

Animal Testing

Adrian Dumitrescu^{1*} and Evan Hilscher^{2**}

¹ Computer Science, University of Wisconsin–Milwaukee, USA
dumitres@uwm.edu

² Computer Science, University of Wisconsin–Milwaukee, USA
hilscher@uwm.edu

Abstract. A configuration of unit cubes in three dimensions with integer coordinates is called an *animal* if the boundary of their union is homeomorphic to a sphere. Shermer discovered several animals from which no single cube may be removed such that the resulting configurations are also animals [16]. Here we obtain a dual result: we give an example of an animal to which no cube may be added within its minimal bounding box such that the resulting configuration is also an animal. We also present two $O(n)$ -time algorithms for determining whether a given configuration of n unit cubes is an animal.

1 Introduction

An *animal* is defined as a configuration of axis-aligned unit cubes with integer coordinates in 3-space such that the boundary of their union is homeomorphic to a sphere. There are several properties that may manifest in cube configurations, but are not allowed in animals. First of all, an animal must be “connected”, that is, it must form a whole piece. Moreover, it must be *face-connected*: any two cubes are connected by a path of face-adjacent cubes. Second, an animal may not have any handles, *i. e.*, the genus of its boundary surface is zero. Third, an animal may not have any interior holes. Holes (sometimes called shells, voids, or hollow areas) are empty regions that are completely surrounded by cubes. Finally, an animal may not have any illegal adjacencies, where the boundary locally is not homeomorphic to a disk. Illegal adjacencies are defined precisely later on, but some examples are shown in Fig. 1. Three configurations that are not animals are shown in Fig. 2.

A fundamental question regarding animals is: Given two animals A_1 and A_2 , can A_1 be transformed into A_2 by a sequence of single-cube additions and removals, such that each intermediate configuration is also an animal? This problem was first communicated by Pach in 1988 at the Seventh NYU Computational Geometry Day [12, 13], and has since remained open. More recently, it has been mentioned again in [4].

A tentative algorithm for computing such a sequence is as follows. First reduce A_1 to a single cube by removals only, and then expand this single cube into

* Supported in part by NSF grant CCF-0444188 and NSF grant DMS-1001667.

** Supported by NSF CAREER grant CCF-0444188 and NSF grant DMS-1001667.

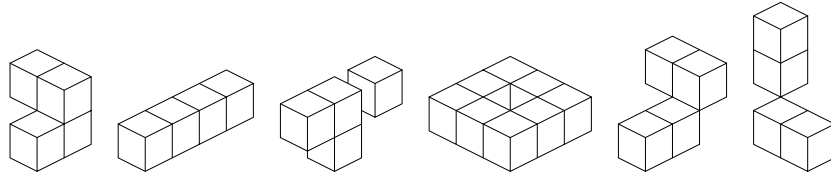


Fig. 1: Two simple animals (first two from the left), and four configurations that are not animals (3rd, 4th, 5th, 6th from the left): a disconnected configuration, a configuration with genus 1, a configuration with an illegal edge adjacency, and a configuration with an illegal vertex adjacency.

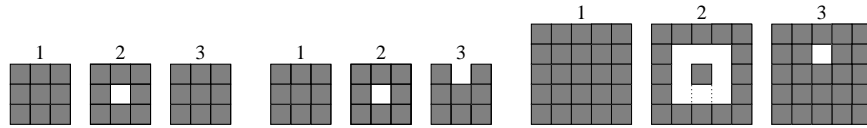


Fig. 2: Three 3-layer configurations that are not animals. Adding a cube to the configuration on the right in the cell marked with dotted lines makes it a valid animal.

A_2 by additions only. Unfortunately this algorithm was shown by Shermer [16] to be infeasible. He presented constructions of several animals such that no cube could be removed while maintaining the configuration as an animal. Such an animal is said to be *irreducible*. O'Rourke accredits the smallest irreducible animal known so far, made from 119 cubes, to Shermer [13]. We have been unable to obtain a description of this animal in the literature. The smallest irreducible animal of which we have been able to find a description consists of 128 cubes, and can fit inside of a $7 \times 7 \times 4$ box (details in Section 2).

Here we study a dual algorithm for transforming A_1 to A_2 . Consider the smallest axis-aligned box containing an animal A . Let A' be the animal such that there is a cube at every integer coordinate within the box, *i. e.*, it is a solid rectangular box containing the given animal. The algorithm is as follows:

1. Transform A_1 to A_1' by addition only.
2. Transform A_1' to A_2' .
3. Transform A_2' to A_2 by removal only.

It is easy to see that A_1' can be transformed to A_2' . We simply add or remove one layer of A_1' , one cube at a time. The only question is, can any animal A be transformed to A' by addition only? If the answer is yes, then the third step above is also feasible. As it turns out, the answer is no, thus our alternative algorithm is also infeasible.

Our results. In Section 2 we present a construction of an animal to which no cube may be added within the minimal bounding box such that the resulting collection of unit cubes is an animal. Such an animal is said to be *weakly inexpandable*. Note that any animal can be expanded by adding a cube, but not always within the

minimal bounding box. In Sections 3 and 4 we present two linear-time algorithms for deciding whether a given configuration of unit cubes is an animal, a problem which has not been studied so far. A third alternative approach is also examined in Section A. Clearly any algorithm must look at each of the n cubes to make this decision, thus our two algorithms are asymptotically optimal.

Theorem 1. *There exists an animal that is weakly inexpandable.*

Theorem 2. *Given a configuration \mathcal{C} of n unit cubes, it can be determined in $O(n)$ time whether \mathcal{C} forms an animal.*

2 Construction

In this section we prove Theorem 1. In Fig. 3, we see one of the few known irreducible animals. No cube may be removed from this animal such that the resulting configuration remains an animal. It was discovered by Shermer [16], and it was conjectured then to be the smallest possible irreducible animal.

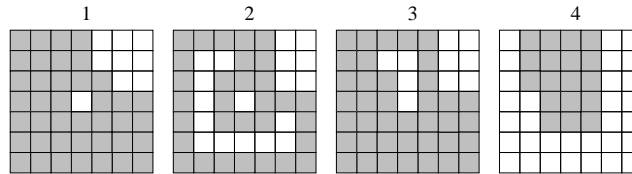


Fig. 3: The four layers of an irreducible animal due to Shermer. Shaded squares represent cells in which cubes are present, unshaded squares represent empty cells.

To prove Theorem 1, consider the cube configuration shown in Fig. 4. It can be checked that it forms an animal. Note that this animal is in some sense the dual of the animal in Fig. 3. The idea is that since no cube may be removed from an irreducible animal, then no cube may be added to the complement of that irreducible animal, with the addition of a simple bounding box construction. It can be verified that adding any single cube within the $9 \times 9 \times 6$ bounding box yields a configuration of cubes that is not an animal. An automatic verification can be also performed using an implementation of the two algorithms we present in Section 3 and Section 4.

3 Algorithm 1 Based on Fundamental Polygons and Davenport-Schinzel Sequences

In this section we present our first linear-time algorithm for determining whether a given configuration of n unit cubes is an animal and thereby prove Theorem 2.

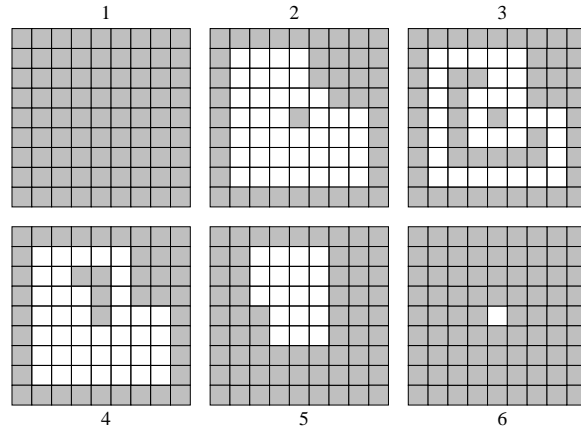


Fig. 4: The six layers of a weakly inexpandable animal. Shaded squares represent cells in which cubes are present, unshaded squares represent empty cells.

Preprocessing. The input consists of a list of n unit cubes $\mathcal{C} = \{C_1, \dots, C_n\}$ given by their (center) coordinates: (x_i, y_i, z_i) is the coordinate of C_i . First the input configuration is translated so that it lies in the first octant of the coordinate system: $x, y, z \geq 0$, and there is a cube in each of the three planes $x = 0$, $y = 0$, and $z = 0$. This step is easily accomplished in $O(n)$ time. Test now whether there exists a cube with some coordinate that is larger or equal to n : $x_i \geq n$, $y_i \geq n$ or $z_i \geq n$. Note that any animal is face-connected, *i. e.*, for any pair of cubes there is a connecting path made of cubes, such that any two consecutive cubes are face-adjacent. Since $|\mathcal{C}| = n$, the existence of a cube with a large coordinate as above implies that \mathcal{C} is not face-connected, hence not an animal.

Next we determine the set of (at most 26) other cubes that are *adjacent* to C_i , for each $i = 1, \dots, n$. From \mathcal{C} , a “neighborhood” list $\Gamma = \Gamma(\mathcal{C}) = \{N_1, \dots\}$ of at most $27n$ cells is constructed. Γ contains \mathcal{C} and all empty cells that are adjacent to some cube in \mathcal{C} . Each element of Γ is marked as a cube or as an empty cell. During construction, pointers are constructed that link each C_i to the positions in Γ of itself and of its 26 adjacent cells. Using *counting sort*, or *radix sort*, Γ is sorted six times in the following orders: (x, y, z) , (x, z, y) , (y, x, z) , (y, z, x) , (z, x, y) , (z, y, x) . This allows finding for each cell $(x, y, z) \in \Gamma$, and for each coordinate, the at most two cells of Γ next to it in that coordinate. The results (after pruning duplicate elements) are stored in six arrays A_1, \dots, A_6 , each with $k \leq 27n$ elements. During the sorting, pointers are maintained that link each C_i to the positions in the six arrays A_1, \dots, A_6 of the six occurrences of each cell adjacent to C_i . Conversely, each cell in Γ is linked to all cubes C_i to which it is adjacent to (this can be done in the process of pruning duplicate elements). The six occurrences of each cell N_j , $j \leq k$ are also linked together. Using this data structure, one can determine for each cube C_i the set of cubes adjacent to it, by following only a constant number of pointers.

Outline. The algorithm is performed in four phases:

1. Initial scan and check for illegal adjacencies.
2. Identification of twins.
3. Circular list expansion.
4. Crossing pair detection.

In phase one, we first determine the set of (at most 26) other cubes that are *adjacent* to C_i , for each $i = 1, \dots, n$. We then determine the set of boundary faces of the configuration (those faces that are adjacent to some empty cell). We then scan the input array for invalid edge-adjacencies, vertex-adjacencies, and inverse-vertex-adjacencies (described in detail below). Each cube has a constant number of neighbors, and a constant number of combinations of these neighbors must be checked for these potential problems, any of which would preclude the given configuration from being an animal. Thus this phase can be performed in $O(n)$ time.

In phases two and three, we construct a circular list of edges, called a *fundamental polygon* of the boundary of the configuration of cubes. This fundamental polygon will help us determine whether the given configuration has genus > 0 , which would indicate that the configuration is not homeomorphic to a sphere or if it has genus $= 0$, which would indicate that the configuration is an animal. We represent the fundamental polygon as a circular list L of edges. Once we have a fundamental polygon, phase four of the algorithm determines the presence (or absence) of handles by finding (or failing to find) a *crossing pair* of twins (defined below) in the overall list.

To construct this list for the entire configuration, we first construct a list of four edges for each boundary face. Each small list is oriented in a clockwise direction as viewed from the interior of the cube to which the face belongs. These lists will be then merged into a larger list. Each edge on the boundary of the cube-configuration is shared by precisely two boundary faces. As each face has its own list of four edges, there are exactly two instances of this shared edge. We say that these two instances form a *twin pair* of edges, and we point them to each other. Note that each edge has exactly one twin. The task in phase two is to determine the twin for each of the edges. Once this is done we can begin phase three, expanding some initial list of four edges, one face at a time, until all small lists of the boundary faces are merged, resulting in a fundamental polygon in the form of a circular list of edges.

Cubes, faces and edges form a hierarchy and are labeled accordingly: $E_{i,j,k}$ is the k th edge of the j th face of the i th cube, $i \in \{1, \dots, n\}$, $j \in \{1, \dots, 6\}$, $k \in \{1, \dots, 4\}$. We write $T(E_{i,j,k}) = E_{i',j',k'}$, if $E_{i',j',k'}$ is the twin of $E_{i,j,k}$. Observe that $T(T(E_{i,j,k})) = E_{i,j,k}$. A *crossing pair* of twins is a pair of twins $T(E_{i_0,j_0,k_0}) = E_{i'_0,j'_0,k'_0}$ and $T(E_{i_1,j_1,k_1}) = E_{i'_1,j'_1,k'_1}$ in list order $E_{i_0,j_0,k_0}, E_{i_1,j_1,k_1}, E_{i'_0,j'_0,k'_0}, E_{i'_1,j'_1,k'_1}$. If twin edges are labeled by the same symbol, a crossing pair of twins (for symbols a and b) is a subsequence of L of the form a, b, a, b , where $a \neq b$. It is known that a fundamental polygon represents a surface of genus > 0 (resp., $= 0$), if and only if it contains (resp., does not contain) a crossing pair of

twins. It is also true that a fundamental polygon containing some adjacent twin edges is equivalent to a fundamental polygon with those adjacent edges omitted from the list. See [1] in reference to these properties.

Using these two properties together gives us a simple method for determining the existence of a crossing pair of twins. Search for a pair of twins that are adjacent in the list (if any), remove them, and check the two edges immediately adjacent to the removed pair. If these edges are twins, we can remove them as well, and repeat until the list becomes empty. If the two edges are not twins, then continue searching the remainder of the list for another pair of adjacent twin edges. Note that removal of an adjacent pair of edges can only create a new adjacent pair of edges at that same location. Therefore we only need to continue searching the remainder of the list, rather than start the search over. If such a pair is found, then repeat the above steps. If no such pair is found, and the list is non-empty, then there must exist a crossing pair of twins in the circular list, and thus the given configuration is not an animal. If we reach a point where the list becomes empty, then there cannot have been a crossing pair of twins in the original list, and thus the given configuration is an animal. The whole procedure takes linear time in the size of the list.

Phase 1. We distinguish several types of cube *adjacencies*. We say that two cubes C_1 and C_2 are *adjacent* if and only if they share either a single vertex, a single edge, or a single face. Cubes that share a single vertex are said to be *vertex-adjacent*, cubes that share a single edge are said to be *edge-adjacent*, and cubes that share a face are said to be *face-adjacent*.

Since this adjacency information is available from the preprocessing step, we next verify whether \mathcal{C} has any illegal adjacency present, and if such is found, \mathcal{C} is ruled out as an animal. If a configuration of unit cubes contains a pair of edge-adjacent cubes C_1 and C_2 , then the configuration is an animal only if there is a face-adjacent path of 3 cubes from C_1 to C_2 ; see Fig. 5.

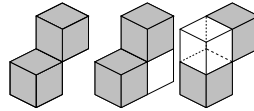


Fig. 5: An illegal edge adjacency (left), and the two minimal legal edge adjacencies (middle and right).

If a configuration of unit cubes contains a pair of vertex-adjacent cubes C_1 and C_2 , then the configuration is an animal only if there is a face-adjacent path of 4 cubes from C_1 to C_2 ; see Fig. 6.

There is one more type of adjacency, which we refer to as the *inverse-vertex-adjacency*. Two empty cells are *inverse-vertex-adjacent* if and only if they share a single vertex. There is only one situation where inverse-vertex-adjacency is

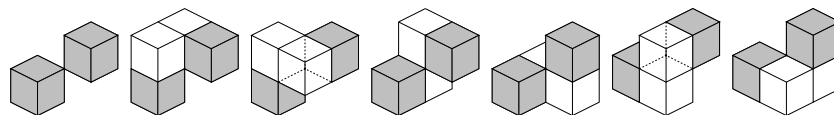


Fig. 6: An illegal vertex-adjacency (left), and the six minimal legal vertex-adjacencies.

illegal: in a $2 \times 2 \times 2$ block of cells, if only two of the cells are empty and inverse-vertex-adjacent; see Fig 7.

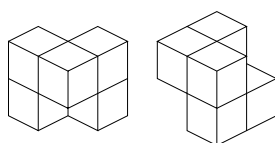


Fig. 7: Two views of the illegal inverse-vertex-adjacency configuration.

These are the only three types of illegal adjacencies that could preclude locally a cube configuration from being an animal. Given any cube along with its adjacent cubes, it is easy to determine, in constant time, if all of the above conditions are met. Since the input configuration consists of n cubes, Phase 1 takes $O(n)$ time.

Phases 2,3,4. The remainder of the algorithm makes use of the concept of *fundamental polygon* from geometric topology; see [1, pp. 60–61].

Definition 1. [1, p. 61]. A fundamental polygon is an even-sided convex polygon in the plane, whose edges are ordered clockwise. If D is a fundamental polygon, a gluing scheme S for the edges of D is a labeling of each edge of D , where each letter used in the labeling appears on precisely two edges.

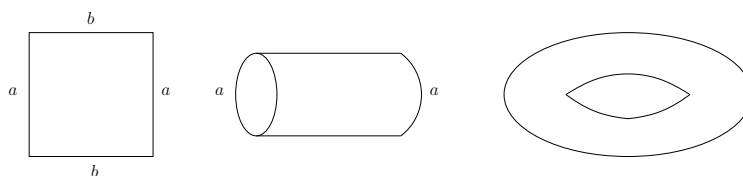


Fig. 8: A simple fundamental polygon (left), the result (a cylinder) of gluing the twin edges labeled b (middle), and the final surface (a torus) obtained by then gluing the twin edges labeled a (right); [1, p. 61].

By the following theorem, we know that we can construct a fundamental polygon for the boundary of a connected configuration of unit cubes without any illegal adjacencies.

Theorem 3. [1, p. 64]. (i) *Let D be a fundamental polygon, and let S be a gluing scheme for the edges of D . Then there is a surface $Q \subset \mathbb{R}^d$ that is obtained from D and S .*
(ii) *Let $Q \subset \mathbb{R}^d$ be a compact connected surface. Then there is a fundamental polygon D and a gluing scheme S for the edges of D such that Q is obtained from D and S .*

We now give a more detailed description of phases 2, 3 and 4. We start by examining each boundary face of the configuration. Our objective at this stage is to identify and label every pair of twin edges uniquely. This can be done in linear time, as each cube has at most six boundary faces, each boundary face has exactly four edges, and the twin of each edge instance must be in one of three possible locations; see Fig. 9.

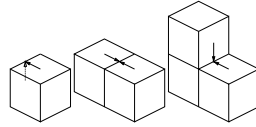


Fig. 9: The three possible locations of a twin edge.

Once the boundary edges have been labeled, we begin constructing a fundamental polygon for the configuration of unit cubes, by using breadth-first search. Start with any boundary face, say F_{i_0, j_0} , and mark this face as visited. Insert the edges of F_{i_0, j_0} into a circular list in clockwise order. Examine each of the twins of the edges of F_{i_0, j_0} . For instance, say we examine $T(E_{i_0, j_0, k_0}) = E_{i_1, j_1, k_1} \cdot E_{i_1, j_1, k_1}$ belongs to boundary face F_{i_1, j_1} . Replace E_{i_0, j_0, k_0} in our circular list with a sequence of edges: those of F_{i_1, j_1} , except for E_{i_1, j_1, k_1} , in clockwise order starting with the edge following E_{i_1, j_1, k_1} . The replacement and insertion of edges can be easily done in constant time if we maintain pointers from the edges to nodes of a circular linked list. Mark F_{i_1, j_1} as visited, and continue examining the twins of the edges of F_{i_0, j_0} . As we are doing a breadth-first search of the boundary faces, we also insert F_{i_0, j_0} into a queue so that we know to examine its edges later. It should be clear that if $T(E_{i', j', k'})$ belongs to a visited face, we return to the face containing $E_{i', j', k'}$ without doing anything, and proceed to the next edge. Let L denote the circular list (representing the fundamental polygon) obtained. Observe that L has even size, and $|L| = 2m = O(n)$. Identification of twins and circular list expansion (Phases 2 and 3) take $O(n)$ time.

Once the search is complete, we scan for unvisited faces. The existence of any such face implies that there are at least two connected components of cubes in the given configuration, and that it is therefore not an animal. Once again,

this test can easily be done in $O(n)$ time. Should it happen that all faces have been visited, we then have a fundamental polygon, in the form of a circular list with an even number of edges. Recall that a fundamental polygon represents a surface of genus > 0 (resp., $= 0$), if and only if it contains (resp., does not contain) a crossing pair of twins. Finally, with the assistance of Lemma 1 below, we can determine whether the boundary of the given configuration of unit cubes has genus 0 or genus > 0 .

A sequence $a_1 a_2 \dots$ of integers between 1 and m is called an (m, d) -Davenport-Schinzel sequence, if (i) it has no two consecutive elements which are the same, and (ii) it has no *alternating* subsequence of length $d + 2$, *i. e.*, there are no indices $1 \leq i_1 < i_2 < \dots < i_{d+2}$ such that

$$a_{i_1} = a_{i_3} = a_{i_5} = \dots = a, \quad a_{i_2} = a_{i_4} = a_{i_6} = \dots = b,$$

where $a \neq b$. Let $\lambda_d(m)$ denote the maximum length of an (m, d) -Davenport-Schinzel sequence (see [3, 14]). Obviously, we have $\lambda_1(m) = m$, and it is known that $\lambda_2(m) = 2m - 1$, for every m ($\lambda_d(m)$ is super-linear in m for $d \geq 3$). Here we are only interested in the (simple) case $d = 2$, *i. e.*, in the equality $\lambda_2(m) = 2m - 1$.

Lemma 1. *Either there exist two consecutive elements in L that are the same, or L has at least one crossing pair of twins.*

Proof. Observe that the circular list L has a crossing pair of twins if and only if any linear list derived from it (by breaking the circular list arbitrarily) has a subsequence of the form a, b, a, b , where $a \neq b$. Recall that L has length $2m$ and consists of m distinct symbols, and each symbol appears twice. Then either there exist two consecutive elements in L that are the same and we are done, or no two consecutive elements are the same. In the latter case, we either (i) can find a subsequence of the form a, b, a, b , with $(a \neq b)$, that is, a crossing as defined above, and we are done; or (ii) there is no such subsequence, hence L forms an $(m, 2)$ -Davenport-Schinzel sequence. However, since $|L| = 2m$, this would contradict the equality $\lambda_2(m) = 2m - 1$ given above. In other words, (ii) is impossible and this concludes the proof. \square

As explained in the outline of the algorithm, crossing pair detection (Phase 4) takes $O(m) = O(n)$ time, so the entire algorithm takes $O(n)$ time. Some implementation details and a small example are included in Appendix B.

4 Algorithm 2 Based on Euler-Poincaré Formula

In this section we present another linear-time algorithm and thereby obtain an alternative proof of Theorem 2. The idea to use Euler's polyhedral formula was suggested to us by Todd Phillips [11]. Originally, the algorithm consisted of one very short step: count all the edges, faces, and vertices, and then check Euler's formula [2, 10]: $V - E + F = 2$. In this formula, V , E and F stand for the number of vertices, edges and faces of a convex polytope (or of a connected planar graph).

However, there are animals for which this formula does not hold, for instance the 2-layer configuration shown in Fig. 10. We have $V = 16$, $F = 11$, $E = 24$, and $V - E + F = 16 - 24 + 11 = 3$.

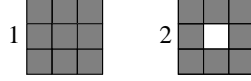


Fig. 10: A 17-cube animal for which Euler's formula does not hold.

Specifically, we need the extension provided by the Euler-Poincaré formula for manifolds [6, pp. 41–45]; see also [15]. To state this formula we need the following definitions:

- V : the number of vertices.
- E : the number of edges.
- F : the number of faces.
- G : the number of handles (holes penetrating the solid), also referred to as *genus*.
- S : the number of *shells*. A shell is an internal void of a solid or the solid itself. A shell is bounded by a 2-manifold surface, which can have its own genus value. Since the solid itself is counted as a shell, $S \geq 1$.
- L : the number of *loops*. Each face has at least one loop (a plane cycle of edges) and all outer and inner loops of faces are counted.

Then, the Euler-Poincaré formula is the following:

$$V - E + F - (L - F) - 2(S - G) = 0.$$

Observe that for $S = 1$ the formula becomes $V + 2F - E - L + 2G = 2$. Assume that the input configuration is first checked for illegal adjacencies and it passes the test. Assume also that the input configuration is face-connected, and further that we can test for $S = 1$ (*i. e.*, no interior holes). Then the cube configuration is a 3-dimensional manifold and its boundary is a single connected component. Hence the formula $V + 2F - E - L = 2$ holds if and only if $G = 0$. The resulting algorithm is as follows:

1. Check that there are no illegal adjacencies (Phase 1 in Section 3).
2. Check that the cube configuration is face-connected.
3. Construct the *unit face graph* of the solid (polyhedron) made up from the boundary faces of the unit cubes and check whether it is connected. (Two boundary unit squares are adjacent in the unit face graph if they share a boundary unit edge.)
4. Compute V , E , F and L .
5. Check the Euler-Poincaré formula $V + 2F - E - L = 2$ for the solid made up from the unit cubes.

6. If all checks pass, return that the cube configuration is an animal, otherwise return that it is not.

For instance, on the animal in Fig. 10, all checks pass and $V + 2F - E - L = 16 + 22 - 24 - 12 = 2$. Similarly, on the animal in Fig. 11(center), all checks pass and $V + 2F - E - L = 16 + 20 - 24 - 10 = 2$. On the other hand, on the 8-cube torus in Fig. 10(left), the last check fails with $V + 2F - E - L = 16 + 20 - 24 - 12 = 0$.

Algorithm analysis. Note that if there are no illegal adjacencies and the unit face graph is connected then there are no interior holes (the second assumption alone does *not* imply this conclusion; see e.g., Fig. 2, middle). So the cube configuration is a 3-dimensional manifold and its boundary forms a single connected component. Hence the formula $V + 2F - E - L = 2$ tested by the algorithm in Step 5 holds if and only if the cube configuration is an animal, as required.

The numbers V , E and F can be computed in $O(n)$ time. By determining the set of loops associated with each face, L can be also computed in $O(n)$ time. Overall, the algorithm can be implemented so that it runs in $O(n)$ time, where n is the number of cubes.

Acknowledgment. The authors are grateful to an anonymous reviewer for uncovering an error in our earlier version of Algorithm 2.

References

1. E. Bloch, *A First Course in Geometric Topology and Differential Geometry*, Birkhäuser, Boston, 1997.
2. B. Bollobás, *Modern Graph Theory*, Springer-Verlag, 1998.
3. H. Davenport and A. Schinzel, A combinatorial problem connected with differential equations, *American Journal of Mathematics*, **87** (1965), pp. 684–694.
4. A. Dumitrescu and J. Pach, Pushing squares around, *Graphs and Combinatorics*, **22(1)** (2006), pp. 37–50.
5. D. Eppstein and E. Mumford, Steinitz theorems for orthogonal polyhedra, *Proc. 26th ACM Sympos. on Computational Geometry*, ACM press (2010), pp. 429–438.
6. C. Hoffmann, *Geometric & Solid Modeling: An Introduction*, Morgan Kaufmann, 1989.
7. T. Y. Kong and A. Rosenfeld, Digital topology: introduction and survey, *Computer Vision, Graphics, and Image Processing*, **48** (1989), 357–393.
8. A. Nakamura and K. Aizawa, On the recognition of properties of three-dimensional pictures, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **7(6)** (1985), 708–713.
9. A. Nakamura, A solution to the Animal Problem, Technical Report 53, 2010, University of Auckland; www.mi.auckland.ac.nz/tech-reports/MItech-TR-53.pdf
10. J. Pach and P.K. Agarwal, *Combinatorial Geometry*, John Wiley, New York, 1995.
11. T. Phillips, personal communication, October 2010.
12. J. O’Rourke, The Computational Geometry Column #4, *ACM SIGGRAPH Computer Graphics*, **22(2)** (1988), pp. 111–112.
13. J. O’Rourke, Computational Geometry Column 6, *ACM SIGACT News*, **20(2)** (1989), pp. 10–11.

14. M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, Cambridge, 1995.
15. Ching-Kuang Shene's homepage, The Euler-Poincaré formula,
<http://www.cs.mtu.edu/shene/COURSES/cs3621/NOTES/model/euler.html>
16. T. Shermer, A smaller irreducible animal; and a very small irreducible animal,
Snapshots of Computational and Discrete Geometry (1988), pp. 139–143.
17. G. Ziegler, *Lectures on Polytopes*, Springer-Verlag, 1995.

A An Alternative Approach Based on Steinitz' Theorem

This method was suggested to us by an anonymous reviewer of an earlier version of this paper. It makes use of Steinitz' classical theorem stated below; see [17, p. 103].

Theorem 4 (Steinitz' Theorem). *G is a graph of a 3-dimensional convex polytope if and only if it is simple, planar, and 3-connected.*

Obviously animals are not necessarily convex. Given a cube configuration, the idea is to construct the edge graph for the resulting polyhedron, and then check whether the edge graph is both planar and 3-connected. If the edge graph is both planar and 3-connected, then return that the cube configuration is an animal. Otherwise, return that it is not.

There are several problems with applying Steinitz' theorem directly. A first problem is that there exist configurations which are not animals, but for which the edge graphs are both planar and 3-connected. Such a configuration is shown in Fig. 11(left). A second problem is that there exist simple orthogonal polyhedra whose edge graphs are only 2-connected. One such polyhedron, from [5], is shown in Fig. 11(center).

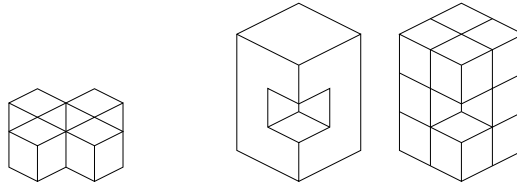


Fig. 11: Left: a 4-cube configuration that is not a valid animal, but for which the edge graph is both planar and 3-connected. Center and right: a simple orthogonal polyhedron made from unit cubes due to Eppstein and Mumford whose edge graph is only 2-connected, and the same polyhedron with the extended graph edges added.

Define the *extended graph* as the graph resulting from subdividing the faces of an orthogonal polyhedron made from unit cubes into unit squares. Note that when the extended graph edges are added to the edge graph of the polyhedron in Fig. 11(center), the graph becomes 3-connected. Since the boundary of an animal is homeomorphic to a sphere it is easy to argue that the extended graph of the polyhedron obtained from any animal is planar. We suspect that the extended graph of the polyhedron obtained from any animal is also 3-connected but we have been unable to confirm it. In addition, we would need to show that any configuration that is not an animal and has no illegal adjacencies yields an extended graph that is either not 3-connected or not planar.

Interestingly, the extended edge graph of the 4-cube configuration in Fig. 11(left) does not change if a cube is added to the central position, making it a valid animal. Due to this problem, we are forced to add an extra step that checks for

illegal adjacencies before constructing the extended edge graph of a given cube configuration. The resulting tentative algorithm is as follows:

1. Check that there are no illegal adjacencies.
2. Construct the extended edge graph of the solid.
3. Check for planarity.
4. Check for 3-connectivity.
5. If all checks pass, return that the cube configuration is an animal, otherwise return that it is not.

Whether this algorithm works as intended or needs further amendments remains an open question.

B Algorithm 1: Implementation Details and an Example

Let a cube be represented by two tuples. The first tuple (x, y, z) represents the location of C_i in three-dimensions. The second tuple consists of 26 entries

$$\delta_{\Delta x, \Delta y, \Delta z} \mid \Delta x, \Delta y, \Delta z \in \{-1, 0, 1\}, \quad (\Delta x, \Delta y, \Delta z) \neq (0, 0, 0).$$

The value of $\delta_{\Delta x, \Delta y, \Delta z}$ is an index $\in \{1, \dots, n\}$ of another cube located at $(x + \Delta x, y + \Delta y, z + \Delta z)$, into the input array, if it exists. If no cube is present at $(x + \Delta x, y + \Delta y, z + \Delta z)$, then the value of $\delta_{\Delta x, \Delta y, \Delta z}$ is set to -1 . Note that $\delta_{0,0,0}$ is not considered, as the resulting coordinates correspond to the location of C_i itself. A face $F_{i,j}$ is the j th face of the i th cube, $i \in \{1, \dots, n\}$ and $j \in \{1, \dots, 6\}$. $E_{i,j,1}, E_{i,j,2}, E_{i,j,3}$ and $E_{i,j,4}$ are the edges of $F_{i,j}$ listed in clockwise order as viewed from the interior of C_i .

As an example of the algorithm, consider the configuration of cubes shown in Fig. 12. In this configuration, the input array $A = [C_1, C_2, C_3]$, $C_1 = ((0, 1, 0), \delta_{0,-1,0} = 1, \delta_{0,-1,1} = 2)$, $C_2 = ((0, 0, 0), \delta_{0,1,0} = 0, \delta_{0,0,1} = 2)$, and $C_3 = ((0, 0, 1), \delta_{0,0,-1} = 1, \delta_{0,1,-1} = 0)$. Any omitted $\delta_{\Delta x, \Delta y, \Delta z}$ are assumed to be -1 . The convention for numbering faces is that the face in the positive z direction (from the center of C_i) is $F_{i,1}$, the face in the positive x direction is $F_{i,2}$, the face in the negative y direction is $F_{i,3}$, the face in the negative x direction is $F_{i,4}$, the face in the positive y direction is face $F_{i,5}$, and the face in the negative z direction is $F_{i,6}$. The convention for numbering the edges of vertical faces is that the edge in the positive z direction (from the center of $F_{i,j}$) is $E_{i,j,1}$. For horizontal faces, the edge in the positive x direction is $E_{i,j,1}$.

We first determine the set of boundary faces of the configuration. We then start by examining the face $F_{1,1}$, which has four edges, labeled $E_{1,1,1}$ through $E_{1,1,4}$ in a clockwise order as viewed from the interior of C_1 . Our goal is to determine the twin of each edge, $E_{1,1,1}$ through $E_{1,1,4}$. It is easy to see that $T(E_{1,1,1}) = E_{1,2,1}$, $T(E_{1,1,2}) = E_{3,5,3}$, $T(E_{1,1,3}) = E_{1,4,1}$, and $T(E_{1,1,4}) = E_{1,5,1}$. After each boundary face has been processed, every edge will have a twin. Once all twins have been identified, we begin constructing a circular list of edges representing a fundamental polygon.

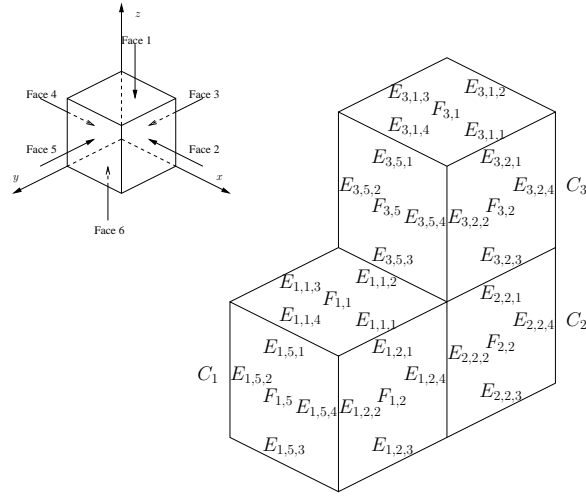


Fig. 12: Illustration of the convention for numbering faces (left), and an example configuration of unit cubes (right).

We start breadth-first search at an arbitrary face, $F_{1,1}$. To begin, we create a circular list of the edges of $F_{1,1}$ in clockwise order: $[E_{1,1,1}, E_{1,1,2}, E_{1,1,3}, E_{1,1,4}]$. We mark $F_{1,1}$ as visited, and explore the first edge, $E_{1,1,1}$. We know that $T(E_{1,1,1}) = E_{1,2,1}$. If face $F_{1,2}$ had been visited, we would return to face $F_{1,1}$ and consider $E_{1,1,2}$. However, at this point, it is clear that $F_{1,2}$ has not yet been visited, so we mark it as visited, and insert the edges of $F_{1,2}$ into our circular list of edges. As we know the clockwise ordering of the edges of $F_{1,2}$, we know the order in which they should appear in our circular list. We replace the edge $E_{1,1,1}$ with the edges of $F_{1,2}$, excluding the $E_{1,1,2}$ (the twin of $E_{1,1,1}$) and starting with the edge following $E_{1,1,2}$. Thus the circular list at this stage appears as follows: $[E_{1,2,2}, E_{1,2,3}, E_{1,2,4}, E_{1,1,2}, E_{1,1,3}, E_{1,1,4}]$; see Fig. 13, Step 2. The next edge to explore is $E_{1,1,2}$, whose twin is $E_{3,5,3}$. After inserting the edges of $F_{3,5}$ into our circular list, the list appears as follows: $[E_{1,2,2}, E_{1,2,3}, E_{1,2,4}, E_{3,5,4}, E_{3,5,1}, E_{3,5,2}, E_{1,1,3}, E_{1,1,4}]$; see Fig. 13, Step 3.

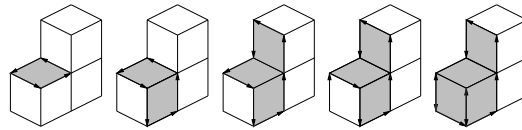


Fig. 13: The first five steps in the expansion phase of the algorithm. On the newly visited front left bottom face in the last step note two pairs of consecutive twin vertical edges (oriented both ways in the figure).

At the end of the expansion phase, we obtain the following circular list of 30 edges: $[E_{1,2,2}, E_{1,6,2}, E_{1,6,3}, E_{2,6,3}, E_{2,6,4}, E_{2,6,1}, E_{2,2,3}, E_{2,3,3}, E_{2,3,4}, E_{2,3,1}, E_{2,2,1}, E_{3,2,3}, E_{3,2,4}, E_{3,2,1}, E_{3,1,1}, E_{3,3,2}, E_{3,3,3}, E_{3,3,4}, E_{3,1,3}, E_{3,4,1}, E_{3,4,2}, E_{3,4,3}, E_{2,4,1}, E_{2,4,2}, E_{2,4,3}, E_{1,4,3}, E_{1,4,4}, E_{1,5,2}, E_{1,5,3}, E_{1,5,4}]$. We now determine if this circular list represents a fundamental polygon with genus zero. Recall that this is done by repeatedly removing adjacent twin edges from our list. We have two such pairs of adjacent twin edges in our list: $T(E_{1,2,2}) = E_{1,5,4}$ and $T(E_{3,2,1}) = E_{3,1,1}$. We remove one of these pairs and examine the pair of edges that was previously separated by this pair: $E_{3,2,4}$ and $E_{3,3,2}$. It is the case that $T(E_{3,2,4}) = E_{3,3,2}$, so we remove this pair as well. If we proceed in this manner, it will eventually be the case that the list becomes empty. We therefore conclude that the given configuration is an animal.

C Recent Update

Preliminary results of Nakamura [9] suggest that Pach's *Animal Problem* (formulated in Section 1) admits a positive solution. However, this remains to be confirmed.