2)$. Therefore, if $P_2$ denotes the set of all pairs of $B_2$ up to symmetry, then the search for common unfoldings can be reduced to the cases $(M(s_1'), M(s_2')) = (s_1, s_2)$ for every pair $(s_1', s_2') \in P_2$. Furthermore, if $(s_1, s_2)$ are chosen on $B_1$ such that $s_1$ is in the orbit of $s_2$ under symmetries of $B_1$, then it suffices to consider the pairs in $P_2$ as unordered, thereby reducing the number of cases by a factor of 2. Table 2 depicts the number of such pairs relative to the total number of pairs $\binom{\mathrm{Area}}{2}$. As shown, the number of pairs up to symmetry is relatively small compared to the number of all pairs.

Finally, when the dimensions of $B_1$ is of the form $1\times1\times n$, we always pick the pair $(s_1, s_2)$ to be the $1\times1$ faces. In addition to them being symmetric, $B_1$ has full rotational symmetry about these faces and therefore the orientation of $s_1$

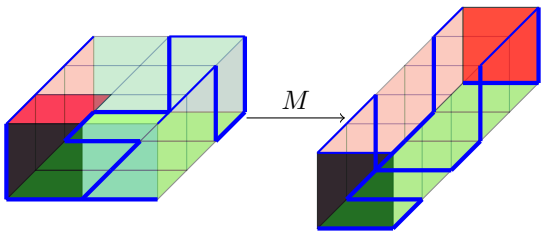



Fig. 10: Image of pairs of squares under $M$, mapping the (black, red) pair on $B_2$ to the (black, red) pair on $B_1$.

Table 2: Number of pairs

| Area | Dimensions | | #Pairs | Total% |
|---|---|---|---|---|
| 22 | 1×1×5 | 1×2×3 | 45 | 9.74% |
| 30 | 1×1×7 | 1×3×3 | 47 | 5.40% |
| 34 | 1×1×8 | 1×2×5 | 97 | 8.64% |
| 38 | 1×1×9 | 1×3×4 | 120 | 8.53% |
| 42 | 1×1×10 | 2×3×3 | 77 | 4.47% |
| 46 | 1×1×11 | 1×2×7 | 169 | 8.16% |
| 54 | 1×1×13 | 1×3×6 | 231 | 8.07% |
| 58 | 1×1×14 | 1×2×9 | 261 | 7.89% |

can be altered by applying such rotations *without rotating the underlying net*. Consequently, this implies that we may always assume without loss of generality that $o(s_1) = 0$, in addition to fixing the orientation of a square $s'_1 \in B_2$. To summarize, when enumerating common unfoldings between two boxes $B_1, B_2$, the following symmetry-breaking procedure is carried out.

1. Fix a pair of squares $(s_1, s_2)$ on $B_1$ that belong to the same equivalence class under symmetry.
2. If $B_1$ is of the form $1\times1\times n$, choose $(s_1, s_2)$ to be the $1\times1$ faces and enforce $o(s_1) = 0$ using a unit clause.
3. Let $P_2$ be the set of (unordered) pairs of squares on $B_2$ unique up to symmetry. For each pair $(s'_1, s'_2) \in P_2$, encode $M(s'_1) = s_1, M(s'_2) = s_2, o(s'_1) = 0$ using 3 unit clauses.
4. Solve the corresponding sub-problem for each pair $(s'_1, s'_2) \in P_2$. Note that since each sub-problem is independent, they can be run in parallel.

## 7 Experimental results

Using the encodings given in Sections 4 and 5 without symmetry-breaking, we were able to compute unfoldings between all pairs of boxes with equal area up to 86 (Table 3), thereby establishing $\Delta(2) > 86$. As indicated in Table 3, most of such unfoldings with high area were previously unknown. The first missing

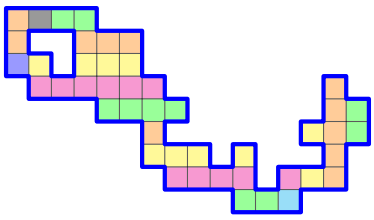


Fig. 11: Overlapping net (left) and touching net (right) of 1×1×11.

Table 3: Existence of common unfoldings.

| Area | Dimensions | | First | Area | Dimensions | | First | Area | Dimensions | | First |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 22 | 1×1×5 | 1×2×3 | [6] | 78 | 1×1×19 | 1×3×9 | ✓ | 108 | 1×4×10 | 3×4×6 | ✓ |
| 30 | 1×1×7 | 1×3×3 | [6] | 78 | 1×1×19 | 1×4×7 | ✓ | 110 | 1×3×13 | 1×6×7 | ✓ |
| 34 | 1×1×8 | 1×2×5 | [6] | 78 | 1×1×19 | 3×3×5 | ✓ | 110 | 1×3×13 | 3×5×5 | ✓ |
| 38 | 1×1×9 | 1×3×4 | [6] | 78 | 1×3×9 | 1×4×7 | ✓ | 112 | 2×2×13 | 2×4×8 | ✓ |
| 40 | 1×2×6 | 2×2×4 | [10] | 78 | 1×3×9 | 3×3×5 | ✓ | 112 | 2×3×10 | 4×4×5 | ✓ |
| 42 | 1×1×10 | 2×3×3 | ✓ | 78 | 1×4×7 | 3×3×5 | ✓ | 112 | 1×2×18 | 4×4×5 | ✓ |
| 46 | 1×1×11 | 1×3×5 | [6] | 80 | 2×2×9 | 3×4×4 | ✓ | 118 | 1×4×11 | 1×5×9 | ✓ |
| 46 | 1×1×11 | 1×2×7 | ✓ | 82 | 1×1×20 | 1×2×13 | ✓ | 118 | 1×3×14 | 1×5×9 | ✓ |
| 46 | 1×2×7 | 1×3×5 | [6] | 82 | 1×1×20 | 1×5×6 | ✓ | 118 | 1×4×11 | 2×5×7 | ✓ |
| 48 | 1×4×4 | 2×2×5 | ✓ | 82 | 1×1×20 | 2×3×7 | ✓ | 118 | 1×3×14 | 2×5×7 | ✓ |
| 54 | 1×1×13 | 3×3×3 | [6] | 82 | 1×2×13 | 1×5×6 | ✓ | 120 | 2×2×14 | 2×6×6 | ✓ |
| 54 | 1×1×13 | 1×3×6 | [6] | 82 | 1×2×13 | 2×3×7 | ✓ | 124 | 1×6×8 | 2×4×9 | ✓ |
| 54 | 1×3×6 | 3×3×3 | [6] | 82 | 1×5×6 | 2×3×7 | ✓ | 126 | 1×3×15 | 3×5×6 | ✓ |
| 58 | 1×1×14 | 1×2×9 | ✓ | 86 | 1×1×21 | 1×3×10 | ✓ | 126 | 1×7×7 | 3×5×6 | ✓ |
| 58 | 1×1×14 | 1×4×5 | [6] | 86 | 1×2×14 | 1×4×8 | ✓ | 126 | 1×7×7 | 3×3×9 | ✓ |
| 58 | 1×2×9 | 1×4×5 | ✓ | 86 | 1×2×14 | 2×4×6 | ✓ | 126 | 3×3×9 | 3×5×6 | ✓ |
| 62 | 1×1×15 | 1×3×7 | ✓ | 88 | 1×4×8 | 2×2×10 | [6] | 128 | 1×4×12 | 4×4×6 | ✓ |
| 62 | 1×1×15 | 2×3×5 | ✓ | 88 | 1×4×8 | 2×4×6 | ✓ | 128 | 2×2×15 | 4×4×6 | ✓ |
| 62 | 1×3×7 | 2×3×5 | [6] | 88 | 2×2×10 | 2×4×6 | [6] | 132 | 2×3×12 | 2×5×8 | ✓ |
| 64 | 1×2×10 | 2×2×7 | [6] | 90 | 2×5×5 | 3×3×6 | ✓ | 136 | 2×4×10 | 2×6×7 | ✓ |
| 64 | 1×2×10 | 2×4×4 | ✓ | 94 | 1×5×7 | 3×4×5 | ✓ | 136 | 2×6×7 | 3×4×8 | ✓ |
| 64 | 2×2×7 | 2×4×4 | [6] | 94 | 1×3×11 | 3×4×5 | ✓ | 138 | 1×4×13 | 1×6×9 | ✓ |
| 66 | 1×1×16 | 3×3×4 | ✓ | 94 | 1×3×11 | 1×5×7 | ✓ | 138 | 1×6×9 | 3×3×10 | ✓ |
| 70 | 1×1×17 | 1×2×11 | ✓ | 96 | 1×6×6 | 2×2×11 | ✓ | 142 | 1×5×11 | 1×7×8 | ✓ |
| 70 | 1×1×17 | 1×3×8 | ✓ | 96 | 1×6×6 | 4×4×4 | ✓ | 142 | 2×3×13 | 3×5×7 | ✓ |
| 70 | 1×1×17 | 1×5×5 | [6] | 100 | 1×2×16 | 2×4×7 | ✓ | 142 | 1×5×11 | 2×3×13 | ✓ |
| 70 | 1×2×11 | 1×3×8 | [6] | 102 | 2×3×9 | 3×3×7 | ✓ | 142 | 1×7×8 | 2×3×13 | ✓ |
| 70 | 1×2×11 | 1×5×5 | ✓ | 102 | 1×3×12 | 2×3×9 | ✓ | 142 | 1×5×11 | 3×5×7 | ✓ |
| 70 | 1×3×8 | 1×5×5 | ✓ | 104 | 2×2×12 | 2×5×6 | ✓ | 144 | 2×2×17 | 3×6×6 | ✓ |
| 72 | 2×2×8 | 2×3×6 | ✓ | 106 | 1×1×26 | 1×2×17 | ✓ | 148 | 1×4×14 | 4×5×6 | ✓ |
| 76 | 1×2×12 | 2×4×5 | ✓ | 106 | 1×2×17 | 1×5×8 | ✓ | 174 | 3×3×13 | 3×5×9 | ✓ |

entry is the pair $1\times2\times14, 2\times2\times10$ at area 88. It is worth noting that common unfoldings can always be scaled up by subdividing squares, e.g. a common unfolding of $1\times1\times5, 1\times2\times3$ directly implies the existence of a common unfolding for $2\times2\times10, 2\times4\times6$ by dividing each square into 4 sub-squares [6].

With the symmetry-breaking procedure described in Section 6, we were able to completely enumerate all solutions for the pairs of boxes shown in Table 4. Local computations were conducted using a standard laptop, computations labeled with "cluster" were conducted on a supercomputer [3] running 64 SAT solvers in parallel. Solutions were obtained by enumerating all solutions for each fixed pair of squares on the second box (Section 6), using an allsat variant of CaDiCaL [2], available at https://github.com/jreeves3/allsat-cadical. Through these computations, we

Table 4: Exhaustive enumeration of solutions. The columns labeled $|S|$, $|T|$, and $|O|$ show the number of simple, touching, and overlapping solutions, respectively.

| Area | Dimensions | | #SAT | Unique | $\lvert S\rvert$ | $\lvert T\rvert$ | $\lvert O\rvert$ | Time | First |
|---|---|---|---|---|---|---|---|---|---|
| 22 | 1×1×5 | 1×2×3 | 3 942 | 2 303 | 2 263 | 27 | 13 | 2 mins (local) | [1] |
| 30 | 1×1×7 | 1×3×3 | 1 790 | 1 080 | 1 070 | 10 | 0 | 10 mins (local) | [12] |
| 34 | 1×1×8 | 1×2×5 | 131 054 | 35 700 | 35 675 | 22 | 3 | 1 hour (local) | ✓ |
| 38 | 1×1×9 | 1×3×4 | 5 854 | 4 509 | 4 469 | 36 | 4 | 6 hrs (local) | ✓ |
| 42 | 1×1×10 | 2×3×3 | 128 558 | 111 948 | 111 387 | 559 | 2 | 1 day (local) | ✓ |
| 46 | 1×1×11 | 1×2×7 | 16 928 | 15 236 | 14 971 | 16 | 249 | 3 hrs (PSC) | ✓ |
| 54 | 1×1×13 | 1×3×6 | 56 087 | 51 884 | 51 836 | 48 | 0 | 1 day (PSC) | ✓ |
| 58 | 1×1×14 | 1×2×9 | 2 150 373 | 551 935 | 551 923 | 7 | 5 | 2 days (PSC) | ✓ |

Table 5: Exhaustive enumeration of 1×1×$n$ nets

| Dimensions | #SAT | Unique | $\lvert S\rvert$ | $\lvert T\rvert$ | $\lvert O\rvert$ | Time |
|---|---|---|---|---|---|---|
| 1×1×1 | 384 | 11 | 11 | 0 | 0 | 0.05 s |
| 1×1×2 | 12 124 | 723 | 723 | 0 | 0 | 1.79 s |
| 1×1×3 | 240 304 | 15 061 | 14 978 | 79 | 4 | 77.78 s |
| 1×1×4 | 3 708 380 | 231 310 | 228 547 | 2 603 | 160 | 9 hrs |

found unfoldings with diameter close to the surface area (we include examples in the extended arXiv version [7]) making it difficult to compute using a BFS-style encoding for connectivity.

In addition to standard nets (simply connected, no overlaps), our encoding is also capable of finding non-standard unfoldings, which we classify into "touching" and "overlapping". Both are nets in the sense that one can obtain them by unfolding a box, however they are non-standard as the resulting shape is not simply-connected. The gray square in the overlapping net shown in Figure 11 is an overlapping green and yellow square. Touching nets can always fold into the original box if cuts were allowed, e.g. the touching net shown in Figure 11 can fold into 1×1×11 if a cut is made along the red edge.

Beyond common unfoldings, we also enumerated all nets for boxes of the form 1×1×$n$ for $n \leq 4$ (Table 5). To the best of our knowledge, the number of such nets was previously unknown except for $n = 1$. Such computations were carried out locally on a standard laptop.

## 8 Related work

The algorithmic search for common unfoldings of non-isomorphic boxes has been investigated in many earlier works [1,6,8,10,11,12] using various techniques. One body of work is the complete enumeration of all unfoldings [1,12], computing all possible unfoldings between two boxes. The best previous result was due to

[12], providing a complete enumeration of all unfoldings at area 30. Our efficient approach allows us to completely enumerate all unfoldings up to surface area 58, providing complete enumerations for many pairs of boxes that were previously unknown (Table 4). Furthermore, solutions of area 22 were enumerated by [1] taking 10 hours, whereas our approach takes 2 minutes. Solutions of area 30 were enumerated by [12] taking 10 days using a computer with 128 GB memory, our approach only takes 10 minutes using a standard laptop.

We were also able to find many new common unfoldings between boxes that were previously unknown. The best lower-bound for $\Delta(2)$ from earlier works was 40 and already at area 42 it was not known if boxes of dimensions $1\times1\times10, 2\times3\times3$ admit a common unfolding. Using our approach, we compute all possible common unfoldings between all boxes up to surface area 86, more than doubling the previous best bound.

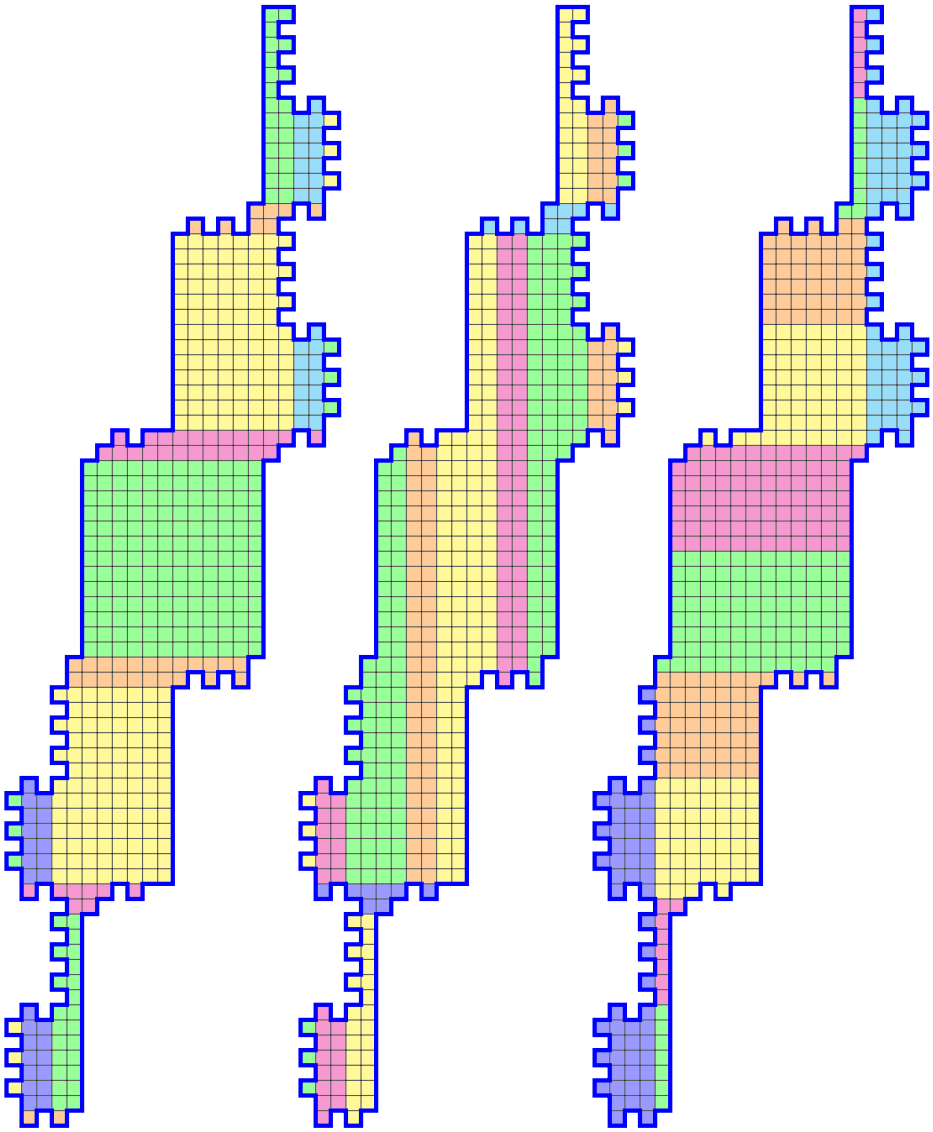
The use of SAT solvers to find common unfoldings has also been explored in earlier work [10], our approach has two fundamental improvements over the previous approach. The first being the use of implicit equivalences which alleviates the need to consider nets as subsets of the 2D plane (Section 5). This significantly improves the efficiency of the encodings by eliminating the auxiliary variables needed to represent the nets in 2D. Secondly, the use of local constraints (Section 4) allows us to efficiently detect connectedness in the cut edges of boxes, and furthermore this does not place an *a priori* restriction on the radius of the underlying net. Thereby allowing us to completely enumerate *all solutions* between boxes and establishing $\Psi(3) > 58$ rather than only finding solutions with a bounded radius. One can theoretically enumerate all solutions by using the surface area as the upper-bound, but this is too costly on performance [10].

## 9 Concluding remarks

We have presented a new SAT-based approach for finding and enumerating polyominos that can be folded into multiple non-isomorphic boxes, outperforming previous approaches. We have introduced a technique that could be applicable to searching for other combinatorial objects: approximating constraints that are hard to encode, or result in large numbers of clauses, with simpler local constraints, and then discard the solutions that do not satisfy the original constraints.

Further supporting the correctness of our encoding, we were able to reproduce the common unfolding of 3 boxes at area 532 [8] as shown in Figure 12. This was obtained by specifying the cut edges for one of the boxes ($2\times13\times16$) with unit clauses and using a SAT solver to solve for the remaining variables.

The quest for the smallest area allowing a common unfolding of three boxes remains open, and part of our future work. Another promising direction is to formally verify our work, in the line of [9], given how intricate the encoding is and the delicate arguments for its completeness. We suspect that a particularly challenging aspect of that verification will be to formally define *(un)foldings* (cf. Demaine and O'Rourke [4, Ch. 15]).



(a) 2×13×16 (b) 2×4×43 (c) 7×8×14

Fig. 12: Common unfolding of 3 boxes at area 532 [8].

**Acknowledgements.** We thank Joseph Reeves for implementing solution enumeration in CaDiCaL, which was crucial to performing the experiments. This work was supported by the U. S. National Science Foundation under grant DMS-2434625.

**Disclosure of Interests.** The authors have no competing interests to declare that are relevant to the content of this article.

## References


1. Abel, Z., Demaine, E.D., Demaine, M.L., Matsui, H., Rote, G., Uehara, R.: Common developments of several different orthogonal boxes. In: Proceedings of the 23rd Annual Canadian Conference on Computational Geometry, Toronto, Ontario, Canada, August 10-12, 2011 (2011), http://www.cccg.ca/proceedings/2011/papers/paper49.pdf
2. Biere, A., Faller, T., Fazekas, K., Fleury, M., Froleyks, N., Pollitt, F.: CaDiCaL 2.0. In: Gurfinkel, A., Ganesh, V. (eds.) Computer Aided Verification - 36th International Conference, CAV 2024, Montreal, QC, Canada, July 24-27, 2024, Proceedings, Part I. Lecture Notes in Computer Science, vol. 14681, pp. 133–152. Springer (2024). https://doi.org/10.1007/978-3-031-65627-9_7
3. Brown, S.T., Buitrago, P., Hanna, E., Sanielevici, S., Scibek, R., Nystrom, N.A.: Bridges-2: A Platform for Rapidly-Evolving and Data Intensive Research. In: Practice and Experience in Advanced Research Computing. PEARC '21, Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3437359.3465593
4. Demaine, E., O'Rourke, J.: Geometric Folding Algorithms: Linkages, Origami, Polyhedra. Cambridge University Press (2007)
5. Gebser, M., Janhunen, T., Rintanen, J.: SAT Modulo Graphs: Acyclicity. In: Fermé, E., Leite, J. (eds.) Logics in Artificial Intelligence. pp. 137–151. Springer International Publishing, Cham (2014)
6. Mitani, J., Uehara, R.: Polygons folding to plural incongruent orthogonal boxes. In: The 20th Canadian Conference on Computational Geometry (CCCG'08) (2008)
7. Qian, L., Wang, E., Subercaseaux, B., Heule, M.J.H.: Unfolding boxes with local constraints (2025), https://arxiv.org/abs/2506.01079
8. Shirakawa, T., Uehara, R.: Common developments of three different orthogonal boxes. In: The 24th Canadian Conference on Computational Geometry (CCCG'12) (2012)
9. Subercaseaux, B., Nawrocki, W., Gallicchio, J., Codel, C., Carneiro, M., Heule, M.J.H.: Formal Verification of the Empty Hexagon Number. In: 15th International Conference on Interactive Theorem Proving (ITP 2024). Leibniz International Proceedings in Informatics (LIPIcs), vol. 309, pp. 35:1–35:19. Dagstuhl, Germany (2024). https://doi.org/10.4230/LIPIcs.ITP.2024.35
10. Tadaki, R., Amano, K.: Search for developments of a box having multiple ways of folding by SAT solver (2020), https://arxiv.org/abs/2005.02645
11. Uehara, R.: A survey and recent results about common developments of two or more boxes. In: Origami $^6$: proceedings of the sixth international meeting on origami science, mathematics, and education (2015)
12. Xu, D., Horiyama, T., Shirakawa, T., Uehara, R.: Common developments of three incongruent boxes of area 30. Computational Geometry **64**, 1–12 (2017). https://doi.org/10.1016/j.comgeo.2017.03.001
13. Zhou, N.F., Wang, R., Yap, R.H.C.: A Comparison of SAT Encodings for Acyclicity of Directed Graphs. In: 26th International Conference on Theory and Applications of Satisfiability Testing (SAT 2023). Leibniz International Proceedings in Informatics (LIPIcs), vol. 271, pp. 30:1–30:9. Dagstuhl, Germany (2023). https://doi.org/10.4230/LIPIcs.SAT.2023.30