A Partition-Based Heuristic for Translational Box Covering

BEN ENGLAND
University of Massachusetts Lowell
Department of Computer Science
One University Avenue, Lowell, MA
USA

KAREN DANIELS
University of Massachusetts Lowell
Department of Computer Science
One University Avenue, Lowell, MA
USA

Abstract: Geometric covering problems are found in application domains such as sensor coverage and repairing of materials. This paper considers covering problems in which a target shape to be covered is specified and a set of covering shapes is supplied. The goal is to find translational positions of the covering shapes that allow them to collectively cover the target. All the shapes are orthotopes (convex boxes). The approach provides the best known results for translational, two-dimensional orthotope covering and the first results in three and four dimensions. Experiments suggest why the strategy succeeds in certain regions of the problem domain in spite of the NP-hardness of translational orthotope covering. This leads to a novel, dimension-independent measure that significantly improves the speed of the method. Orthotopes can form enclosures for nonconvex shapes, so progress for orthotope covering is beneficial in a more general context.

Key–Words: Geometric covering, orthotopes, Lagrangian relaxation, simulated annealing

1 Introduction

The goal of this research is to increase our insight into translational geometric covering problems. Such problems arise in a variety of areas ranging from sensor coverage to manufacturing and repair work.

Problem Definition. Borrowing from definitions in [12], let \( Q = \{ Q_j \} \) be a set of \( N \) closed, convex subsets of \( \mathbb{R}^d \), and let \( P \) be a closed compact subset of \( \mathbb{R}^d \). The set \( \{ \tau_j(Q_j) \} \) is called a covering of \( P \) if there exist translations \( \tau_j \) such that: \( P \subseteq \bigcup_{j=1}^{N} \tau_j(Q_j) \). \( P \) is the target shape and the \( Q_j \)’s are covering shapes. Rotations are not permitted in this version of the problem. Our goal is to find a single solution to the translational box covering problem, not all possible solutions. We restrict all shapes to be orthotopes, i.e. axis-aligned convex boxes. Since orthotopes can be used as enclosures for more complicated shapes, advances in orthotope covering can improve the state-of-the-art for more general covering problems. An example of a polygonal covering problem for large sensors is illustrated in Figure 1. Polygonal problems are also generated by repair work in which a collection of pieces of material must collectively patch a hole. If the material is anisotropic, so that it has different properties in different directions, then the patching problem can be restricted to translational motion.

Related Work. The literature on packing and covering is vast. See Toth [20] for a survey of results on covering the plane with congruent convex shapes. Many covering problems are worst-case NP-hard. Even set covering is generally NP-complete [9, 3]. Papers such as [14, 8] have shown that various two-dimensional polygonal translational geometric covering problems are NP-complete. Daniels et al. showed in [4] that the nonconvex, two-dimensional, polygonal, translational generalization of the translational box covering problem is NP-complete using a reduction from PARTITION; this proof is extended for \( d \)-dimensional translational orthotope covering in [7]. Cheeseman et al. describe in [2] how some NP-complete problems can be summarized by order parameter(s) and that the hard instances occur at a critical value of such
a parameter. This motivates the instance characterization in our paper. Approximation algorithms and optimization-based approaches have been developed for some geometric covering problems [15, 13, 1].

The literature on packing and covering contains some useful orthotope feasibility tests. In particular, Groemer [12] states that for a set of orthotopes with edge lengths at most 1, if \( \sum_{j=1}^{N} v(Q_j) \geq (s+1)^d - 1 \), where \( v(\cdot) \) denotes volume, then \( Q \) is sufficient to cover an orthotope with equal sides of length \( s \). Toth [20] proves that for orthotopes that are homothetic copies of the target, then \( 2^d \) of these shapes are needed to cover the target. Toth describes the Hadwiger-Levi covering problem that is a precursor of the corner-constrained covering problems identified in this paper. Box covering work by Lassak is tangentially related [16].

The current paper evolved from work done by Grinde and Daniels on a set covering problem [11, 10] which involved placing small pattern pieces into unoccupied spaces in between large pattern pieces in layouts for apparel manufacturing. The goal was to create two-dimensional, compact layouts. For each unoccupied space, multiple groups of small pieces were found that fit into the space. An optimization model using the technique of Lagrangian relaxation [19] attempted to maximize the number of placed pieces by selecting one group of small pieces to place into each unoccupied space; the result was a Lagrangian heuristic that successfully solved the apparel problem.

Subsequently, Daniels observed that the set covering problem from [11, 10] could be interpreted so as to form the core of a covering heuristic for tackling the polygonal version of the orthotope problem stated above. The key was to break \( P \) (the shape to be covered) into parts, interpret each covering shape \( Q_j \) as the equivalent of a region of unoccupied space, find groups of parts that fit into each \( Q_j \), and then use the Lagrangian heuristic to try to cover as many parts of \( P \) as possible. A triangular decomposition of \( P \) was used. If the Lagrangian heuristic did not cover all parts of \( P \), then one triangular part of \( P \) was subdivided, the group structure was expanded, and the Lagrangian heuristic was invoked again. While promising results were obtained in this manner by Daniels et al.[5] for arbitrarily nonconvex two-dimensional shapes, the running time was dominated by the group formation process, which relied on the Minkowski sum\(^2\) (for “fitting”) and other polygon set operations. Consequently, this approach has been limited to instances with only a small number of covering shapes (approximately \( \frac{1}{2} \) dozen, as depicted in Figure 1). The expense of set operations on polygons in three and higher dimensions has prevented this strategy from being generalized beyond two dimensions. Neacsu and Daniels worked on generalizing from two-dimensional polygons to spline shapes in [17].

**Overview.** This paper treats the orthotope problem stated above. The strategy is described in Section 2. It uses a modified version of the iterative Lagrangian heuristic approach described above and used in [5]; there are five important differences. First, because the geometric entities are boxes, the cost of geometric operations is greatly reduced. Second, a uniform refinement strategy is used rather than triangular subdivision and, as a natural consequence, instead of subdividing a single part of \( P \) during each iteration, multiple parts of \( P \) are subdivided at the same time.

Third, dimension is an input to the orthotope heuristic, which has been tested in two, three, and four dimensions. Experimental highlights are given in Section 3. This provides the first published results on box covering in dimensions higher than two. In some cases, the new heuristic is able to quickly find covers using as many as 50 covering shapes. In two dimensions, the orthotope solver is at least two orders of magnitude faster than the two-dimensional nonconvex heuristic of [5].

Fourth, orthotope covering is NP-complete in the worst case, so in Section 4 we explore what makes a translational orthotope covering instance hard in the context of a uniform orthotope refinement scheme. We offer a novel dimension-independent measure that is based on maximizing the volume of the intersection of the covering orthotopes with the target orthotope. Tests in two, three, and four dimensions involving more than 6,000 randomly generated examples suggest that this measure is a good predictor of coverage. This measure, in turn, is used to greatly improve the efficiency of the orthotope heuristic.

Fifth, the group generation process and geometric operations are fast enough to make the Lagrangian heuristic the computational bottleneck, even though the Lagrangian heuristic is very fast. So, in Section 5 we investigate issues relative to its performance, such as a 1-OPT\(^3\) search procedure embedded inside it, as well as the crucial question of whether or not coverage improves monotonically across successive calls to the Lagrangian heuristic. We found that 1-OPT performs better than a 2-OPT, a randomized 1-OPT, or a simulated annealing strategy. We also determined that 1-

\(^2\)The Minkowski sum of two sets \( A \) and \( B \) is \( \{a+b|a \in A, b \in B\} \).

\(^3\)A 1-OPT heuristic is typically a local improvement strategy. In our context, it seeks and performs the group swap for the container that best improves the objective function and repeats this process until no improving swap is found.
OPT was not behaving as a purely local improvement heuristic. This knowledge allowed us to further improve our the performance of the Lagrangian heuristic by adding 1-OPT preprocessing in a more global sense. Although we have no formal proof of monotonicity across applications of the Lagrangian heuristic, our experiments exhibit this influential property. Section 6 draws conclusions and suggests directions for future work.

2 Orthotope Covering Heuristic

Pseudocode for the OrthotopeCover( ) heuristic is given in Algorithm 1. As introduced in Section 1, \( P \) is the target to be covered, \( Q \) is the set of \( N \) covering shapes and \( d \) is the dimension. This is a uniform partitioning heuristic. Array \( \text{slices} \), initialized in lines 1-3, specifies how many parts are used in each dimension. The total number of parts is denoted by \( \varphi \) and \( \delta \) is a generic part. The \( \text{groups} \) structure is a set of part groups generated in each pass of the \text{repeat} loop in lines 5-25, and \( \text{cSet} \), initialized in line 14, holds the indices of the part groups that each covering orthotope \( Q \) can cover. The \text{result} structure in line 24 is a proposed solution that assigns a single position to each covering orthotope.

As noted in Section 1, volume is denoted by \( v(\cdot) \). Additional definitions related to volume are:

- \( v(Q_j, P) \) is the effective volume of \( Q_j \) with respect to \( P \). This equals the maximum possible volume of \( (\tau_j(Q_j)) \cap P \). For orthotopes this is easily calculated from the dimensions of \( Q_j \) and \( P \), unlike the nonconvex polygonal case.

- \( v'(Q_j, \delta) \) is quantized volume of \( Q_j \) given \( \delta \). This is \( \prod_{i=1}^{d} \left[ \frac{s_i(Q_j)}{s_i(\delta)} \right] s_i(\delta) \), where \( s_i(\cdot) \) denotes (non-0) size in a dimension. Quantized \( Q_j \) of this size is denoted by \( Q'_j \). \( Q' = \sum_{j=1}^{N} Q'_j \).

- \( v'(Q_j, \delta, P) \) is quantized effective volume of \( Q_j \) given \( \delta \). This equals the maximum possible volume of \( (\tau_j(Q'_j)) \cap P \).

Inside the \text{for all} parts loop of lines 15-21, a group is generated by positioning \( Q_j \) at each corner of each part \( p \) and intersecting \( Q_j \) with \( P \). \text{LGC.Cover(\cdot)} in line 24 executes the Lagrangian set covering heuristic of [11, 10, 5] in the orthotope context, which addresses the set covering problem described in Section 1. Details relative to the performance of the internal structure of \text{LGC.Cover(\cdot)} are deferred until Section 5.

OrthotopeCover( ) terminates if coverage reaches 100\%, or the heuristic makes \( d+1 \) passes without significant improvement in the result, or if the problem is determined to be infeasible. Standard feasibility tests from the literature [12] plus simple tests such as one based on total volume of \( Q \) versus \( P \) are employed but not shown in the pseudocode. The effective quantized volume test \( \sum v'(Q_j, \delta, P) \geq v(P) \) rests on a maximum overlap proposition that shows that if an orthotope covering exists, then one exists in which each covering shape has maximum overlap with \( P \) [7]. Note that this test can only be employed because the stated problem goal is to find a \text{single} solution, if one exists. Also, this test does not apply to general polygonal shapes, whereas it does apply to orthotopes.

3 Experimental Highlights

Here we briefly discuss several of the best results that we have achieved; Sections 4 and 5 explore the reasons for this success and present additional experiments using large, randomized test suites to support the experimental conjectures.

Software. The software for this project was imple-
mented using GNU C++ with the C++ Standard Template Library. This results in an efficient, portable implementation that can fully exploit whatever hardware is available to it. The implementation used in [7] and much of this paper is freely available from the web site at [6, 18]. No use of LEDA, CGAL, or other computational geometry libraries is required because of the orthotope nature of the problem domain. It is shown in [7] that if a cover exists for a translational orthotope instance, then a cover also exists for the scaled instance in which \( P \) is a unit orthotope. Thus, without loss of generality, our problem instances in this paper use a unit orthotope for \( P \).

**Hardware.** Three different hardware configurations are employed to investigate different aspects of our heuristic:

- 450 MHz CPU Sun SPARC Ultra 60\(^{TM}\) with 512 MB memory (used in [5])
- 1 GHz Pentium\(^{TM}\) 4 CPU with \( \frac{1}{2} \) GB memory (used in [7])
- 3 GHz 64-bit Intel Pentium\(^{TM}\) D CPU with 2 GB memory.

These different profiles affect running time of the heuristic but do not alter its other characteristics, such as the number of calls to LGC\_Cover( ).

**Two-Dimensional Validation Experiment.** A small set of 20 two-dimensional problem instances was run on hardware configuration 1 to compare the performance of this implementation to the earlier two-dimensional, general, translational geometric covering problem solver documented in [5]. The number of covering shapes ranged from two to six. OrthotopeCover( ) found a cover for all the test cases in which the general solver finds a cover. The thickness of the cover is given by Eqn. 1 below; this is a standard measure from the covering literature [20]. The thickness ranges from 1.1 to 4.

\[
\frac{\sum_{j=1}^{N} \left( \tau_j(Q_j) \cap P \right)}{v(P)}.
\]

As expected, the simplification to orthotopes allowed OrthotopeCover( ) to outperform the general covering heuristic. The OrthotopeCover( ) implementation was always at least two orders of magnitude faster than the general solver. As noted earlier, the relative speed of OrthotopeCover( ) is due primarily to its simpler geometric operations (e.g., avoidance of the Minkowski sum).

**Results in Three and Four Dimensions.** From our experiments on hardware configuration 2, our heuristic found covers with as many as 40 covering shapes with total cover shape volume as small as 1.03, using as many as 16K parts. Figure 2 (left) is a two-dimensional view\(^4\) of a three-dimensional partial cover for 20 covering shapes. Only a small region of \( P \) was not covered. In this example, \( P \) was partitioned into 4096 parts, 47,603,138 groups were tried, and 99.78\% of the parts were covered. This was achieved using just under one minute of execution time in LGC\_Cover( ), which dominated the total running time of the heuristic. Figure 2 (right) used 40 covering shapes. \( P \) was partitioned into 4096 parts, 2,344,940 groups were tried, and 100\% of the parts were covered. Again, this was achieved using just under one minute of execution time in LGC\_Cover( ).

**Table 1:** Experimental highlights for feasible covering problems. Cover shape volume is denoted by \( csv \). Edge ratio 4 applies to all cover shapes. Time is for LGC\_Cover( ) and is in seconds on hardware profile 3.

<table>
<thead>
<tr>
<th>( d )</th>
<th>( N )</th>
<th>( csv )</th>
<th>( \varphi )</th>
<th># groups</th>
<th>time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>50</td>
<td>1.7</td>
<td>4,096</td>
<td>8,610,188</td>
<td>56.97</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>1.8</td>
<td>4,096</td>
<td>2,939,730</td>
<td>21.71</td>
</tr>
<tr>
<td>3</td>
<td>50</td>
<td>1.9</td>
<td>4,096</td>
<td>3,293,814</td>
<td>21.67</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>2.0</td>
<td>4,096</td>
<td>1,574,748</td>
<td>7.20</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>2.2</td>
<td>16,384</td>
<td>6,783,913</td>
<td>234.50</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>2.3</td>
<td>16,384</td>
<td>6,112,860</td>
<td>209.80</td>
</tr>
<tr>
<td>4</td>
<td>2.4</td>
<td>8,192</td>
<td>3,591,600</td>
<td>46.27</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.5</td>
<td>8,192</td>
<td>3,445,338</td>
<td>39.55</td>
<td></td>
</tr>
</tbody>
</table>

In subsequent experiments using hardware configuration 3, our heuristic found covers using as many as 50 covering shapes. Table 1 summarizes these results. In three dimensions the cover shape volume \((\sum v(Q_j))\) ranges from 1.7 to 2.0; in four dimensions it is between 2.2 and 2.5. The edge ratio is the maxi-

---

\(^4\)Views of three-dimensional covering problems are generated using OpenGL.
minimum aspect ratio of a covering shape. In most of the cases several million groups are considered in under one minute.

4 Dimension-Independent Measure

Without regard to the details of an implementation, suppose that a heuristic solver uses a uniform orthotope partition of \( P \) in order to accomplish the reduction to a set covering problem as described in Section 1. We discuss below practical characterizations derivable directly from a problem instance and number of parts. These lead to significant improvements in the covering heuristic.

Corner Constraints. This discussion is motivated by the theoretical orthotope covering work of Groemer [12] and Toth [20] referenced in Section 1. Translational orthotope covering problems are strongly constrained (especially in high dimensions) by the fact that a solution must cover all corners of the target orthotope. A corner-constrained instance is one in which the cover orthotopes cannot cover all of the corners. It satisfies this condition and cannot be solved: \( N < 2^d \), and \( s_i(Q_j) < s_i(P) \), \( \forall 1 \leq i \leq d \). The constraint’s second inequality guarantees that any cover orthotope can cover at most one target corner. Consequently, the first inequality guarantees that not all corners can be covered by \( Q \).

On the other hand, corner covering conditions can sometimes make an instance easier to solve. For example, if \( Q_j \) can completely cover a \( d-1 \) dimensional facet of \( P \), then, because the instance uses orthotopes, it can be reduced by placing \( Q_j \) into a position of maximum overlap with \( P \), subtracting this overlap from \( P \) and removing \( Q_j \) from \( Q \) (see [7] for details). We assume that each instance has already been reduced as much as possible.

Partition Constraints. In this region of the problem domain, the ability to solve an instance with a uniform orthotope partitioning scheme is limited by the effective quantized volume available. Let \( \rho(Q, \delta, P) = \frac{\left( \sum v'(Q_j, \delta, P) \right)}{v(P)} \) be the quantized effective volume ratio. Testing if \( \rho(Q, \delta, P) \) is at least 1 is equivalent to the feasibility test used in the pseudocode in Algorithm 1.

Characterization Experiment. Based on the above, we expect to find the hardest instances near the boundary between the potentially solvable and partition-constrained instances. Here we experimentally test the validity of this assertion. This experiment uses two-dimensional, three-dimensional, and four-dimensional randomly generated instances. The random instance generator lets the user select the following parameters: 1) dimensionality, 2) number of cover shapes, and 3) total volume of the cover shapes. Here \( P \) is a unit orthotope and each \( Q_j \) has uniformly chosen side lengths in the open interval \((0, 1)\). We desire a dimension-independent measure that predicts whether an instance with \( \varphi \) parts has a cover. With these assumptions, \( v(Q_j) = v(Q_j, P) \), so Groemer’s condition [12] applies with \( s = 1 \) (see Section 1). This yields a sufficient condition for coverage:

\[
\sqrt[4]{\sum v(Q_j, P)} \geq 2^d - 1.
\]

Instances with total effective volume close to this bound should be easy to solve, so we use instances with significantly smaller total effective volume. We offer the following definition of a per-dimension volume margin:

\[
\Psi = \sqrt[4]{\rho(Q, \delta, P)} - (1 + \frac{1}{\sqrt{\varphi}}).
\]

This is motivated by the following intuition. There appears to be a hole migration effect that occurs when there is just barely enough cover shape volume. Shifting a cover shape to cover one uncovered region of \( P \) often uncovers a different region of \( P \). However, if the effective quantized volume of \( Q \) exceeds the target volume by a single slice, then the heuristic may not have to look far to find overlapping cover volume that can be utilized to cover a hole without exposing a new hole. The formula in Eqn. 2 for our per-dimension volume margin \( \Psi \) makes the simplifying assumption that there is an equal number of slices in all dimensions.

To test the usefulness of \( \Psi \) as a predictor of coverage, for each problem instance, each pass of the repeat loop of OrthotopeCover() (doubling the number of parts used) results in a separate data point per pass. For each combination of input parameters we randomly generated approximately 10 problem instances. Data points were deliberately chosen to avoid the corner-constrained condition so that would not distort the relationship between variables. Effective volume ratios were chosen to yield similar values for \( \Psi \) and are significantly smaller than \( 2^d - 1 \). Input parameters were varied as described in Table 2. There are 1954 two-dimensional data points, 2637 three-dimensional data points, and 1664 four-dimensional data points. Thus, the choice of 10 instances for each combination of input parameters generates a statistically significant number of instances.

Results were collected on hardware profile 2. Instance data and test results are available from our web site at [6, 18]. The scatterplot in Figure 3 illustrates the effect of \( \Psi \) on coverage. Only the four-dimensional scatterplot is shown, as the scatterplots are roughly similar across dimensions two, three, and four.

Table 3 summarizes scatterplot results. The vast
Figure 3: Scatterplot of percentage of parts covered vs. $\Psi$ for four dimensions [7].

Table 2: Experiment parameters for $\Psi$ test.

<table>
<thead>
<tr>
<th>$d$</th>
<th>$N$</th>
<th>effective volume ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5.6,...,10</td>
<td>1.1,1.15,1.2,...,1.6</td>
</tr>
<tr>
<td>3</td>
<td>12,14,16,18</td>
<td>1.2,1.3,...,2.5</td>
</tr>
<tr>
<td>4</td>
<td>20,22,24,26</td>
<td>2.0,2.1,...,2.7</td>
</tr>
</tbody>
</table>

Table 3: Summary of coverage results for $\Psi$ test. Total points is denoted by $tp$. Number of points with $\Psi < 0$ is in column 3. Number of points with $\Psi < 0$ and 100% coverage is in column 4. Number of points with $\Psi \geq 0$ and 100% coverage is in column 5. Column 6 shows correlation coefficient $r$ between $\Psi$ and % coverage.

<table>
<thead>
<tr>
<th>$d$</th>
<th>$tp$</th>
<th>$# \Psi &lt; 0$</th>
<th>$# \Psi &lt; 0$</th>
<th>$# \Psi \geq 0$</th>
<th>$# \Psi \geq 0$</th>
<th>$r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1954</td>
<td>21</td>
<td>6</td>
<td>397</td>
<td>.43</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2637</td>
<td>98</td>
<td>5</td>
<td>658</td>
<td>.64</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1664</td>
<td>13</td>
<td>0</td>
<td>191</td>
<td>.77</td>
<td></td>
</tr>
</tbody>
</table>

majority of points fall on or to the right of the $\Psi = 0$ border. In two dimensions 99% of the points are in this category, with 96% in three dimensions and 99% in four dimensions. Occasionally the heuristic is lucky and finds a solution with $\Psi < 0$. Note that the number of points with positive $\Psi$ and complete coverage is far greater than the number of points with negative $\Psi$ and complete coverage. The correlation coefficients between $\Psi$ and percent coverage range from .43 in two dimensions to .77 in four dimensions. We believe that the large spread in coverage percentage for a given value of $\Psi$ can be attributed to variability in the amount of overlap between $Q_j$'s.

These results suggest that $\Psi$ is a good predictor of coverage success in the context of our OrthotopeCover( ) heuristic. Consequently, given $Q, P$ and the partitioning granularity, success can be fairly well predicted without generating groups of parts or running the Lagrangian heuristic LGC_Cover( ). This is important because there does not exist an exact algorithm to determine if a covering problem instance is feasible.

Although this does not provide a definitive test for instance hardness outside the context of a uniform partitioning heuristic, it can be used to speed up OrthotopeCover( ). The value of $\Psi$ can be calculated and $\Psi \geq 0$ can be used as a second condition in the statement that tests if $\sum v(Q_j, \delta, P) \geq v(P)$. To assess this, a follow-up experiment was performed with 17 of the randomly generated three-dimensional instances that achieved 100% coverage using 8192 parts. The number of covering shapes ranged from 12-18. On average, 2.8 calls to LGC_Cover( ) were saved per instance, with average relative percentage savings of 47.2% of calls to LGC_Cover( ) over the results reported in [7]. An additional 13 of the randomly generated three-dimensional instances that achieved 100% coverage were also tested. The number of covering shapes ranged from 12-16, and the number of parts was between 128 and 4096. On average, 2.3 calls to LGC_Cover( ) were saved per instance, with average relative percentage savings of 58.8% of calls to LGC_Cover( ) over the results in [7]. These $\Psi$ tests were run on hardware configuration 3.

## 5 Computational Bottleneck

### Execution Time. The discovery of $\Psi$ is significant because it allows the heuristic to skip over some of the time-consuming parts of OrthotopeCover( ). Due to the speed of group generation and manipulation, the running time of OrthotopeCover( ) is dominated by the running time of LGC_Cover( ). This is in contrast to the general polygonal heuristic of [5], whose running time is dominated by group maintenance. As derived in [7], the worst-case asymptotic running time of OrthotopeCover( ) is in $O(N\varphi_L^2)$, where $\varphi_L$ is the number of parts in the final iteration of the repeat loop. In our context, we have found that the running time of LGC_Cover( ) is dominated by the time required by the I-OPT improvement heuristic. As we will see below, this heuristic greatly influences the convergence of OrthotopeCover( ).

#### Monotonicity. OrthotopeCover( ) repeatedly calls LGC_Cover( ) in order to try to construct a cover if one exists. Overall success of OrthotopeCover( ) is more likely if, from one iteration of its repeat loop to the next, the objective function for the Lagrangian heuristic gets progressively closer to the total number.
of parts. This is a significant challenge for two reasons: 1) $\phi$ doubles before each call to LGC_{Cover}( ), and 2) LGC_{Cover}( ) is only a heuristic and not an exact algorithm.

From a theoretical perspective, there is no monotonicity or convergence guarantee for a feasible problem instance. Success depends directly on several obvious factors, such as $N$, $d$, and the thickness of the cover. It also depends on the richness of the group structure and the strength of LGC_{Cover}( ). We first briefly discuss the group structure and then delve into LGC_{Cover}( ).

OrthotopeCover( ) provides an effective group structure that can be efficiently created, stored, and manipulated due to the fact that $P$ is uniformly partitioned. The group construction process makes sure that if, during one iteration of the repeat loop, a part $p$ is part of a group $g$ that fits inside covering shape $Q_j$, then, at the next iteration, $p$ is also part of a group $g'$ fitting inside $Q_j$ such that $g$ fits within $g'$.

In our experiments, the percentage of parts covered by LGC_{Cover}( ) increases surprisingly well across iterations of OrthotopeCover( )'s repeat loop. To understand why in more detail, we selected a subset of 30 feasible instances from the randomly generated datasets of Section 4. This consists of 10 from each of dimensions two, three, and four. The quantized effective volume ratio $\rho$, reflecting covering thickness, varies between 1.15 and 2.7. In all cases the percentage of parts covered increases monotonically across successive calls to LGC_{Cover}( ). This is due partly to judicious choice of parameters for the search for Lagrange multipliers within the Lagrangian relaxation. However, use of the 1-OPT heuristic is highly influential.

Grinde and Daniels [11, 10] observed that, in the apparel layout context, the success of the set covering Lagrangian heuristic of Section 2 is largely due to the 1-OPT heuristic. Our experiments with hardware configuration 3 agree with this in the orthotope covering scenario. In 27 of the 30 examples, without the 1-OPT heuristic OrthotopeCover( ) is unable to find a cover using at most the number of parts required with the 1-OPT strategy. The difference is often dramatic. For example, for one two-dimensional instance with four covering shapes, with 1-OPT only 128 parts are needed to find a cover and 128 is the first part number for which LGC_{Cover}( ) is called due to the $\Psi$ test. Without 1-OPT the progression is: 124 of 128 parts covered (96.9%) → 252 of 256 (98.4%) → 496 of 512 (96.9%) → 1008 of 1024 (98.4%) → 2030 of 2048 (99.12%) → 4050 of 4096 (98.9%) → 8099 of 8192 (98.9%). At this point the heuristic fails to progress and terminates without finding a cover. Note that the coverage percentages are high but do not converge to 100%.

Our experiments exhibit monotonic improvement in coverage across successive calls to LGC_{Cover}( ) even though such behavior is not theoretically guaranteed. This observation is motivated by test runs on hardware configuration 3 for the $\Psi$ test from Section 4. The percentage of parts covered increases monotonically across calls to LGC_{Cover}( ) in all 30 of the feasible test examples examined in this section. Furthermore, the average number of calls to LGC_{Cover}( ) for finding a cover is 1.3. The maximum number of calls is from a two-dimensional example with six covering shapes and cover area ratio of 1.25, which has a tight cover. The monotonic coverage progression is: 504 of 512 parts covered (98.4%) → 1014 of 1024 (99%) → 2039 of 2048 (99.6%) → 4096 of 4096 (100%).

Alternatives to 1-OPT. In [11] it is suggested that future work on this Lagrangian heuristic consider replacing the 1-OPT heuristic with either a 2-OPT heuristic or a meta-heuristic. We experimented with both of these alternatives in the orthotope covering setting. From a computational complexity perspective a 2-OPT heuristic is too expensive, given that the number of parts ($\phi$) can sometimes be close to 10,000 and millions of part groups can be considered. The worst-case running time of 2-OPT is at least proportional to $\phi^2G^2$, where $G$ is the number of part groups per covering shape. The gains in the objective function are not sufficient to warrant this extra running time.

We tried two different ways of introducing randomization. First we substituted 1-OPT with simulated annealing. Simulated annealing’s random swaps failed to provide results as good as 1-OPT using equivalent amounts of computation time. Alternatively, we modified 1-OPT to randomly sample the list of groups for each covering shape. One round of experiments was conducted using each of the following percentages of groups during each pass of the lower bound improvement: 25%, 50%, 75%, and 100%. While the results were often comparable, especially in the 75%, and 100% cases, and the 25% case sometimes achieved superior results, the results were sometimes poor. In one two-dimensional example for 8 covering shapes, the 25% case required 199 times the execution time of the deterministic 1-OPT procedure in order to find a cover. The deterministic procedure found a cover for $\phi = 1024$ but the 25% randomized case did not find a cover until $\phi = 8192$. For this same example, the 50% case did not find a cover until $\phi = 2048$. Both the 75% and 100% cases found a cover with $\phi = 1024$, but the number of internal iterations was much larger than in the deterministic case, resulting in 4 and 11 times more computation.
time, respectively. Thus, it appears that randomization can greatly affect the convergence. Deterministic 1-OPT therefore appears to be a preferable improvement heuristic in our context.

The 1-OPT procedure was originally designed as a local improvement technique [11, 10]. We investigated the locality behavior of 1-OPT by measuring the percentage of parts that changed when comparing the starting group to the 1-OPT group chosen for each covering shape. For most of the covering shapes, 1-OPT chose groups with change percentages significantly large and often close to 100%. Thus, 1-OPT does not truly behave as a local improvement strategy. We experimented with imposing a neighborhood structure on 1-OPT via hashed group signatures. This locality restriction adversely affected convergence so the idea was discarded. To further analyze the 1-OPT behavior, we started 1-OPT from varying initial group assignments (other than those originally coming from the internals of the Lagrangian heuristic). As one might expect, it is indeed sensitive to the starting assignments. Once we concluded that 1-OPT behaves as a greedy global improvement strategy, it was natural to wonder if 1-OPT, starting with no group assignments, could completely replace the entire Lagrangian heuristic. The answer was no, but 1-OPT’s performance was strong enough to make it worthwhile as a preprocessing step before the start of the Lagrangian heuristic. In the resulting new hybrid heuristic, 1-OPT preprocessing frequently provided a better bound on the optimal solution than the one generated by 1-OPT inside the Lagrangian heuristic (whose starting assignments come from inside that heuristic). Specifically, in our set of 30 feasible covering instances on hardware configuration 3, 1-OPT preprocessing outperformed internal 1-OPT in 75% of the two-dimensional instances, 87% of the three-dimensional instances, and 64% of the four-dimensional cases. Thus, insight into the convergence behavior led to a better optimization heuristic.

6 Conclusion and Future Work

Our heuristic for translational orthotope (box) covering problems uses a uniform refinement scheme based on successive calls to a revised implementation of the Lagrangian set covering heuristic of [11, 10]. On rectangular problem instances, the orthotope solver is at least two orders of magnitude faster than the best known implementation for two-dimensional translational covering of nonconvex shapes [5]. Dimension is an input to the orthotope heuristic, which has been tested not only in two dimensions, but also in three and four dimensions. In some three and four-dimensional cases, the new heuristic is able to quickly find covers using as many as 50 covering shapes.

Because orthotope covering is NP-complete in the worst case, we investigate what makes a translational orthotope covering instance challenging in the context of a uniform orthotope refinement scheme. We provide a novel dimension-independent measure that is based on maximizing the volume of the intersection of the covering orthotopes with the target orthotope. Tests in two, three, and four dimensions using more than 6,000 randomly generated problem instances suggest that this measure is a good predictor of coverage. This measure significantly improves the efficiency of the orthotope heuristic.

The Lagrangian heuristic is fast, but it still dominates the overall running time of our method. In a representative subset of our experiments, the percentage of parts covered increased monotonically across successive calls to the Lagrangian heuristic. This is a key element of the heuristic’s success. Another important ingredient is the 1-OPT heuristic that is used inside each call to the Lagrangian heuristic. Alternatives to the deterministic 1-OPT heuristic, such as randomized 1-OPT, deterministic 2-OPT, and simulated annealing, appear less effective in the orthotope covering context. We found that the 1-OPT heuristic, which was originally intended as a local improvement strategy, does not stay within a local neighborhood of its starting point. The realization that it is actually functioning as a global, greedy optimization heuristic motivated us to add 1-OPT as a preprocessing step prior to the Lagrangian heuristic. This frequently results in an even stronger covering heuristic.

The heuristic performs quickly enough on a sufficient number of covering shapes to make it practical. Since orthotopes can be used as enclosures for nonconvex shapes, future work will attempt to employ this heuristic to speed up two-dimensional, translational covering of nonconvex shapes. Plans also include using it as part of a heuristic for three-dimensional, translational covering of nonconvex polyhedra. Allowing rotations is also a possibility. It may be worthwhile to interpret the covering problem formulation in terms of matching and covering in a tri-partite graph. Perhaps this might lead to an even better greedy preprocessing method. Ultimately, an exact algorithm is needed to test feasibility of covering instances. Software, experimental data, and results of our early experiments with our orthotope covering heuristic are available from our web site at [6, 18].

Acknowledgements: This research was supported in part by NSF/DARPA Grant 0310589. The authors are grateful for comments provided by Michelle Daniels.
References:


