Assembling and disassembling planar structures with divisible and atomic components

Yinan Zhang, Emily Whiting, Devin Balkcom

Dartmouth Computer Science

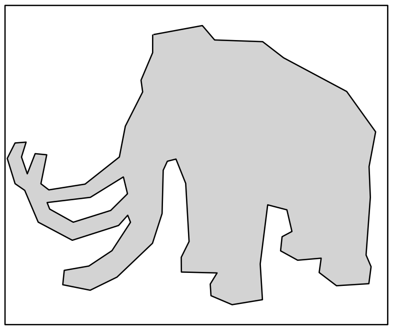
Abstract. This paper considers an assembly problem. Let there be two interlocking parts, only one of which may be cut into pieces. How many pieces should we cut the divisible part into to separate the parts using a sequence of rigid-body motions? In this initial exploration, we primarily consider 2D polygonal parts. The paper presents an algorithm that computes a lower bound on the number of pieces that the divisible part must be cut into. The paper also presents a complete algorithm that constructs a set of cuts and a motion plan for disassembly, yielding an upper bound on the required number of pieces. Applications of the future extension of this work to three dimensions may include robot self-assembly, interlocking 3D model design, search-and-rescue, packaging, and robotic surgery.

1 Introduction

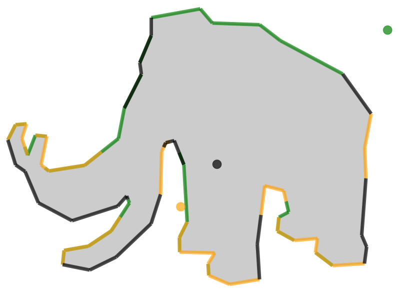
Assembly of parts is one of the oldest problems in robotics. This paper considers a variant of the assembly problem in which some parts can be cut into pieces, and others cannot be. How many pieces must an amber fossil be cut into to extract a fly? How many pieces must a model ship be broken into in order to construct a ship-in-a-bottle? How many pieces must rubble be cut into to rescue an injured person? How should styrofoam packaging be assembled to support a delicate object for transport?

As an initial exploration, we consider planar devices composed of one polygon of each of the two material types. We allow arbitrary rigid body motion of the parts after cutting; the cuts may be along arbitrary curves. Figure 1a shows an example of a mammoth in a ice cube. The gray material may not be cut, but the white material may be cut to be separated from the indivisible, or *atomic*, part.

We provide algorithms to find lower and upper bounds on the number of pieces that the device must be cut into. We also provide a prove-ably complete algorithm for design and for determining an assembly sequence. Figure 1b shows three rotation centers that could be used to locally separate ice edges of corresponding colors from the mammoth; the ice must be cut into at least three pieces. However, there is no guarantee that three is a sufficient number, because



(a) A mammoth body inside an ice cube. We want to separate the mammoth and the ice (except the hole) by breaking the ice into pieces.



(b) Edges with the same color might be locally separated from the atomic part using a single rotation about the corresponding rotation center.

Fig. 1. A mammoth in the ice and one necessary solution to remove ice without damaging the mammoth body.

global properties of the geometry also matter; we find that cutting into 61 pieces is sufficient (Figure 8c).

We believe that extension to three dimensions would have practical value for many problems in 3D printing and prototyping for robotics. Many structures cannot be printed out of a single material; robots may contain atomic components such as motors, wires, micro-controllers, and batteries that cannot be cut into pieces, supported by a rigid but divisible structure that fits around and supports the atomic components. We can also imagine applications in other areas, including search-and-rescue and robotic surgery.

However, the focus of the current paper is not on applications, but on the exploration of a fundamental robotics problem: the difficulty of assembly, measured by the number of required pieces, if some of the parts can themselves be disassembled. Such analysis of lower and upper bounds on *physical complexity* forms a useful basis for thinking about robotics problems, just as bounds on computational complexity are useful in computer science. Because the difficulty of assembly appears to depend on the shapes of the parts in non-trivial ways, we cannot simply provide an interesting bound directly; however, we can devise algorithms that compute bounds for an input shape. These bounds may give insights into the question of which shapes are hard, which are easy, and how to design appropriate shapes.

There are both local and global properties of the geometry that may cause a part to need to be cut into many pieces. Locally, two contacting rigid bodies may be interlocked, in the same way that a robot grasp may interlock with a part, effectively forming a single rigid body that must be cut to allow separation. Globally, there may be narrow openings though which only small parts can fit, the classic furniture-mover's problem.

When computing a *necessary* number of pieces, we focus on the local geometry, and relax or ignore the global geometric constraints. We consider cutting the interlocking parts into sufficiently many pieces such that there is no single immobilizing grasp, using a geometric method inspired by Reuleaux's method [20, 24] to formulate the problem as a minimum-set-cover problem. Computationally, minimum-set-cover is NP-complete, but can be approximated in polynomial-time to within a logarithmic factor. In future work, better lower bounds might be found by also considering global constraints.

We also give an algorithm to compute a *sufficient* number of pieces, by constructing cuts and an assembly order that respect global and local constraints. We prove that as long as atomic components do not contain voids, the parts can be cut into a finite set of pieces and disassembled using only translation; rotations are not required.

1.1 Related work

The current paper is closest in spirit to work on k-moldability. In the k-moldability problem, a separable k-piece mold is taken apart using a single translation per piece to expose a molded atomic part [16, 22]. Ravi and Srinivasan [23] give a list of criteria to aid the engineer in making decisions of parting surfaces. Pryadarshi and Gupta [22] used accessible directions to decompose molds into a small number of pieces. Exact-cast-mold design methods require models to be moldable or result in a large number of mold pieces. Herholz *et al.* [14] deform a model into an approximate but moldable shape, and then decompose mold pieces.

The primary contribution of the current paper is the relaxation of the requirement that the mold be separated using single translations; this allows study of the fundamental theoretical limits of dis-assembly. Most of the structures studied in this paper are not k-moldable, because there is no set of directions from which all of the divisible structure is visible; some portions of the structure are occluded. We show that in fact, any structure without inaccessible voids can be dis-assembled using only sequences of translations. The lower and upper bounds that we study are true physical bounds — they hold over *any* sequence of rigid body motions, not just single translations or rotations.

This paper is also inspired by Snoeyink's work on the number of hands required to dis-assemble a collection of rigid parts [31]. Because we allow parts to be cut, simultaneous motions are not required for dis-assembly. In the *Carpenter's Rule* problem studied by Rote, Demaine, Connelly [9], Streinu [35], and others, the rigid pieces are also all atomic, and connected by joints, typically requiring many simultaneous motions.

This paper is therefore somewhat closer in spirit to work by Wang [37] that studies the number of fingers needed to tie a knot – in that work, the string is treated as a collection of rigid bodies, but the joints may be placed arbitrarily, essentially cutting the carpenter's rule, without disconnecting the pieces. The current work is also close in spirit to work by Bell and coauthors [5] that studies the number of pieces that a mechanical knotting device must be cut into to extract the knotted string.

There has been significant work in the graphics community in computational fabrication. Song et al. [32]'s approach to fabricating large 3D objects was to break the shell of the object into pieces and assemble after fabrication. Hu et al. [15] presented a method to decompose 3D object into a set of pyramidal shapes such that no support material is needed when 3D printing the model. Fu and Song [13, 33] studied computational interlocking furniture design.

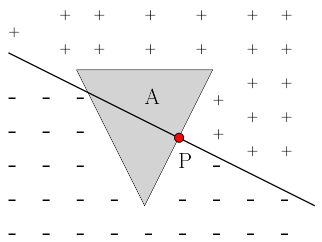
Our approach to the lower-bounds problem in particular grows out of seminal work on immobilizing rigid bodies, or *grasping*. Traditional geometric approaches to grasping attempt to prevent all possible sliding and rotational motions of a polygonal object by placing fingers around the object. Reuleaux [24] is credited with the concept of *form closure*. Mishra et al. [21] proved the sufficiency of four fingers to immobilize any polygonal object. Czyzowicz et al. [10, 11] showed that polygons without parallel edges can be immobilized using three fingers. Rimon and Burdick [26, 27] showed how two-finger grasps can be analyzed using *second-order immobility*. Cheong [7] provided an algorithm to compute all immobilizing grasps of a simple polygon. Cheong et al. [8] also showed that $n + 3$ contacts suffice to immobilize a chain of n hinged polygons.

A polygonal object can be *caged* by surrounding an object with fingers, such that the object has some freedom locally but cannot escape the cage. Some of the earliest work on caging was by Rimon and Blake [25]. Vahedi et al. and Allen et al. [36, 1] proposed algorithms to find all caging grasps of two disk fingers. Erickson et al. [12] studied the case of three-finger caging for arbitrary convex polygon. Makita et al. [19] extend the caging problem from 2D to 3D with multi-fingers. Our work, instead of caging an object, can be viewed as removing contacting pieces to uncage a polygon in the plane.

A classic problem of *self-assembly* is to move a set of small robots to specified target positions; some of the challenges with narrow corridors and coordination are similar to those faced in the current work. Kotay et al. [17], for example, designed a robotic module, groups of which aggregate into 3D structures. Rus and Vona's early work on the Crystalline robot [30] presented an algorithm to do self-reconfiguration. Recently, working on the problem of scale, Rubenstein et al. [29] provided an algorithm for moving kilo-bots one-by-one to form certain planar shapes. Arbuckle et al. [2] allowed identical memoryless agents to construct and repair arbitrary shapes in the plane. In the authors' own work [38] on assembly of interlocking structures, 9 kinds of blocks are used to build large-scale voxelized models such that all blocks are interlocked and the whole structure is rigid as a whole. Self-assembly and modular robots in the presence of obstacles has also been extensively studied. Becker et al. [4] proposed an algorithm to efficiently control a large population of robots in this scenario. Rubenstein et al. [28] used multiple robotic units to manipulate the positions of obstacles.

2 Computing a lower bound on the number of pieces

Let parts A and B be interlocked. In this section, we show how to find a lower bound on the number of pieces that part B must be cut into to separate the



(a)

Fig. 2. Contacting point P can rotate in positive direction about centers in $+$ area and in negative direction about centers in $-$ area.

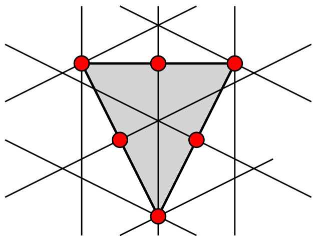

(a)

Fig. 3. The normal lines through the contact points form a set of cells, such that the number of points that may be separated from the gray part is maximized by choosing a rotation center at a cell vertex.

parts. The approach is inspired by analysis of contact modes for contacting 2D rigid bodies [3, 20], as well as by Reuleaux's graphical method for analyzing whether a collection of points fully constrains (or in our case, is constrained by) the motion of a rigid body [24].

We first replace B with a collection of points P from B along the boundary of A; whichever points we choose, *at least* these points must be separated from A using a collection of rigid-body motions. For simplicity, we place three vertices per edge: one in the center, and one at each endpoint. Choosing more points may allow a larger lower bound to be computed, at the cost of some additional computation.

Now consider any subset of these points. Can this subset be contained in a single rigid piece after cutting, in such a way that the rigid piece may be separated from A? In order to separate this piece from A, there must at least instantaneously be a single rigid body motion that at least does not cause collision for any of the points, and ideally, simultaneously separates all of the points from A. Every motion of a planar rigid body is instantaneously a rotation or a translation. Does there exist a translation direction or a rotation center that allows separation? How do we compute good (large) subsets without considering the power set over P?

Theorem 1. *If n points P of a planar rigid body B are in contact with a polygonal rigid body A, then there are at most $n(n-1)/2 + 2n$ maximal subsets of P, such that each subset may be moved together as a rigid body without colliding with A, and any other non-colliding subset is a contained within one of the maximal subsets.*

Proof. We would like to group subsets of points in P and see if they can be separated from A as a group. Let us first consider rotations. To find a compatible group, we might choose a particular rotation center and a positive or negative direction for rotation. Then find all points in P that separate from (or at least do not collide with) A under that motion; we have found a compatible subset. This is our basic approach; but how many potential rotations must we consider, and how many compatible subsets may be generated?

Consider a rotation center r somewhere in the plane, with an associated direction (either positive or negative rotation). Reuleaux's method makes use of the fact that for a particular point P_i, for most choices of rotation center, one direction of rotation (either positive or negative) is permissible, while the other causes collision of P_i with A. Along the normal to the edge of A at P_i, either negative or positive rotation is possible. Let $P_r \in P$ be the set of points compatible with rotation center and direction r. If we move r along a continuous trajectory, membership in P_r only changes as r crosses one of the normals through one of the points in P. The constraint is least restrictive along the normals themselves, so to compute the *maximal* subsets of compatible points, such that any other compatible subset is either a singleton or a subset of a computed subset, we need only consider rotation centers at the intersections of the normals, as shown in Figure 3. The normals form an *arrangement* [34], and there are $n(n-1)/2$ possible intersections, each with at most one corresponding maximal subset(Figure 3). Once candidate maximal subsets have been generated, discard any that are contained within other computed subsets.

Translation directions may be analyzed similarly. Each point P_i, if included in a subset, forbids an open half-plane of translation directions. So for each point, it is sufficient to test two translations directions, each corresponding to sliding in one direction or another along the point. Collect all $2n$ directions, and for each direction, test the remaining points against that direction to generate candidate maximal subsets.

The proof of Theorem 1 implies an algorithm for computing a lower bound on the number of pieces that the divisible part must be cut into to allow assembly or disassembly. Compute the maximal subsets as suggested; then solve the minimum-set-cover problem to find the minimum number of such sets needed to separate all points in P from A. If the number of maximal subsets is small, minimum-set-cover may be solved exactly. The simple examples presented in this paper were solved exactly using integer-linear programming. If there is a large number of subsets, then a greedy approach yields a solution in polynomial time, with guaranteed logarithmic approximation quality.

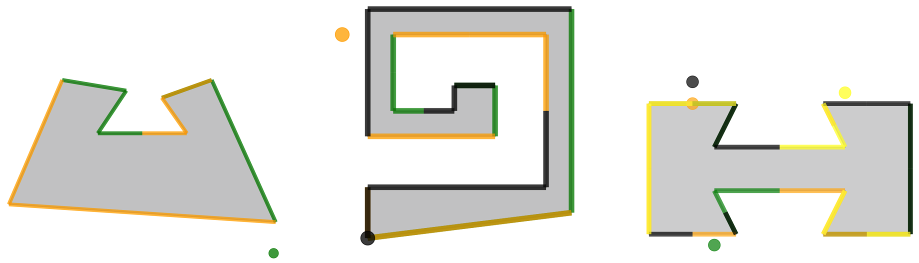
Figure 1b shows a solution of the necessary number of pieces needed to extract the planar mammoth from the ice. If two points on the same edge are in the same set, the segments between the points are considered able to rotate about centers in the same region as the two points. In this case, three sets cover all edges.

Edges containing points in the same set are not necessarily connected. Whether there exist cuts to separate edges exactly into the derived sets as connected rigid bodies is an open question, as is whether those bodies can be extracted after initial separation. This technique thus yields only a lower bound, and we expect that the lower bound might be significantly improved in future work.

Results for other shapes can be found in Figure 4. A few statistics are shown in Table 2. Time costs were measured on a 2016-model MacBook Pro with a 2.6 GHz Intel processor and 8 GB 1600 MHz DDR3 memory, and are intended only to give a sense of the practicality of analysis of analyzing structures with varying numbers of edges. From the table, we can see that with the increase of maximum number of rotation sets, the time cost increases dramatically, as we would expect for a $O(n^3)$ checking of rotations centers and a linear-integer program optimal solution to minimum-set-cover; we expect that much larger problems could be solved with good approximation by using greedy minimum-set-cover techniques.

shape	# edges	largest set size	set cover size	time cost
cavity	8	144	2	0.2018 s
spiral	14	612	3	1.8764 s
dumbbell	12	1104	4	7.2543 s
mammoth	64	12012	3	540.327 s

Table 1. Lower-bound analysis examples.



(a) Cavity. Orange edges move to the left, and green edges rotate about the green point.

(b) Spiral. Black and orange edges rotate about black and orange points respectively. Green edges move to the right.

(c) Dumbbell. 4 sets of edges in 4 colors. Each set of edges rotates about centers with the same color.

Fig. 4. Three examples of analyzing necessary number of pieces the divisible part should be cut into. In each example, edges with the same color are in the same set.

3 Computing a sufficient number of pieces

In this section, we present a complete algorithm that chooses where to cut the divisible part B, and finds a motion plan to achieve separation from the indivisible part A. We want to find an upper bound on the minimum number of pieces B can be cut into to move each piece out of a planar box that contains both A and B.

The algorithm first decomposes the part into a small number of convex polygons, and moves pieces within these convex shapes; the number of pieces depends on the number and size of polygons. In our implementation, we used a Delaunay triangulation, but better bounds could be achieved by finding larger convex components.

After decomposition of B into convex polygons, consider two adjacent convex polygons that share an edge. Flipping one polygon about the shared edge and intersecting with the other polygon gives a new convex polygon. Inside the new polygon we compute a largest inner square with one edge on the shared edge. This square, called the *transit square*, can move freely between the two convex polygons without leaving the interior.

We would like to find an axis-aligned grid such that at least one complete grid cell fits entirely within the transit squares for each pair of adjacent convex polygons. For each transit square, we find the largest axis-aligned inscribed square, divide the width by two (shown in Figure 5c), and take the minimum over all such values as the width of cells in the grid.

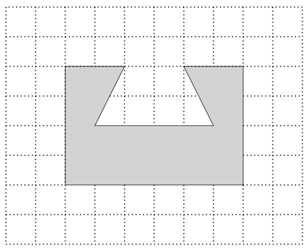
To find the size of the largest axis-aligned square, assume the width of the outer square is L and the small angle between x-axis and edges of the square is α. There exists another square of edge length $l = L\sqrt{1 - 2 \cdot \tan\alpha/(1+\tan\alpha)^2}$ inside the outer square.

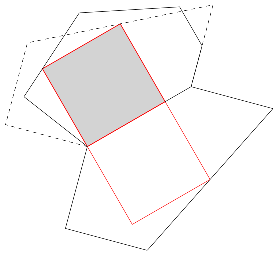
We intersect the grid cells with the convex polygons to create small pieces that we will call *components*. We will extract the material from each convex polygon; we will say that a polygon that has already had its material extracted is empty, and one that has not is full. We will prove that components can move from one full convex polygon to an adjacent empty polygon without leaving either convex polygon (and thus without collision with atomic parts or uncleared divisible parts), assuming each component disappears once it has completely entered the empty polygon. This is sufficient to prove inductively that the entire structure can be disassembled.

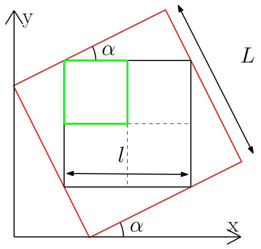
Lemma 1. *In a planar grid of square cells with a designated target square, continuous translation of each grid cell to the target will not cause collision, if the cells are translated in order of L_2 distance from the target.*

Proof. Let A be a square whose bottom-left point is at position (x_a, y'_a), moving in one direction towards the target square O, with bottom-left point at (x_o, y_o). Without loss of generality, assume $x_a > 0, y_a > 0$ and $x_o = 0, y_o = 0$, and that the width of each cell is 1. We know that, during the motion, every point of square A is bounded by the rectangle R defined by its bottom-left point $(0, 0)$ and top-right point $(x_a + 1, y_a + 1)$.

Assume A collides with a square B whose bottom-left point is at (x_b, y_b), $x_b, y_b \in \mathbb{Z}^+$. Then $0 \le x_b \le x_a$ and $0 \le y_b \le y_a$; otherwise, no point in the square is in R. Because OA is along the diagonal of R, OA is the longest edge in


(a) Two interlocked parts where the gray part is atomic and the rest is divisible. Gridding the divisible part into small enough cells and removing them will separate both parts.


(b) Between two adjacent convex shapes, there exists a square that can move freely from one to another without leaving either shape.



(c) Partitioning the space using a (green) square of width $l/2$ guarantees there exists at least one square in the transition square.

Fig. 5. Gridding the divisible part to find sufficient pieces to separate parts.

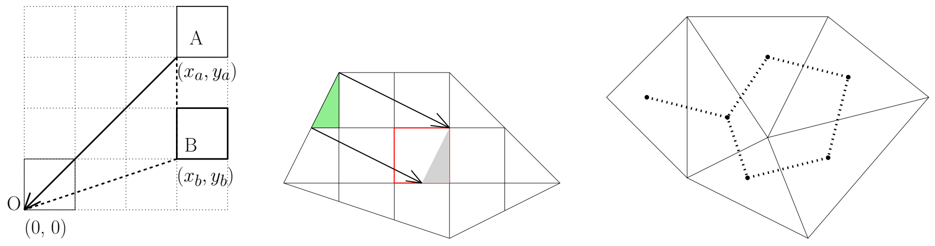
triangle $\triangle OAB$. So $|OB| < |OA|$, and square B would have been moved before A using the proposed order.

Although we claim and prove Lemma 1 in the plane, it extends easily to similar results in arbitrary dimensions. We now come to the main result of this section:

Theorem 2. *Given an intersection of a grid with a pair of adjacent convex polygons, such that at least one complete grid cell is completely contained within each of the transit squares of the polygons, one polygon can be emptied into the other without collisions, using the intersections of the cells with the polygons as components, assuming each component disappears once it has completely entered the empty polygon.*

Proof. Lemma 1 indicates that cells in a grid can be translated to a target in order of distance without collision. Since all motions of all points in cells are along straight lines during translation, the result extends trivially to a case where the space is constrained to a convex polygon, and the cells are clipped by the convex polygon, as are the previously-defined *components*.

We therefore have the following approach. First, empty the transit square in the full polygon into the transit square in the empty polygon, by sorting the grid cells in order of distance from an arbitrarily chosen complete grid cell in the empty polygon, and translating those cells to the target in that order; since the pair of transit squares is together a convex polygon, there are no collisions. Then choose a target cell in the now-empty transit square in the first polygon.



(a) Squares in a grid space can move to any target position with no collision following the order of their distances to the target.

(b) Cells can also move to any target square with no collision in the grid space bounded by a convex polygon using the same method.

(c) The convex polygons forms a graph that may be weighted by the size of the transit squares connecting each pair of polygons. Removing the smallest edges but keeping the connection between vertices improves the grid square size.

Fig. 6. Moving cells inside a convex shape, and between multiple convex shapes.

Sort the components in the polygon based on their distance from the target cell. In this order, first translate each component first into the transit square, and then into the adjacent empty polygon.

3.1 Algorithm 1: simple separation

The previous theorem suggests a simple, but complete, algorithm for cutting and separating the interlocked parts. First, using triangulation or some other means, decompose B and its containing square (from which we would like to remove B) into convex polygons. Polygons within B will be full, and polygons outside of B will be empty. Define a *boundary* polygon as a polygon that is connected by a sequence of adjacent empty polygons (the *exit sequence*) to the outside of the containing square, the *exit*.

Figure 7 shows the approach to extraction. Choose a boundary polygon, and an exit sequence. Extract components from that polygon one at a time in the order suggested in the proof of the theorem. As each component enters the exit sequence, move it through the sequence of empty polygons to the exit, using translation first to and then through each pair of transit squares along the exit sequence.

3.2 Algorithm 2: greedy separation with grouped components

There are many possible ways to improve Algorithm 1. For example, if there is a tiny shape in the convex decomposition of the divisible part, the cell size will be very small. In this section, we propose an algorithm to plan a path for every cell generated by the decomposition algorithm. Based on the paths, we

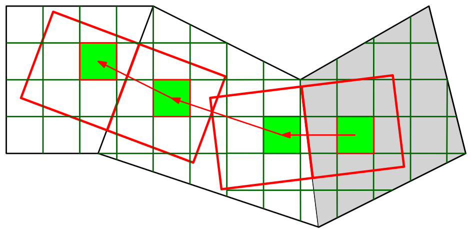


Fig. 7. Moving components through a chain of convex polygons. White polygons are empty and the gray one is filled. Using the pairs of transit squares, cells can move between any two adjacent polygons, or through a chain of connected polygons.

aggregate components into *pieces* and cut only along piece divisions; this will greatly reduce the number of pieces needed.

Let the *path* of a component be $P = [p_0, p_1, p_2, \ldots, p_n]$ where $p_i \in P$ is the intermediate position after the i-th control, and p_0 is the initial position of the component, where a *control* is a single translation. Define the *control sequence* for the component as $C = [c_1, c_2, \ldots, c_n]$ where $c_i = p_i - p_{i-1}$ is the i-th control.

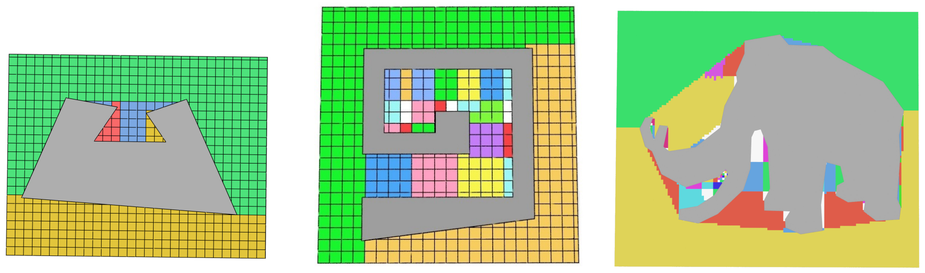
A simple greedy approach to grouping components is as follows. Components may be *whole* (entire grid cells) or *partial.* We first deal only with whole components. Simulation motion of all of the whole components in each of the four cardinal directions. For each direction, count the number of components that reach the exit area, and may be grouped together based on 4-connectivity; greedily choose the direction that gives the fewest such grouped pieces.

Add the resulting cleared squares as a target area, and attempt motion in each of the four cardinal translation directions, potentially creating new piece groups; attempt to then move these new groups to the exit.

Table 2 shows some statistics for the sufficient decomposition for some example shapes, and a few illustrative run times. The main time cost is spent on testing collisions. Decomposition solutions are shown in Figure 8.

shape	# components	# pieces	decomposition time	planning time
cavity	144	6	0.862 s	0.822 s
spiral	462	26	0.786 s	1.500 s
dumbbell	269	8	0.529 s	0.496 s
mammoth	61	9954	16.816 s	149.192 s

Table 2. Analysis of sufficient number of pieces for a few example shapes.



(a) Cavity. The divisible part is divided into 6 pieces.

(b) Spiral. The divisible part is divided into 26 pieces.

(c) Mammoth. The divisible part is divided into 61 pieces.

Fig. 8. Decompositions of separable parts into pieces. Colors are reused as needed; each set of same-color components is a single piece.

4 Conclusions, limitations, and future work

We proposed the problem of separating a divisible polygon from an interlocked atomic polygon by cutting. We explored lower and upper bounds on the number of pieces the divisible part must be cut into, and presented algorithms to make the cuts and achieve the separation using a sequence of translations. Both bounds are very conservative; the main contribution of this work is the proposal of the problem and an initial exploration of solutions.

Improving the quality of bounds is of great interest for future work. In the section of computing the *necessary* number of pieces the divisible part must be cut into, we only considered some sets of points immediately adjacent to the edges of the atomic part, ignoring global properties. For example, extremely small exit corridors through the atomic part should increase the lower bound. Such properties might be considered by how large a component might be before it 'plugs' a hole in configuration space.

While computing a *sufficient* number of pieces, the algorithm firstly decomposes the divisible part into convex polygons, then computes a grid resolution using all pairs of adjacent polygons. The algorithm can generate a large number of pieces if there exists a single narrow corridor anywhere in the divisible part. Different convex decomposition methods might also yield better decompositions, as might the selection of good disassembly sequences [18, 6].

In future work, we would also like to expand the problem scope. We expect that most practical problems in this area are 3D, rather than planar. Also, we can imagine situations where there are several atomic components, and several divisible components. How should such multi-component 3D puzzles be designed, assembled, or disassembled?

References

1. Thomas F Allen, Joel W Burdick, and Elon Rimon. Two-finger caging of polygonal objects using contact space search. *IEEE Transactions on Robotics*, 31(5):1164–1179, 2015.
2. DJ Arbuckle and Aristides AG Requicha. Self-assembly and self-repair of arbitrary shapes by a swarm of reactive robots: algorithms and simulations. *Autonomous Robots*, 28(2):197–211, 2010.
3. Devin J Balkcom and Jeffrey C Trinkle. Computing wrench cones for planar rigid body contact tasks. *The International Journal of Robotics Research*, 21(12):1053–1066, 2002.
4. Aaron Becker, Golnaz Habibi, Justin Werfel, Michael Rubenstein, and James McLurkin. Massive uniform manipulation: Controlling large populations of simple robots with a common input signal. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 520–527. IEEE, 2013.
5. Matthew P Bell, Weifu Wang, Jordan Kunzika, and Devin Balkcom. Knot-tying with four-piece fixtures. *The International Journal of Robotics Research*, 33(11):1481–1489, 2014.
6. Lukas Beyeler, Jean-Charles Bazin, and Emily Whiting. A graph-based approach for discovery of stable deconstruction sequences. In *Advances in Architectural Geometry 2014*, pages 145–157. Springer, 2015.
7. Jae-Sook Cheong, Herman J Haverkort, and A Frank van der Stappen. Computing all immobilizing grasps of a simple polygon with few contacts. *Algorithmica*, 44(2):117–136, 2006.
8. Jae-Sook Cheong, A Frank Van Der Stappen, Ken Goldberg, Mark H Overmars, and Elon Rimon. Immobilizing hinged polygons. *International Journal of Computational Geometry & Applications*, 17(01):45–69, 2007.
9. Robert Connelly, Erik D Demaine, and Günter Rote. Straightening polygonal arcs and convexifying polygonal cycles. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 432–442. IEEE, 2000.
10. Jurek Czyzowicz, Ivan Stojmenovic, and Jorge Urrutia. Immobilizing a polytope. In *Workshop on Algorithms and Data Structures*, pages 214–227. Springer, 1991.
11. Jurek Czyzowicz, Ivan Stojmenovic, and Jorge Urrutia. Immobilizing a shape. *International Journal of Computational Geometry & Applications*, 9(02):181–206, 1999.
12. Jeff Erickson, Shripad Thite, Fred Rothganger, and Jean Ponce. Capturing a convex object with three discs. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2242–2247. IEEE, 2003.
13. Chi-Wing Fu, Peng Song, Xiaoqi Yan, Lee Wei Yang, Pradeep Kumar Jayaraman, and Daniel Cohen-Or. Computational interlocking furniture assembly. *ACM Transactions on Graphics (TOG)*, 34(4):91, 2015.
14. Philipp Herholz, Wojciech Matusik, and Marc Alexa. Approximating Free-form Geometry with Height Fields for Manufacturing. *Computer Graphics Forum (Proc. of Eurographics)*, 34(2):239–251, 2015.
15. Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. Approximate pyramidal shape decomposition. *ACM Trans. Graph.*, 33(6):213–1, 2014.
16. Jun Huang, Satyandra K Gupta, and Klaus Stoppel. Generating sacrificial multi-piece molds using accessibility driven spatial partitioning. *Computer-Aided Design*, 35(13):1147–1160, 2003.

17. Keith Kotay, Daniela Rus, Marsette Vona, and Craig McGray. The self-reconfiguring robotic molecule: Design and control algorithms. In *Workshop on Algorithmic Foundations of Robotics*, pages 376–386. Citeseer, 1998.
18. Anne Loomis. {Computation reuse in stacking and unstacking}. 2005.
19. Satoshi Makita and Yusuke Maeda. 3d multifingered caging: Basic formulation and planning. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2697–2702. IEEE, 2008.
20. Matthew T Mason. *Mechanics of robotic manipulation.* MIT press, 2001.
21. Bhubaneswar Mishra, Jacob T Schwartz, and Micha Sharir. On the existence and synthesis of multifinger positive grips. *Algorithmica*, 2(1-4):541–558, 1987.
22. Alok K Priyadarshi and Satyandra K Gupta. Geometric algorithms for automated design of multi-piece permanent molds. *Computer-Aided Design*, 36(3):241–260, 2004.
23. R. Ravi and M. N. Srinivasan. Decision criteria for computer-aided parting surface design. *Comput. Aided Des.*, 22(1):11–17, January 1990.
24. Franz Reuleaux. *Theoretische Kinematik: Grundzüge einer Theorie des Maschinenwesens*, volume 1. F. Vieweg und Sohn, 1875.
25. Elon Rimon and Andrew Blake. Caging 2d bodies by 1-parameter two-fingered gripping systems. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 2, pages 1458–1464. IEEE, 1996.
26. Elon Rimon and Joel W Burdick. Mobility of bodies in contact. i. a 2nd-order mobility index for multiple-finger grasps. *IEEE transactions on Robotics and Automation*, 14(5):696–708, 1998.
27. Elon Rimon and Joel W Burdick. Mobility of bodies in contact. ii. how forces are generated by curvature effects. *IEEE Transactions on Robotics and Automation*, 14(5):709–717, 1998.
28. Michael Rubenstein, Adrian Cabrera, Justin Werfel, Golnaz Habibi, James McLurkin, and Radhika Nagpal. Collective transport of complex objects by simple robots: theory and experiments. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 47–54. International Foundation for Autonomous Agents and Multiagent Systems, 2013.
29. Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799, 2014.
30. Daniela Rus and Marsette Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.
31. Jack Snoeyink and Jorge Stolfi. Objects that cannot be taken apart with two hands. In *Proceedings of the ninth annual symposium on Computational geometry*, pages 247–256. ACM, 1993.
32. Peng Song, Bailin Deng, Ziqi Wang, Zhichao Dong, Wei Li, Chi-Wing Fu, and Ligang Liu. Cofifab: Coarse-to-fine fabrication of large 3d objects. *ACM Transactions on Graphics.*
33. Peng Song, Zhongqi Fu, Ligang Liu, and Chi-Wing Fu. Printing 3d objects with interlocking parts. *Computer Aided Geometric design (Proc. of GMP 2015)*, 35-36:137–148, 2015.
34. Jacob Steiner. Einige gesetze über die theilung der ebene und des raumes. *Journal für die reine und angewandte Mathematik*, 1:349–364, 1826.
35. Ileana Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 443–453. IEEE, 2000.

36. Mostafa Vahedi and A Frank van der Stappen. Caging polygons with two and three fingers. *The International Journal of Robotics Research*, 27(11-12):1308–1324, 2008.
37. Weifu Wang and Devin Balkcom. Grasping and folding knots. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3647–3654. IEEE, 2016.
38. Yinan Zhang and Devin Balkcom. Interlocking structure assembly with voxels. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016.