

A Path Planning Algorithm for Single-Ended Continuous Planar Robotic Ribbon Folding

Anusha Nagabandi, Liyu Wang, and Ronald S. Fearing

Abstract—Ribbon folding is a new approach to structure formation that forms higher dimensional structures using a lower dimensional primitive, namely a ribbon. In this paper, we present a novel algorithm to address path planning for ribbon folding of multi-link planar structures. We first represent the desired structure with a graph-based representation of edges and nodes. We then use graph theory to claim that for any object which is represented by a connected graph, there exists a continuous path which visits all of its edges. Finally, we develop a path planning algorithm that takes into account the physical constraints of the folding machine. The input is the desired planar structure, and the output is the optimal sequence of ribbon folds for creating that structure using the minimum number of folds. The results of this algorithm are successfully used to fold various planar structures.

I. INTRODUCTION

Significant progress has been made in the area of additive structure formation. For instance, 3D printers have made a significant impact in rapid prototyping and manufacturing by laying down thin layers of material in succession to construct solid objects. Such 1-dimensional approaches to structure formation use string or fiber-like material as the building primitive. Having such a simple building primitive does not allow for more complex features, such as joints and moving parts for crawling robots or other structures. Furthermore, such methods require a lot of material and time, as well as a large and complex machine which is not useful for portable applications such as in-field repair. The 2-dimensional approach to structure formation looks at areas of sheet folding, where the building material can be of any specified length and width. Such folded structures (Figure 1) have the advantage of introducing less friction to systems, being lightweight, decreasing build time, and using less material than conventional additive techniques. However, sheet folding needs a larger area for manufacturing, entails dexterous manipulation, and requires either human assembly of separately constructed segments or extensive engineering to create the design for a single fully-connected object. Thus, it

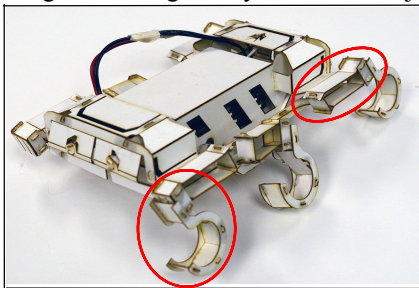


Figure 1. Folded robots like this OpenRoACH [5] contain planar components, such as 4-bar linkages and claws, that can readily be ribbon folded

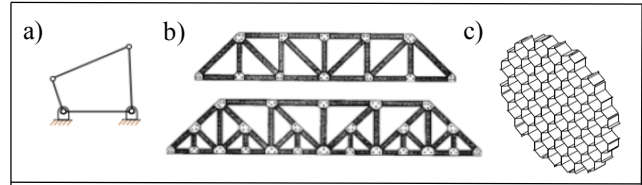


Figure 2. Practical applications of ribbon folded robotic kinematic structures such as (a) a four-bar linkage [2], and structural designs such as (b) a truss [3] and (c) a honeycomb lattice [4]

remains an open problem to develop a method of structure formation which allows for more complex structural features, while addressing concerns such as speed, choice of materials, generalizability, complexity, size, and portability of machine.

Inspired by ribosomal assembly that occurs in biological systems, ribbon folding is a new method of manufacturing that forms higher dimensional structures using a lower dimensional primitive, namely a ribbon. Using ribbons, one could create shells, fill volumes, cover surfaces, and produce other outputs desired from a fabrication method. Additionally, the ribbon material itself is a more complex primitive than just a string or fiber, so it can be altered and chosen to allow flexibility, mobility at joints, or curvature. It further paves the path for more complex construction such as knots, holes, and empty spaces. Ribbon folding is considered to be a 1.5-dimensional approach because the building material is of finite height and infinite length. In this paper, we focus on the planar case, which includes interesting structures (Figure 2). Planar kinematic structures, in particular, lend themselves well to ribbon folding. Of specific interest is the application of ribbon folding to areas of robotics, from rapid prototyping to robotic self-assembly or self-repair.

Ribbon folding takes a passive approach: actuators are only used once to assemble the structure, instead of being in-built at each joint to fold it all at once. While this serial building approach may take more time than a parallel approach, it allows a simpler and less expensive method, where the number of actuators are independent of the number of joints in the object. RATCHET [1] follows a similar philosophy of using an external manipulator to fold a chain under the force of gravity, simplifying its design and eliminating the need for a motor at each joint. This serial approach also has a larger workspace than parallel self-folding approaches; larger structures can be created while at the same time using less building material.

To perform the ribbon folding, we aim to use the simplest possible machine that allows construction of the largest range of structures. We designed a machine that folds the desired shape from one piece of ribbon (continuous), one fold at a time

(serial), from one end to the other (sequential), and from only 1 end (single-ended). Through the serial, sequential, and single-ended requirements, we eliminate the need for a dexterous manipulator with high degrees of freedom. The continuity requirement precludes the need for an additional mechanism to later reassemble the separate pieces that are created, or the need for a human assembly step. An additional benefit of continuity in the structure is that it forms a backbone for future additions, such as wiring and electrical connections throughout the structure.

As explained above, our problem statement is to address sequential single-ended folding of a continuous ribbon into a multilink planar structure. In this paper, we begin by discussing related work in structure formation. We use graph theory to validate our claim that for any object which is represented by a connected graph, there exists a continuous path which visits all of its edges. Then, we show our proof-of-concept ribbon folding machine and discuss the constraints that minimalistic machines impose on the algorithm. Next, we present a novel algorithm, which takes physical constraints into account, to generate the optimal folding sequence for creating planar structures. Finally, we employ this algorithm and discuss results, extensions, and future plans.

II. PRIOR WORK

Previous research efforts have explored approaches such as programmable matter [6], tetrahedron shaped modules folded into arbitrary 3D shapes [1], and chain folding for filling in an object's volume. For creating 3D objects, Cheung et al. [7] discretizes an object's volume into voxels and then uses a universally foldable string to follow a Hamiltonian path through these voxels. A Hamiltonian path is one which visits each node of a graph, and this translates to physically filling in the object's volume. Relatedly, ribbon folding creates objects by covering each edge with ribbon, and recent work [8] proposes a general workflow for robotic ribbon folding.

Folded structures have been well studied through applications such as joints for insect-like robots [9], wings for flying robots [10], pop-up book MEMS with features on the micron scale [11], and shape-memory composites that fold themselves along embedded hinges to create self-folding machines [12]. However, automated folding for structure formation has not yet been thoroughly studied. Lu and Akella [13] apply robotic sheet folding to the automation of the packaging process. By representing the carton as a manipulator with revolute joints and links, they demonstrate a motion planning algorithm which generates a folding sequence for an industrial robot to fold a carton from a sheet.

The area of robotic origami folding also involves the fabrication of 3D structures from planar material. It lends itself to automated assembly, self-assembly, and printable robotics [14]. Origami offers many challenges, such as modeling structures, planning folding sequences to achieve that model, and manipulating the actuators to create the object. Greenberg et al. [15] model kinetic origami behavior using pseudo-rigid-body models because the origami is compliant. Greenberg [16] uses origami to better understand flat-folding mechanisms and lamina emergent mechanisms (LEMs) in general. Balkcom and Mason [17] create kinematic models by representing creases as revolute joints and uncreased paper as rigid.

In the area of folded structures in robotics, Hoover and Fearing [18], Onal et al. [14], and Haldane et al. [5] make use of inexpensive materials to rapidly create fully functional prototypes of folded millirobots. The time saved from such folding processes allows for multiple improvement cycles and immediate discovery of design flaws. Carbon fiber components for milli-robot prototyping [19] are used to create links and flexure elements, since many basic robotic structures can be fabricated from simply these two elements. Micro assembly for the rapid prototyping of structures [20] uses a milli-robot to manipulate beams and attach them to polyester flexures, which serve to replace revolute joints. Similar to our algorithm's search for folding angles, automation of this micro assembly process involves determining folding angles for bending stainless steel sheets into triangular configurations.

Algorithmically, the problem of hyper-redundant serial chain manipulators [21], [22] is similar to ours. They have numerous actuators to allow many degrees of freedom, can avoid obstacles, and aim to achieve some goal configuration. Instead of manipulating a serial chain into a desired configuration, our problem has the additional variable of link lengths. We must create a serial chain out of the desired structure, so the chain itself is not known ahead of time.

From Demaine's work [23], [24], which includes paper folding and unfolding polyhedra, of particular interest is his work in the folding of planar linkages. Demaine states that in the context of not allowing linkages to cross (self-intersect), three general types of linkages are commonly studied: polygonal chains with a single path solution, closed polygonal chains with a single cycle solution, and polygonal trees with a single tree solution. He also states that other more general graphs, such as the ones discussed in this paper, have only been studied in the context of allowing the linkages to cross. In related work, Arkin et al. [25] study bending wires and sheet metal into a specified structure. They study a variation of the carpenter's ruler problem where they must decide whether a

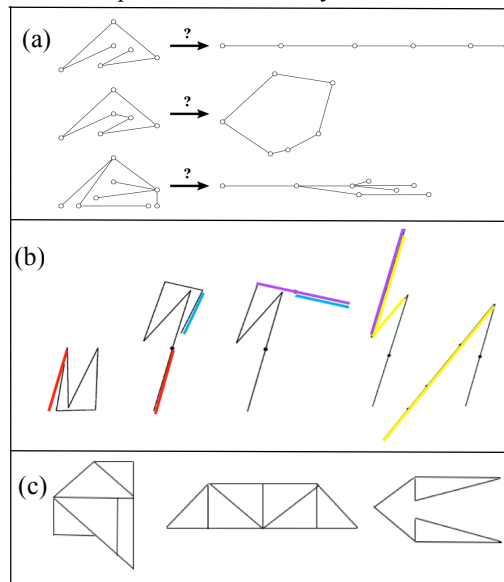


Figure 3. (a) The three linkages commonly studied in the context of non-crossing linkages [23], (b) Unfolding paperclip-like structures by unfolding one vertex at a time, but in any order [25], (c) Examples of the much larger subset of structures discussed in this paper, folded in a continuous, sequential, and single-ended manner

non-self-intersecting wire structure can be straightened, but by modifying only one joint at a time. For a non-self-intersecting linkage, Connelly et al. [26] and Streinu [27] show that it can always be straightened by modifying one joint at a time. With even a single vertex degeneracy, Arkin et al. [25] show that it becomes NP hard to decide if the shape can be straightened one joint at a time. Folding a linkage is simply the reverse of unfolding it, but our problem setup is inherently different. We allow repeated edges, which differentiates it from simply reversing an unfolding. We also allow only sequential and single-ended folding. Furthermore, we demonstrate our algorithm on graphs with numerous vertex degeneracies.

III. UNDERLYING GRAPH THEORY SHOWS GENERALITY

The goal of our algorithm is to take a structure and automatically generate an ordered list of link lengths (where to fold) and a sequence of folding angles (how to fold). Our approach is to first create a graph-based representation of the structure's edges and corners (Figure 4), so the goal is reformulated as finding a continuous path through the graph that traverses all edges without crossing through an edge.

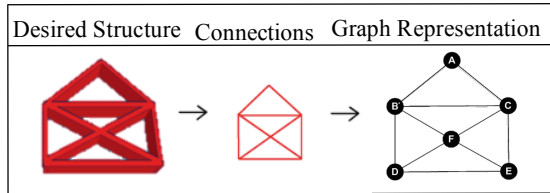


Figure 4. Graph-based representation of planar multi-link structure

For some graphs, there exists an Euler path, which is a path that traverses all edges exactly once. It is intuitive that in order to avoid repeating edges during a graph traversal, every edge that enters a node must also have a corresponding edge that exits the node. Hence, there are two cases in which a graph has an Euler path: There are 0 nodes of odd degree, so all nodes have an even number of edges, or there are 2 nodes of odd degree, where those 2 nodes serve as the starting and ending nodes of the traversal. Furthermore, if an Euler path does exist, there are already complete algorithms such as Fleury's algorithm and Hierholzer's algorithm [28] for finding it.

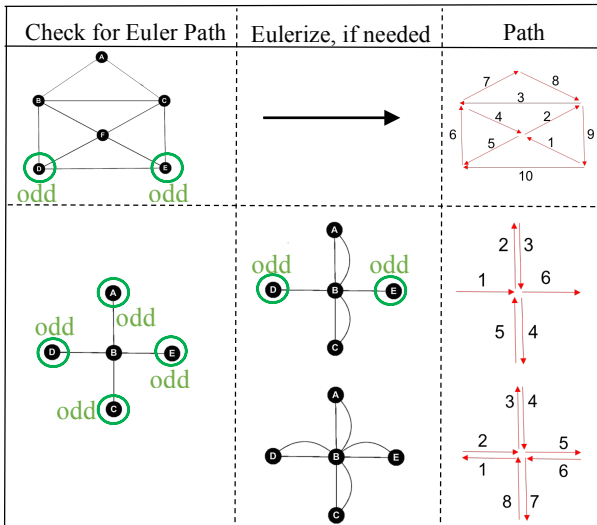


Figure 5. Euler paths traverse each edge exactly once. If a graph has either 0 or 2 nodes of odd degree, an Euler path exists. Otherwise, edges in the graph are repeated until an Euler path does exist.

Alternatively, for graphs that do not have an Euler path, it is possible to Eulerize the graph by doubling certain existing edges: Doubling a graph edge physically corresponds to retracing that edge with the ribbon. To show that repeating certain graph edges brings about the existence of an Euler path, we look at the worst case scenario of doubling every edge. Since every node in this resulting graph has even degree, the graph has an Euler path. This illustrates our claim that for any object represented by a connected graph, we can always find a path that visits all of its edges.

IV. CUSTOM RIBBON FOLDING MACHINE

The ribbon that we use is shown below as part of Figure 6; slits in the ribbon are created through the Smart Composite Microstructures (SCM) process [29], and these slits serve as joints in the structure. Initially, the ribbon is coated in glue; therefore, we heat the joint before folding it and letting it cool. Alternatively, joints can be left flexible by simply making a slit in the ribbon but not applying any glue to it. Our proof-of-concept ribbon folding machine consists of 3 motors: one feeds the ribbon into the machine and keeps it moving forward, one moves the end effector down and up to make contact with the ribbon, and one moves the end effector clockwise and counterclockwise to fold the ribbon. These motors, in combination with automated gluing and cooling, make up the overall folding sequence.

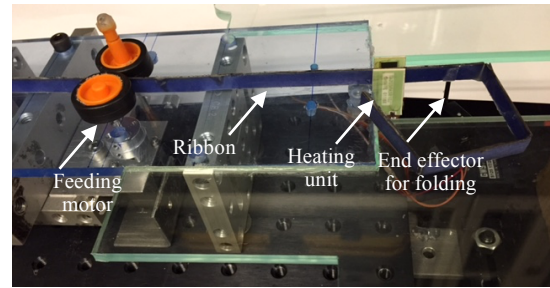


Figure 6. The ribbon folding machine

V. ALGORITHM

Arising from both the nature of ribbon folding and the minimalistic design of the folding machine, our problem definition imposes constraints: using one continuous ribbon, folding only from one end, progressing sequentially, and preventing self-intersections. This makes the path planning problem more rigorous than simply finding an Euler path through the graph; we must find a path that satisfies constraints at all stages of the folding. A guess-and-check approach for selecting which edges to repeat in order to meet constraints leads to an infinitely large state space. Below, we present an algorithm for which the output is the optimal sequence of folding angles and link lengths to create the structure.

A. Variation of A* Search

This algorithm formulates the task into a search problem where it creates a search tree in its attempt to find the optimal path. A* search is a path planning algorithm [30] that is commonly used in graph traversal. In our case, the start point is a state in which no edges have yet been traversed, and the end point is a state in which all edges have been traversed. Each node in this search tree is represented by a state, which holds information regarding the steps taken to get there, as

well as what remains. From the start node, we calculate the cost function for visiting each neighboring node and use that value to enter these nodes into the priority queue (our search tree). At each next step, we choose the node from the priority queue that has the least cost, and we take a step in that direction. Then, we add that node's neighbors to the priority queue and continue until a goal state is reached. A notable feature of this formulation is that in order to incorporate physical constraints, we only add a node onto the priority queue if making that fold does not violate constraints. To this end, we need to verify if a given path satisfies constraints: This subtask is explained in detail in the next section.

The cost function mentioned above is the sum of two values: distance travelled to get to the current node, which is the number of edges traversed thus far, and a heuristic estimate of the distance remaining to the goal, which is the number of unvisited edges. Since our heuristic function never overestimates the cost, this algorithm is guaranteed to return the optimal path. Note that we allow repeated edges in order to widen the subset of achievable structures. However, this algorithm inherently avoids meaningless loops because the cost function increases with the number of steps taken; edges are not repeated unless no better option exists.

B. Subtask: Constraint Violation

Given a list of edges in the order that they are traversed, we must verify whether a path violates the physical constraint. To reiterate, we define a collision to be when the shape must go through the building ribbon (or itself) in order to finish its rotation, as shown in Figure 7. The algorithm's input is each node's absolute coordinates, as well as the path's ordered edges. Algorithm 1 keeps track of a collection of (x, y) coordinates. One by one, it introduces each edge into the collection and calculates the angle it makes with the horizontal. By combining knowledge of the previous edge's angle and the current edge's angle, it finds the angle difference. Next, it simulates that fold by incrementally rotating all points while checking for collisions.

Algorithm 1: Check Constraint Violation: Self-intersections

Input:

- G , graph that lists the node connections
- N , x and y locations of each node
- E , list of ordered edges in the path which needs verification

Output:

True or *False*, for "does" or "does not" satisfy constraints

Do: For each $edge$ in E :

```

Set  $node1 = edge[0]$  and  $node2 = edge[1]$ 
Append node locations  $N[node1]$  and  $N[node2]$  to  $points$ 
 $new\_angle =$  angle that  $edge$  makes WRT horiz
 $old\_angle =$  angle that  $prev\_edge$  makes WRT horiz
 $angle\_difference = old\_angle - new\_angle$ 
 $folding\_angle = 180 - angle\_difference$ 
wrap  $folding\_angle$  between  $-180$  and  $180$ 
For each point in  $points$ :
    Calculate  $\theta$  from each node's  $(x,y)$  to the  $(0,0)$  pivot point
    Find the  $\theta$  closest to the building ribbon (based on turn direction)
     $\theta_{diff} =$  angular distance between  $\theta$  and the building ribbon
    If  $\theta_{diff} < abs(folding\_angle)$ , return False

```

**Note: tolerances and other small details omitted from this description

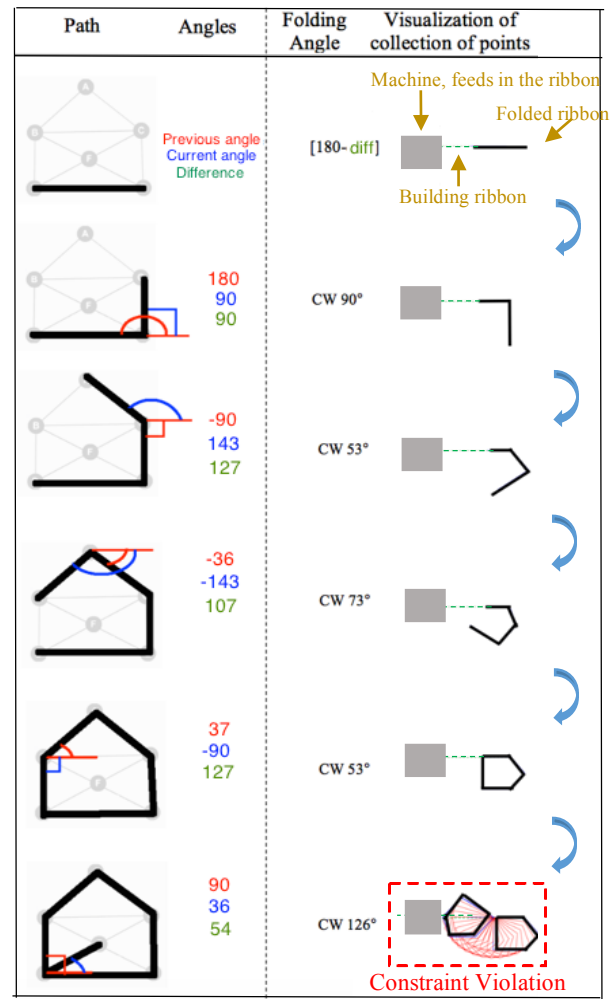


Figure 7. Illustration of Algorithm 1, which checks whether or not a given path violates physical constraints. From top to bottom, we incrementally add each edge's endpoints to the collection of points. After angle calculations, we rotate the points by the "folding angle." In the bottom right image, the folded shape crosses over the building ribbon during the last rotation, so the physical constraint is violated.

C. End Search: Answer both "can do" and "cannot do"

By following the procedure above of keeping nodes on a priority queue, tracking state costs, and monitoring edges, we can successfully find the optimal path for folding a ribbon into the desired shape. However, not every object contains a continuous and end-folded solution. If we work under the constraint of not wanting to repeat any edges, the search space is finite. In the general case, however, our current algorithm has an infinite search space, so if no valid path exists, there is no clear termination point for the search. Thus, we must find a condition that leads to the eventual end of the search, where no nodes remain on the search tree.

An intuitive way to think about the physical constraints explained above is to realize that the folding should generally begin inside of a shape and work its way outward. Once a region is closed off, the ribbon would need to transect itself to reach anything inside. Thus, we gain more intuition about which types of objects are end-foldable. We formalize this as being unable to enter any convex hull that has already been created as part of the traversal. This newly phrased constraint

helps greatly in the pruning of search trees by allowing the early termination of an entire branch, as shown below.

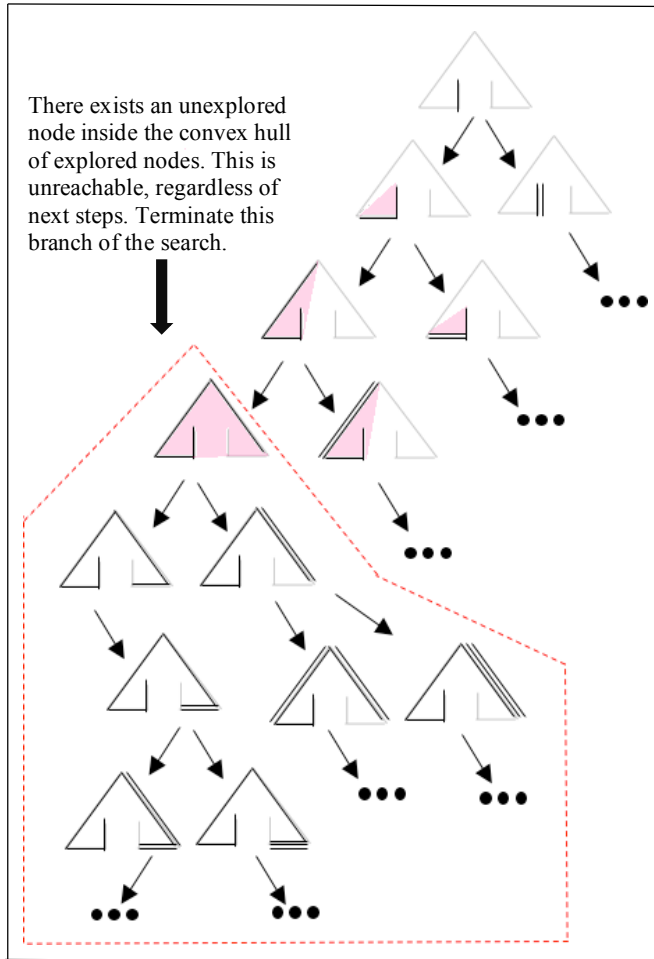


Figure 8. Using the notion of convex hulls to prune the search tree

The search trees are still endless because we allow repeated edges. Practically, the user would place a physical limit, stemming from weight or strength restrictions, on the number of repeats that are allowed for any given edge. Mathematically, however, we seek a logical bound on the number of repeats for any given edge; such a bound would eventually end the search by causing every branch of the tree to be terminated, even if no valid solution exists.

Claim: For an optimal path under our problem definition of sequential and continuous folding, the number of repeats on any edge is less than or equal to the total number of edges.

Proof: Consider a desired structure with N edges, where e_k is the k^{th} edge and r_k is the total number of folds performed on e_k . After exploring the graph's first unexplored edge at the first time step, the max number of repeats on any edge is 1.

At every following time step, there are 2 options. The first option is that there exists a valid unexplored edge, in which case it should be explored. Thus, r_k for that edge is 1, and r_k for all other edges remain the same. If that is not the case, then the second option is that there exists at least one valid explored edge. From this current x_t , the goal is to reach an unexplored edge u . Regardless of whether that path is short or long, each e_k is traversed once along the way. Note that

traversing the same r_k more than once while aiming for a certain goal makes all of the steps in between that repetition useless. Then, that path would not be optimal. However, if A^* found this path, then no other more optimal path exists. Thus, in the quest for each unexplored edge u , no edge gains more than 1 additional repeat. Given that there are N total edges, then no edge should have more than N repeats at any point in time during the search. Therefore, the claim is true.

Algorithm 2: Optimal Path for Ribbon Folding, using A^*

Input:

G , graph that specifies the node connections
 N , x and y locations of each node
 S , starting state: as specified, or a variable to search over

Output:

L , ordered list of link lengths in the optimal path
 A , ordered list of folding angles for the optimal path

Do:

```

open_set = [] //unexplored states
add  $S$  to open_set
while open_set isn't empty:
    current_state = state from open_set w/ lowest cost
    if current_state.unexplored_edges is empty:
        print info //found the optimal path from  $S$  to the goal
        return  $L, A$ 
    remove current_state from open_set
    look at current_state.path_history for curr_node
    for each neighbor of curr_node:
        edge = curr_node to neighbor
        if (adding edge doesn't violate constraints): //use Algorithm 1
            if (not too many repeats on any edge):
                calculate convex hull of explored states
                if (enough non-collinear points to calculate a convex hull):
                    if (no unexplored nodes are inside the convex hull):
                        remove edge from current_state.unexplored_edges
                        current_state.cost = # explored + # unexplored edges
                        add current_state to open_set
    return //no feasible path exists for the given structure

```

VI. RESULTS AND ANALYSIS

Figures 9 and 10 show the resulting optimal folding sequence from our path planning algorithm for an envelope shape, a nontrivial geometric shape, and a truss. We have not optimized the algorithm for time because that was not a top priority in the exploration into ribbon folding; this can be a relatively straightforward future improvement. Even so, the time taken for the algorithm to output the path for the structures shown above are on the order of seconds on a standard laptop computer. As shown in Table 1, pruning the search tree based on the convex hull constraint leads to a significant reduction in the size of the search tree, or the number of nodes explored. Furthermore, for shapes that are impossible to be end-folded in a sequential and continuous manner, the bound on repeated edges makes it possible for the algorithm to terminate and relay this information.

As is apparent when comparing the envelope structure (11 nodes explored) with the truss structure, the number of nodes explored increases rapidly with the number of graph edges. Looking at the geometric shape vs. the truss shape, we see that when a shape requires more repeated edges, the number of explored nodes drastically increases: At each step, it must

explore all options of lower cost before allowing a repeated edge. Although this algorithm is only used on the planar subset of structures, it already surpasses human capability. Especially for objects with a nontrivial number of edges, it is intractable for a human to visualize thousands of rotations and manually navigate a rapidly expanding search space.

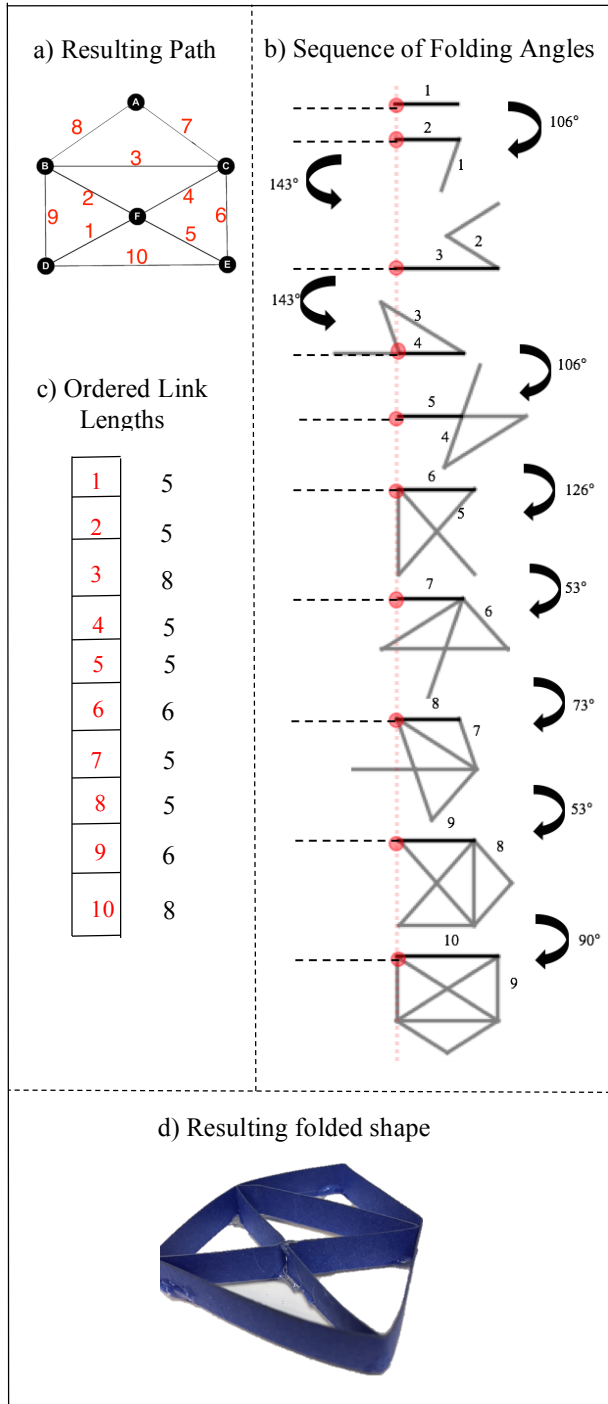


Figure 9. The output of the path planning algorithm for the envelope shape. The black edges in the visualization of folding angles represents the most recent edge, and each red circle indicates the axis of rotation between each step.

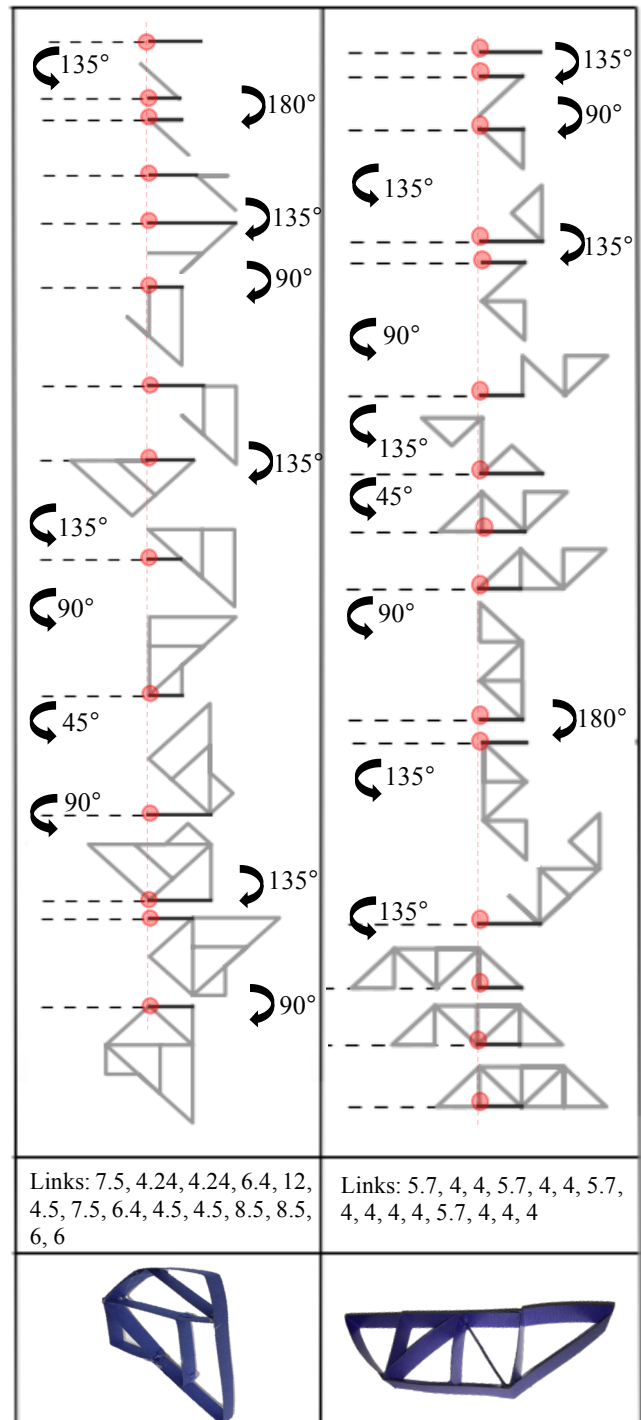


Figure 10. From top to bottom: Visualization of resulting folding angles, resulting list of ordered link lengths, and resulting folded shape. The geometric shape is on the left, and the truss structure is on the right.

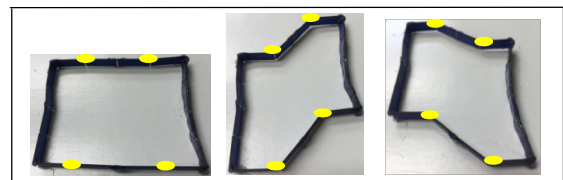
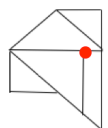
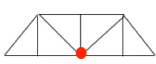

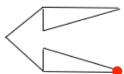
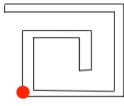


Figure 11. Addition of flexible slits (yellow) to allow movement in the otherwise rigid four-bar linkage, while still allowing for rigid corners to keep the shape intact during folding. Note that joint placement is exaggerated here (placed far from rigid joints) for illustrative purposes.

TABLE I. IMPROVEMENT WITH PRUNING AND BOUNDING

Structure	Before pruning & placing bound		After pruning & placing bound	
	#Nodes Explored	Path Found?	#Nodes Explored	Path Found?
	757	Y	309	Y
	8832	Y	6476	Y
	∞	N/A	25	N
	∞	N/A	154	N
	∞	N/A	29	N

a. The red dot indicates starting point for this data collection

A. Extensions to Movable Structures: Creating Free Joints

As mentioned in the introduction, one benefit and interesting capability of ribbon folding is the ability to have flexible joints. Since each slit in the ribbon is coated with glue and then bent into the desired angle, the joints that we have seen thus far have been relatively rigid. Figures 11 and 12 show examples of attaining free joints by using additional slits in the ribbon, which are not folded and therefore not coated in glue. These extra flexible slits are added adjacent to the rigid object corners. The fixed joints allow the structure to retain its shape during the folding, and the extra slits allow movement around the fixed joints to mimic free joints.

While seemingly straightforward to add joints near existing ones, this technique becomes murky for complex shapes. We plan to formally explore this in the future, but we leave the reader with thoughts about decomposing structures into a representation of fixed and free joints. Note that in our case, free joints are approximated by straight flexures in the small angle approximation. Thus, having repeated edges near a free joint would result in a different type of behavior. To allow the creation of structures with free joints, we must add a layer

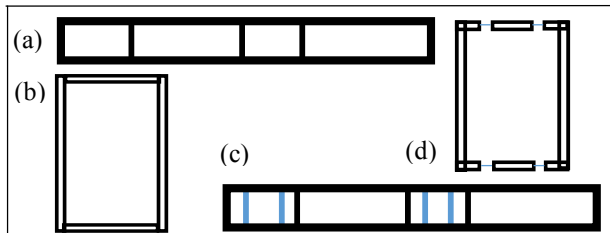


Figure 12. (a) The ribbon template for (b) the rigid four-bar structure, and (c) the ribbon template for (d) a four-bar structure with free joints


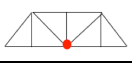



onto our algorithm which intelligently decomposes the desired structure, separates rigid regions from non-rigid regions, and specifies which edges can or cannot be repeated.

VII. CONCLUSION

We have presented a novel path planning algorithm for the ribbon folding of planar objects. By creating graph-based representations of structures, we use Euler paths to show that for any object represented by a connected graph, there exists a continuous path which visits all of its edges. The path planning algorithm considers the constraints imposed by our minimal folding machine, and it results in the optimal sequence of physically feasible ribbon folds for creating that structure. This folding technique provides the user a tool which is simple, but also uses continuity in the path planning to allow efficient structure creation, rather than simply constructing small components, which the user would then need to bring together in an additional assembly step. While this definitely provides benefits, note that at least some level of hierarchical decomposition has to occur in most tasks, i.e. the user would not print out an entire house at once.

While this amalgamation of ideas for the development of a path planning algorithm for ribbon folding is novel, there are many improvements and further tasks that remain. In the worst case of an unbounded search space, such as ours, the complexity of A* is b^d , where b is the branching factor that indicates the average number of successors per state (i.e. degree of the node), and d is the length of the shortest solution, which is unknown and potentially large in our case of allowing repeated edges. To improve the tractability and scalability of this algorithm for larger graphs, we must explore other search techniques that provide a tradeoff between efficiency and optimality. In the future, we also aim to find a tighter bound on the number of repeats allowed on any given edge. For instance, if we begin the path planning algorithm on the maze structure at the inner-most node, then 64,341 nodes are visited before the search can end and report that it is not possible. Since the total number of edges in that object is 14, it is expected that such a high number of repeats per edge takes a long time to achieve; the search does not end

TABLE II. RESULTS OF PATH PLANNING ALGORITHM

Structure	# Vertices	#Repeated edges	Avg. Degree	Time taken
	9	2	2.8	0.59 sec
	8	2	3.3	20 sec
	7	2	3.1	1.3 sec
	13	1	3	1.2 sec
	8	1	3.3	1.6 sec

a. Runtime was not a high priority: Algorithm is not fully optimized for that yet.

quickly. Therefore, finding a lower bound on the number of repeats allowed per edge would achieve tremendous speed-up in run time. Furthermore, there still exist a number of areas that have not yet been addressed algorithmically, such as structures including curves, surfaces, and higher dimensions.

As we head toward such a minimal universal fabricator, ribbon folding shows promise of novelty, generality, and applicability. Particularly in robotics, ribbon folding can serve as an alternative to current manually fabricated systems which takes several weeks or months of effort to create. This further suggests applications in robotic self-repair or shape adaptation. A robot with limited access to human intervention and assembly resources could use this on-demand fabrication to adapt body shape, create new parts, or change its end effectors as a function of the task and terrain.

Other interesting future directions to explore include using different materials for the building primitive, incorporating curvature and additional types of joints, and combining analysis of object strength with the algorithm for the object creation. The user could, for example, deliberately request more repeated folds on certain edges for strength purposes. Products that display such diversity and utility, built from nothing more than a primitive material, could revolutionize on-demand manufacturing. Just as protein folding uses basic building blocks to construct a multitude of chemicals that form the basis of all organic life, ribbon folding aims to use ribbons to solve the structure formation problem.

ACKNOWLEDGMENT

The authors would like to thank David Fridovich-Keil for insightful discussions.

REFERENCES

- [1] P.J. White, C.E. Thorne, and M. Yim. "Right angle tetrahedron chain externally-actuated testbed (RATChET): A shape-changing system." *Proceedings of IDETC/CIE*. 2009.
- [2] "StateMaster - Encyclopedia: Linkage (mechanical)." *RSS Stats*. Nation Master. 27 Jan. 2016.
- [3] "Plane Trusses." *Linear Analysis of Skeletal Structures* (2004): Auburn University College of Engineering. 27 Jan. 2016.
- [4] "Honeycomb Core for Lighting." *Honeycomb Core for Lighting*, 2016.
- [5] D. Haldane, C. Casarez, J. Karras, J. Lee, C. Li, A. Pullin, E. Schaler, D. Yun, H. Ota, A. Javey, R.S. Fearing. "Integrated Manufacture of Exoskeletons and Sensing Structures for Folded Millirobots." *Journal of Mechanisms and Robotics* 7.2 (2015): 021011.
- [6] S.C. Goldstein and T. Mowry. Claytronics: A scalable basis for future robots. *Robosphere*, November 2004.
- [7] K.C. Cheung, E.D. Demaine, J.R. Bachrach, and S. Griffith. Programmable assembly with universally foldable strings (moteins). *Robotics, IEEE Transactions on*, 27 (4): 718–729, Aug. 2011.
- [8] L. Wang, M. Plecnik, R.S. Fearing. "Robotic folding of 2D and 3D structures from a linear ribbon." *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016
- [9] K. Suzuki, I. Shimoyama, H. Miura, and Y. Ezura. "Creation of an insect-based microrobot with an external skeleton and elastic joints," in *Proc. 1992 IEEE Micro Electro Mechanical Systems. An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots*, Traravunde, pp. 190-195.
- [10] E. Shimada, J. A. Thompson, J. Yan, R. Wood, and R. S. Fearing. "Prototyping millirobots using dextrous microassembly and folding," in *Proc. 2000 ASME Int. Mechanical Engineering Congr. Expo.*, Orlando, pp. 933-940.
- [11] J.P. Whitney, P.S. Sreetharan, K.Y. Ma, and R.J. Wood. 2011, "Pop-Up Book MEMS," *J. Micromech. Microeng.*, 21(11), p. 115021.
- [12] S. Felton, M. Tolley, E. Demaine, D. Rus, and R. Wood. 2014, "A Method for Building Self-Folding Machines," *Science*, 345(6197), pp. 644–646.
- [13] L. Lu, and S. Akella. 1999. Folding cartons with fixtures: A motion planning approach. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1570–1576.
- [14] C.D. Onal, R.J. Wood, and D. Rus. "Towards printable robotics: Origami-inspired planar fabrication of three-dimensional mechanisms." *Robotics and Automation (ICRA)*, IEEE 2011.
- [15] H.C. Greenberg, M.L. Gong, S.P. Magleby, and L.L. Howell. "Identifying links between origami and compliant mechanisms." *Mech. Sci* 2.2 (2011): 217-225.
- [16] H.C. Greenberg. "The Application of Origami to the Design of Lamina Emergent Mechanisms (LEMs) with Extensions to Collapsible, Compliant and Flat-Folding Mechanisms." 2012. *All Theses and Dissertations*. Paper 3210.
- [17] D.J. Balkcom, and M.T. Mason. "Robotic origami folding." *The International Journal of Robotics Research* 27.5 (2008): 613-627.
- [18] A. Hoover, and R.S. Fearing. "Fast scale prototyping for folded millirobots," *IEEE Int. Conf. on Robotics and Automation*, pp. 886–892, May 2008.
- [19] R. Sahai, E. Steltz, and R.S. Fearing. "Carbon fiber components with integrated wiring for millirobot prototyping." *Robotics and Automation, IEEE Int. Conf. IEEE*, 2005.
- [20] R. Sahai, J. Lee, and R.S. Fearing. "Semi-automated micro assembly for rapid prototyping of a one dof surgical wrist." *IEEE/RSJ Int. Conf. on Intell. Robots and Systems*, 2003.
- [21] G.S. Chirikjian, and J.W. Burdick. "A hyper-redundant manipulator." *IEEE Robotics and Automation Magazine* 1.4 (1994): 22-29.
- [22] N.G. Cheng, M.B. Lobovsky, S.J. Keating, A.M. Setapen, K.I. Gero, A.E. Hosoi, K.D. Lagnemma. "Design and analysis of a robust, low-cost, highly articulated manipulator enabled by jamming of granular media." *Robotics and Automation IEEE Int. Conf.* 2012.
- [23] E.D. Demaine. *Folding and unfolding*. Diss. University of Waterloo, 2001.
- [24] E.D. Demaine. "Folding and unfolding linkages, paper, and polyhedra." *Discrete and Computational Geometry*. Springer Berlin Heidelberg, 2000. 113-124.
- [25] E.M. Arkin, S.P. Fekete, and J. Mitchell. "An algorithmic study of manufacturing paperclips and other folded structures." *Computational Geometry* 25.1 (2003): 117-138.
- [26] R. Connelly, E.D. Demaine, G. Rote. Every polygon can be untangled, in: *Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci.*, 2000, pp. 432–442.
- [27] I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning, in: *Proc. 41st Annu. IEEE Sympos. Found. Comput. Sci.*, 2000, pp. 443–453.
- [28] L. Lesniak, and O.R. Oellermann. "An eulerian exposition." *Journal of Graph Theory* 10.3 (1986): 277-297.
- [29] R.J. Wood, S. Avadhanula, R. Sahai, E. Steltz, and R.S. Fearing. "Microrobot Design Using Fiber Reinforced Composites." *Journal of Mechanical Design*, 2008
- [30] P.E. Hart, N.J. Nilsson, and B. Raphael. "A formal basis for the heuristic determination of minimum cost paths." *Systems Science and Cybernetics, IEEE Transactions on* 4.2 (1968): 100-107.