Unfolding and Flat Layout Design of Non-Manifold 3D Folded Structures

Kang Tai[1], Wei Liu[2] and Georg L. Thimm[3]

[1]Nanyang Technological University, mktai@ntu.edu.sg
[2]Nanyang Technological University, liuwei@pmail.ntu.edu.sg
[3]Nanyang Technological University, mgeorg@ntu.edu.sg

ABSTRACT

There are practical applications in the sheet metal, packaging and various other industries for research into the automatic development of the flat layout or pattern that can be folded into some desired 3D folded structure. The relevant techniques developed in earlier work are based on generating spanning trees of the face adjacency graph (the graph that represents the connectivity among the faces of the folded structure) since any spanning tree is a potential unfolded flat layout of that structure. However, complications in the structure, such as situations where more than two faces are joined at one common edge, pose problems to the spanning tree unfolding methodology that can lead to incorrect results. This work examines these problems through the connectivity and topological representation issues involved when splitting these common edges, and proposes a strategy to handle such complications by developing algorithms to detect topologically invalid spanning trees. These new algorithms are incorporated into a previous procedure developed by the authors, and the overall methodology is implemented on a computer program and applied to unfold four example structures to generate their corresponding flat layouts.

Keywords: unfolding, folded structure, flat pattern development, topological representation

1. INTRODUCTION

There are practical applications in industry for the study of how to design a flat layout/pattern of sheet material that can be folded into some desired 3D geometric structure. For example, sheet metal products are often produced by bending (folding) a flat pattern of sheet metal, with perhaps some welding (joining) of its edges. Another application is in the field of packaging, where corrugated paperboard is folded into a 3D folded structure of some desired geometry. Usually having more complex geometries than the ubiquitous carton boxes, these structures are actually used inside carton boxes as packaging cushions (also known as packaging buffers) for protective packaging of products, or used as partitions (also known as inserts) for holding/segregating multiple products or components within a package. With manufacturers coming under the impact of stronger environmental legislation, these protective packaging components made of paper are increasingly being considered as viable alternatives to the traditional use of packaging cushions made of polymer foams.

The design of a flat layout (of paperboard) that can be folded into some desired 3D folded structure is a challenging problem that requires creativity/ingenuity, experience and laborious trial-and-error on the part of the designer. As paperboards are manufactured as flat sheets, a 3D board structure (with some complex topology/shape to hold the product that is being packaged) has to be produced by folding from a suitable flat paperboard pattern/layout. Different products with varying geometry and complexity will require much innovation from the designer to create the various 3D folded packaging structures and their corresponding flat layouts. Hence the objective of this research is to aid the designer by developing a methodology to generate solutions to the following question : Given a 3D folded structure, what are the flat layouts that can be folded into that structure? As these 3D structures are composed of piecewise planar faces and are formed by folding some flat sheet of material, their topological configurations are of the main concern and therefore their geometries can be treated in a schematic form with the faces having zero thickness and ignoring any bending radius at the folded edges. As a result, these folded structures are usually non-manifold objects.

Research work related to the folding of sheet or thin-walled materials include some of the mathematical

studies of origami [3][7] and even computational algorithms developed for generating the crease pattern for folding into various origami designs [10]. However, in origami, the sheets of paper are usually folded into objects which are flat or piecewise flat structures. Another related work is the geometric modelling and simulation of pop-up books [11]. However, the mathematical formulations in such work focus mainly on shape and motion information of the pop-up mechanisms and not on the topology. Foldable structures have also been treated and analyzed as mechanisms by Dai and his co-workers [5][15] and Pellegrino and his co-workers [6][8][20]. Another interesting work is that of Kling *et al*. [9] where geometric theories have been developed for generating doubly-periodic folding patterns within one sheet. The resulting structures may be considered as sheets with discrete raised patterns which give the sheet desirable properties suited for a variety of purposes and yet is economical to produce since the manufacturing process is essentially folding only.

Agarwal *et al*. [1] proposed a star unfolding technique for unfolding convex polyhedrons by selecting some point on the face of the polyhedron and cutting from that point to various vertices of the structure via the shortest paths. Biedl *et al*. [4] also proposed algorithms to unfold two classes of orthogonal polyhedra by cutting across faces. However, cutting/breaking a face into two or more faces compromises the strength and integrity of the structure and so from a mechanical point of view, cutting along common edges to unfold a structure (also known as edge unfolding) is the preferred outcome. Bangay [2] proposed heuristic algorithms to iteratively construct the flat layout by successively adding faces to the layout. However, the procedures were demonstrated only on the type of 3D polyhedral models typically used for surface representation of objects in virtual reality environments.

As sheet metal forming is an important manufacturing process, much work has also been done on the geometric reasoning of sheet metal (thin-walled) objects to aid in their design when they are of fairly complex geometries. Lin and Yang [12][13] and Lipson and Shpitalni [14] developed mathematical formulations to analyze/relate the topological properties of these 3D thin-walled objects and their corresponding flat patterns. Using formulae to compute the number of seams (i.e. the common edges of a 3D thin-walled object that need to be split in order to unfold the object into a flat layout), Lin and Yang [12][13] then applied a breath-first search process to assign the required number of seams (splits) to the face adjacency matrix of the object to generate the development matrices (each of which is the face adjacency matrix of a potential flat layout). These development matrices are then checked for validity (i.e. the flat layout is a single connected piece, and there are no overlapping faces). In principle, any spanning tree of the face adjacency graph (that represents the topology of the folded object) is a potential flat layout of the object. Based on this principle, Shpitalni and Lipson [19] proposed an approach that applies the A* algorithm to find a maximum weighted spanning tree that represents a valid (with non-overlapping faces) and optimal (according to some prescribed optimality criteria) flat layout. Liu and Tai [16] adopted the approach of simply enumerating all possible spanning trees, geometrically unfolding each of them and checking for overlapping faces. This approach has the flexibility of allowing a subsequent optimization process where the optimality criteria may be of a global nature (i.e. the criteria can only be verified from the complete flat layout) or allowing a subsequent interactive query/selection process by the designer.

However, none of the above techniques adequately address a complication found in non-manifold 3D folded structures : a situation where more than two faces are joined at one common edge. This is a feature quite often present in the design of non-manifold thin-walled objects and in this work, such common edges are referred to as hyper-common edges. These edges pose difficulties to all the previously described techniques in their geometric representation and the correct way in which they should be split to produce a valid flat layout. The aim in this work is therefore to develop a versatile methodology and implement it into a computer program that can generate all the possible single-piece flat layouts that can be folded into some given manifold or non-manifold 3D folded structure (including any with hyper-common edges). In this work, all common edges of the folded structure are restricted to be straight lines, but there is no restriction on the convexity of the structure or the shape of the faces (i.e. they can be any polygonal shape : triangular, quadrilateral, pentagonal, etc.). The rest of this paper will explain the original algorithm from [16] for generating flat layouts, the difficulties/complications posed by hyper-common edges, the methodology for handling such edges, and the unfolding results of four example problems.

2. BASIC UNFOLDING AND FLAT LAYOUT GENERATION PROCEDURE

This current work is an extension of the basic flat layout generation procedure previously developed in [16] and hence that procedure is briefly reviewed here. The procedure was implemented as a C++ code that reads in a B-rep model data of the 3D structure that is to be unfolded, automatically generates the corresponding face

adjacency graph (FAG) that models the topology/connectivity of the faces of the folded structure, applies a compact output method [18] to enumerate all possible spanning trees of the graph, computes the required transformation of the faces onto each flat layout (as defined by each tree), applies an overlapping detection algorithm to check for overlapping of faces within each layout, and then outputs only those layouts with no overlapping. A simple example to illustrate the overall concept is shown in Fig. 1, where a 3D L-shape structure with seven faces is unfolded and Fig. 1(c) shows just one possible flat layout.

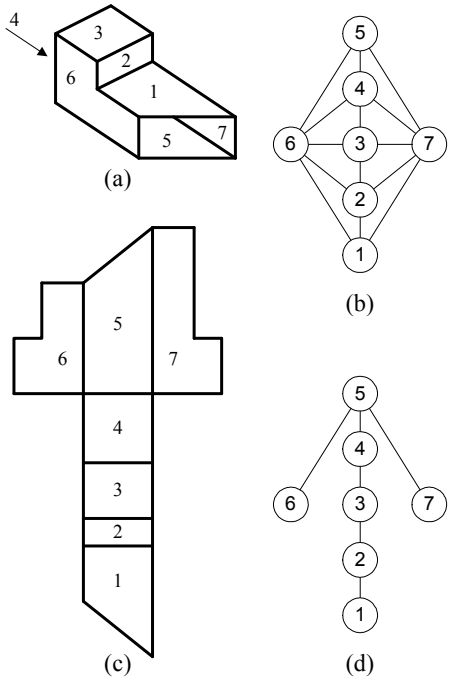

Fig. 1. (a) 3D folded structure. (b) FAG of folded structure. (c) One sample flat layout. (d) Spanning tree corresponding to sample flat layout.

3. PROBLEMS ARISING FROM NON-MANIFOLD OBJECTS AND HYPER-COMMON EDGES

In this section, a few key definitions are first made, and then the problems that arise from attempting to unfold non-manifold objects (especially those with hyper-common edges) are identified and discussed.

3.1 Definitions

To aid further discussion, it is necessary to make some key definitions.

Nodes and links : In relation to graphs, the vertices and edges of a graph will be referred to as nodes and links, respectively. The use of the terms 'node' and 'link' in this paper will avoid confusion with the terms 'vertex' and 'edge' which are the commonly understood terms used in geometry and geometric modelling.

Free, common and hyper-common edges : A free edge is an edge that is exclusively adjacent to one face only. A common edge is an edge that is adjacent to exactly two faces. A hyper-common edge is an edge that is adjacent to three or more faces.

Hyper-common edge set and its sequence : The hyper-common edge set of a specific hyper-common edge is the set of all faces adjacent to this hyper-common edge. The sequence of a hyper-common edge set is a sorted hyper-common edge set. The first element in the sequence can be any of those faces arbitrarily selected. However, the subsequent elements are all the remaining faces sorted in either clockwise or counter-clockwise order.

Manifold and non-manifold structures: By definition, a manifold structure is a topological space where every point has a neighborhood topologically equivalent to an open disk of E^2, where E^2 is the two-dimensional Euclid's Space [17]. This definition asserts that each manifold structure requires that (1) all edges separate exactly two faces and (2) all vertices are surrounded by a single circuit of faces. Those structures that fail to satisfy these two criteria are considered non-manifold. This paper focuses primarily on non-manifold structures that break the first criterion, namely that some edges of the given 3D structure can be adjacent to just one face or to more than two faces.

3.2 Problem I: Different Structures Represented By Same FAG

A FAG alone cannot fully define the topology of a non-manifold 3D structure. As an example, Fig. 2(a), (b) and (c) show three different non-manifold 3D structures (and 2(c) even contains a hyper common edge). However, they have the same FAG as shown in Fig. 2(d). This suggests that enumerating spanning trees from a FAG may not generate the correct flat layout since there is an ambiguity whenever a feature like that of Fig. 2(a), (b) or (c) is present in the folded structure. Fortunately, however, this does not pose a problem as long as the B-rep data of the folded structure is available and utilized in the geometric unfolding/transformation of the faces onto the planar flat layout (as in the basic procedure developed previously and explained in Section 2). This is because the topological information in the B-rep data will correctly identify the common edge that is to be split to unfold the structure. However, this procedure is not entirely workable in the case of hyper-common edges

like in Fig. 2(c), and it is a problem that will be examined in more detail in the following section.

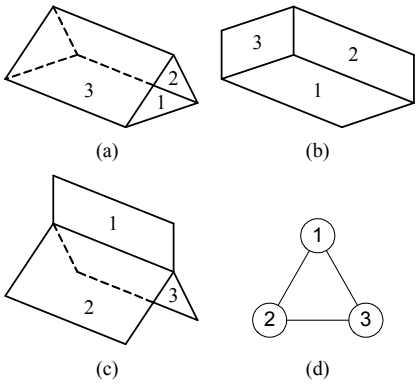

Fig. 2. (a) Triangular tubular structure. (b) Corner structure. (c) Structure with three faces joined at an edge. (d) Same FAG for all three structures.

3.3 Problem II: Incomplete Splitting of Hyper-Common Edge

The presence of hyper-common edges in a folded structure poses a problem to the basic procedure of obtaining a flat layout by generating a spanning tree from the FAG of the structure. This is illustrated in Fig. 3(a) which shows a folded structure featuring one hyper-common edge (with hyper-common edge set {1,2,3}), with 3(b) showing one possible spanning tree that gives rise to a correct flat layout (the dashed lines show the links that are removed from the original FAG of the folded structure to reduce it to a spanning tree). However, Fig. 3(c) also shows a spanning tree but one that is topologically invalid because it represents an erroneous situation where the hyper-common edge has not been split completely since it is impossible for both faces 1 and 3 to be adjacent to face 2 and yet are not adjacent to each other. It will therefore not be an acceptable result for a flat layout.

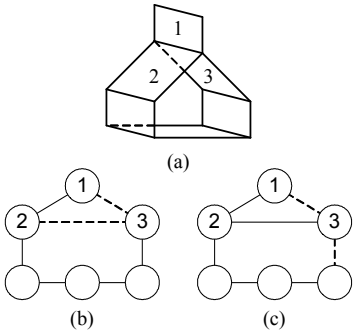

Fig. 3. (a) Structure with hyper-common edge. (b) Topologically valid spanning tree. (c) Topologically invalid spanning tree.

3.4 Problem III: Incorrect Splitting of Hyper-Common Edge

If a hyper-common edge set contains four or more faces, the sequence of the set is a factor to be considered because it restricts the possible ways in which the hyper-common edge can be split up. Consider the structure in Fig. 4(a) featuring a hyper-common edge with the sequence of faces being {2,5,3,6}. The structure has a total of six faces with a FAG as shown in Fig. 4(b). Fig. 4(c) shows one flat layout with its corresponding spanning tree shown in 4(d). However, this layout cannot be folded into the 3D structure because there will be a clash/intersection among the edges where the hyper-common edge used to be since the adjacent faces 5 and 6 have to pass through adjacent faces 2 and 3. In other words, not every spanning tree leads to a layout that can be folded in the required manner because of the possibility of incorrect splitting arrangement among the hyper-common edge set. Such incorrect trees are also considered topologically invalid.

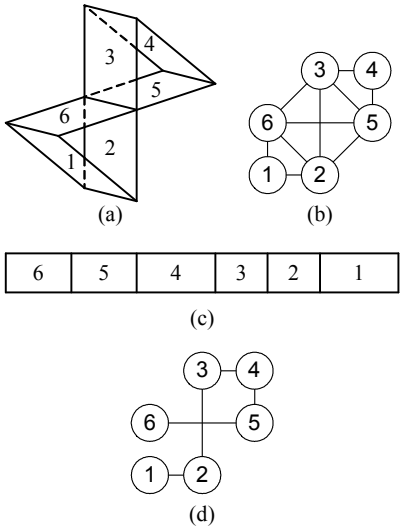

Fig. 4. (a) Structure with hyper-common edge. (b) FAG of structure. (c) Flat layout that cannot be folded into structure. (d) Topologically invalid spanning tree.

4. SOLUTION METHODOLOGY

The overall approach adopted in this work is to employ the basic flat layout generation procedure explained in Section 2 but with the additional capability to detect spanning trees with problems II and III, and eliminate such trees from the results before performing the geometric unfolding, i.e. the transformation of the faces onto a planar flat layout. An overlapping detection algorithm is then applied to each unfolded flat layout to check for overlapping faces within the layout, and those

with overlapping will thus also be discarded (note that a topologically valid spanning tree does not guarantee that it is also geometrically valid, i.e. without overlapping faces).

The following sections will describe the computational algorithms developed to automatically determine the sequence of a hyper-common edge set, and the procedures to detect topologically valid/invalid spanning trees at locations where the hyper-common edges used to be. The sequence of the set has to be determined because this information is required for checking the validity of the spanning tree.

4.1 Determining the Sequence

Consider a hyper-common edge denoted by the vector $\mathbf{e}_h$ and its corresponding edge set $H(\mathbf{e}_h)=\{1,2,3,4\}$ consisting of four faces as shown in Fig. 5. The sequence $S(\mathbf{e}_h)$ of this set is to be determined automatically. The sense of each of the unit normal vectors $\mathbf{n}_1$ to $\mathbf{n}_4$ of the faces of this set can be consistently defined (as shown in Fig. 5) since all the faces share the same edge $\mathbf{e}_h$ and the direction of $\mathbf{e}_h$ is fixed. It should also be noted that all normal vectors of the faces sharing a same hyper-common edge are coplanar. This can be proven by moving all the normal vectors onto the hyper-common edge as well as onto the same point along that edge. Since all the normal vectors must also be perpendicular to that edge, it follows that they now lie along the same plane. Since all the normal vectors are coplanar, those normal vectors from Fig. 5 can be drawn on the same plane with their corresponding orientations with respect to one another as shown in Fig. 6.

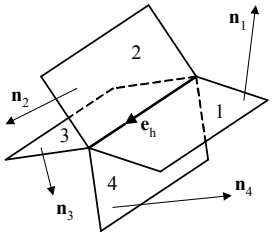

Fig. 5. Faces of a hyper-common edge set.

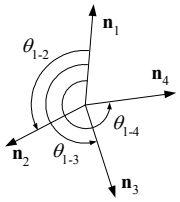

Fig. 6. Orientation of normal vectors of the hyper-common edge set.

Based on the angles between the normal vectors, the sequence of the faces can be deduced. Any one of the faces can be chosen as the start of the sequence and in this illustration, face 1 is chosen. The angles (as shown in Fig. 6) between face 1 and any other face i can be computed as follows :

$$\theta_{1-i}=\begin{cases}\cos^{-1}\left(\mathbf{n}_1\cdot\mathbf{n}_i\right) & \text{if } \left(\mathbf{n}_1\times\mathbf{n}_i\right)\cdot\mathbf{e}_h\geq 0\\ 2\pi-\cos^{-1}\left(\mathbf{n}_1\cdot\mathbf{n}_i\right) & \text{if } \left(\mathbf{n}_1\times\mathbf{n}_i\right)\cdot\mathbf{e}_h<0\end{cases} \quad (1)$$

According to Eqn. (1), each angle $\theta_{1\text{-}i}$ will correctly range from 0 to 2π. The sequence can then be determined by sorting according to the size of the angles and, in this illustration, the sequence is of course $S(\mathbf{e}_h)=\{1,2,3,4\}$.

4.2 Detecting Topological Validity

Consider a hyper-common edge set consisting of six faces with a sequence $S(\mathbf{e}_h)=\{1,2,3,4,5,6\}$ as shown in Fig. 7(a) together with its FAG. Note that this set may be just one part of a larger structure. Fig. 7(b) shows one possible way in which the hyper-common edge has been split and its corresponding spanning tree (actually only a portion of the overall layout's spanning tree because the nodes and links related to the rest of the structure are not shown). Notice that Problem II is present in this result because there is still a hyper-common edge set {2,3,6} remaining, while Problem III is also present because adjacent faces 1 and 4 have to intersect adjacent faces 2, 3 and 6 if the layout is to be folded into the structure. Such a spanning tree will therefore be considered invalid.

A possible valid way of splitting the hyper-common edge is shown in Fig. 7(c) together with its corresponding spanning tree (note that face 1 and face 3 are not connected to any other faces from among the hyper-common edge set but may instead be connected to some other faces from the rest of the structure not shown). To detect the validity of any given spanning tree, the nodes and links pertaining to any particular hyper-common edge set $H(\mathbf{e}_h)$ can first be placed along a circle in order according to its sequence $S(\mathbf{e}_h)$, with all links drawn as straight lines. To illustrate, the valid spanning tree from Fig. 7(c) (which is already in its correct sequence) is placed within the dashed circle as shown in Fig. 8. Based on such an arrangement, it is now claimed that any spanning tree is valid only if the following two conditions are satisfied :

1. No node along the circle is incident on more than one link.
2. No link intersect any other link within the circle.

The first condition eliminates Problem II because any node that is incident on two or more links represents an incomplete splitting of the hyper-common edge. This

condition also implies that, in a valid spanning tree, the maximum allowable number of links among the hyper-common edge set is $c_{\max}$, given by

$$c_{\max} = \left\lfloor \frac{n(\mathbf{e}_h)}{2} \right\rfloor \quad (2)$$

where $n(\mathbf{e}_h)$ is the number of faces in the hyper-common edge set and $\lfloor\ \rfloor$ is the floor function. The second condition eliminates Problem III because any intersection of links represents an incorrect splitting (i.e. an intersection of faces when folding). The algorithm to detect this condition can be based on traversing each node in sequence and applying a form of last-in-first-out mechanism to check the links incident to each node. If a structure has more than one hyper-common edge, then both conditions must be satisfied for each and every such edge before a spanning tree is certified valid.

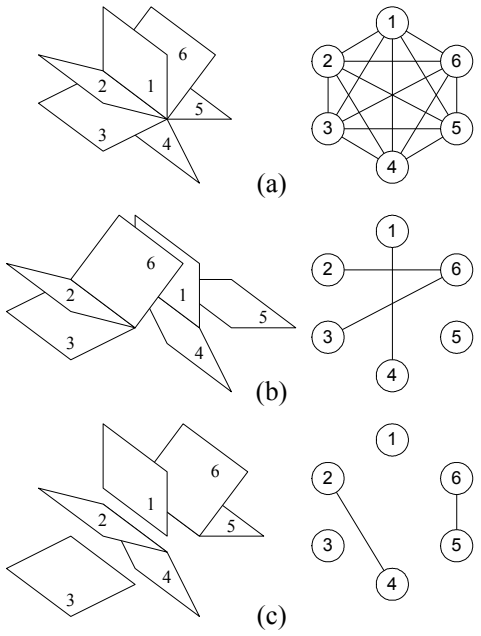

Fig. 7. (a) Hyper-common edge set and FAG. (b) Invalid splitting arrangement and spanning tree. (c) Valid splitting arrangement and spanning tree.

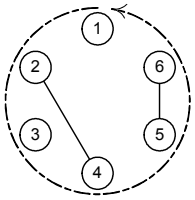

Fig. 8. Topologically valid spanning tree.

5. RESULTS

The overall procedure has been implemented in a C++ program running on a personal computer with a clock speed of 2GHz. The program is applied on four non-manifold 3D folded structures and these structures are adapted from generic packaging buffer and partition designs. The structures are shown with and without hidden lines in Fig. 9 to 12, together with some of their resulting flat layouts. Example 1 (with 10 faces and 5 hyper-common edges) is a tubular structure (partition) with both ends open. Example 2 (with 11 faces and 5 hyper-common edges) is the same structure as that of Example 1 except that there is one extra end face added to cover the front bottom left opening. Example 3 (with 16 faces and 9 hyper-common edges) is an example of an edge buffer that can be used for cushioning the edge of a product. Example 4 (with 16 faces and 10 hyper-common edges) is an example of a corner buffer that can be used for cushioning the corner of a product. Note that in the number labelling of the faces in the folded structure of Example 4, a number in a box (e.g. 2) indicates an interior face, i.e. a face that is on the inside and cannot be seen from the outside of the structure.

The results of the unfolding and flat layout generation procedure are summarized in Table 1. Note that there are no topologically valid resulting spanning trees for the tubular partition in Example 1 (Fig. 9), but with one face added to it as in Example 2, a sizable number of valid trees are realized but only a sampling of four valid non-overlapping flat layouts are shown here (Fig. 10). Four sample layouts are also shown for Example 3 (Fig. 11). Due to the extremely large total number of possible spanning trees for Example 4, only 245,831,029 out of that total number of trees have been evaluated and the values in Table 1 pertain only to this limited number of results evaluated (and 20 sample layouts are shown in Fig. 12).

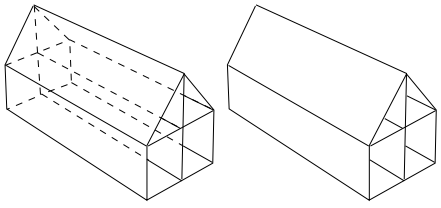

Fig. 9. Tubular partition that has no topologically valid flat layout (Example 1).

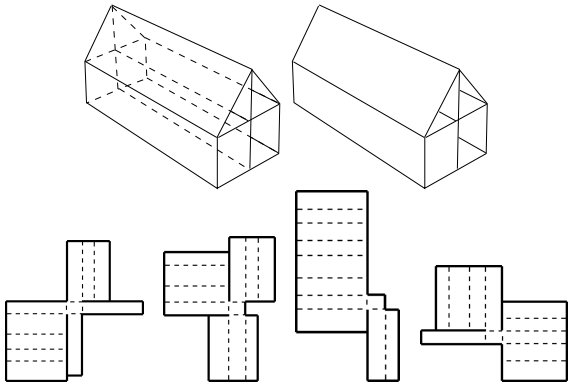

Fig. 10. Tubular partition with an end face, and four sample layouts (Example 2).

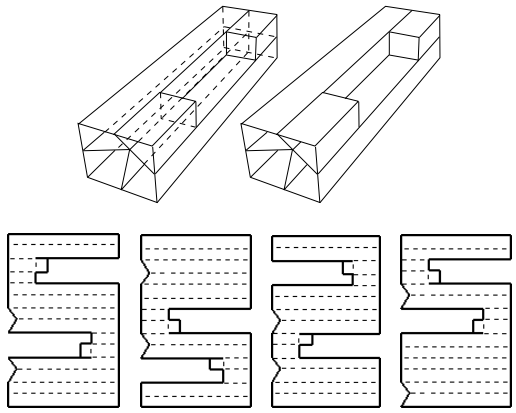

Fig. 11. Edge buffer with four sample layouts (Example 3).

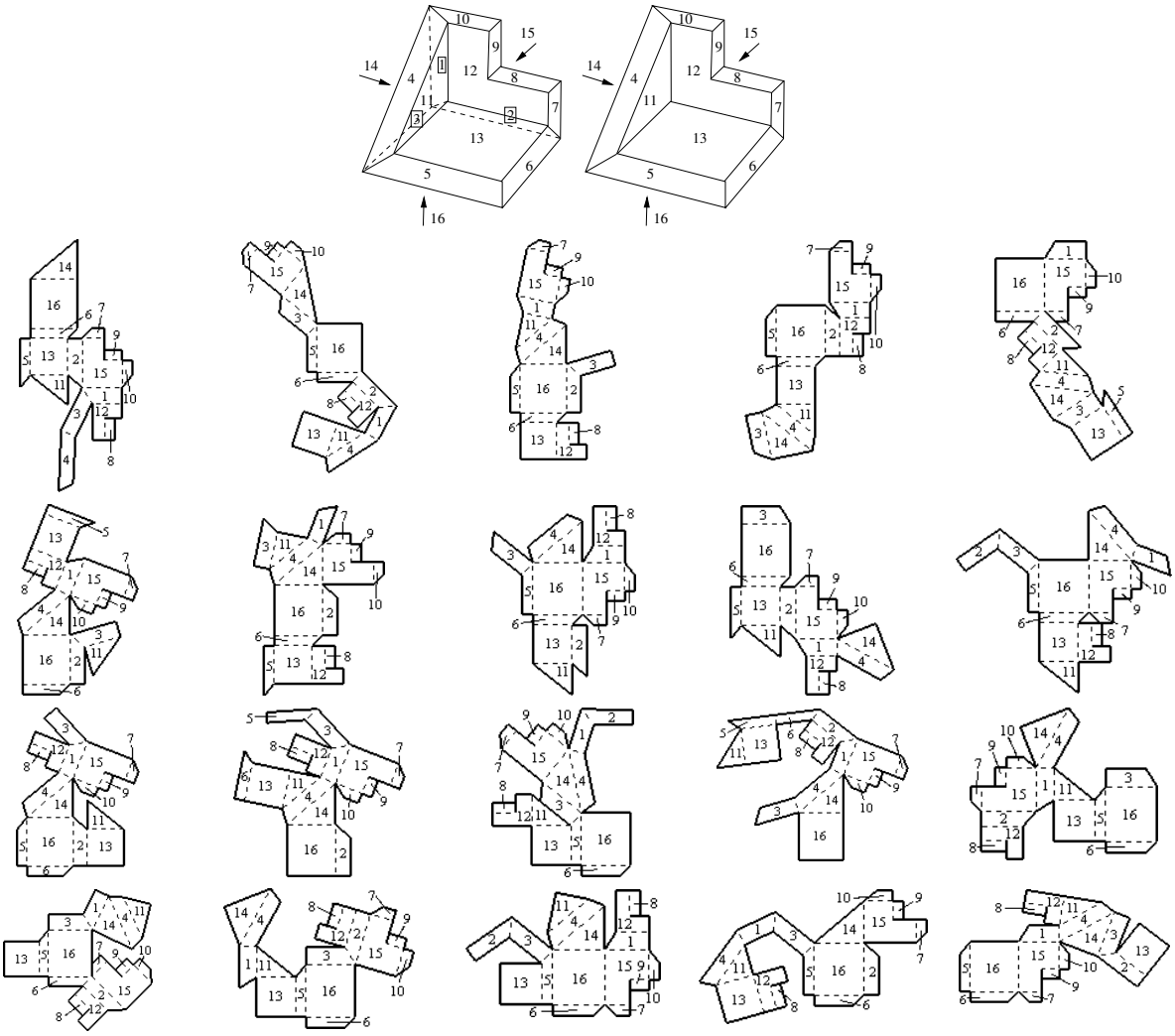

Fig. 12. Corner buffer with 20 sample layouts (Example 4).

Example	No. of Spanning Trees	No. of Topologically Valid Trees	No. of Non-Overlapping Flat Layouts	CPU Time
1	25,365	0	0	4 sec
2	204,644	758	552	53 sec
3	6,981,120	1,836	144	7 min
4	1.9626×10^{10}	-	361,514	42 hr 25 min

Tab. 1. Summary of results for the four examples.

6. CONCLUDING REMARKS AND FUTURE WORK

Problems posed by the unfolding/splitting of faces adjacent to hyper-common edges in non-manifold 3D folded structures have been examined to understand their effects on the graph-theoretic modelling of the topology of the folded structure and the corresponding spanning tree representation of the unfolded flat layout. An overall strategy to generate flat layouts for any given 3D folded structure is then developed based on a procedure to enumerate all possible spanning trees, detect and discard topologically invalid trees (invalid due to hyper-common edge complications), geometrically transform valid trees into their planar layouts, detect/discard those with overlapping faces, and then output the results. The procedure was applied successfully to four example structures featuring numerous hyper-common edges.

Depending on the topology, some structures can have large numbers of possible spanning trees and computational time needed to unfold them may be prohibitive. Instead of enumerating all trees and detecting invalid ones, it may be more efficient to be able to bypass all those topologically invalid trees and exclusively enumerate only those valid ones. Future work will focus on developing systematic strategies to do that.

7. ACKNOWLEDGEMENT

This work was partially supported by the Singapore Ministry of Education Academic Research Fund through research grant RG36/98, which the authors gratefully acknowledge.

8. REFERENCES

[1] Agarwal, P. K., Aronov, B., O'Rourke, J. and Schevon, C. A., Star unfolding of a polytope with application, *SIAM Journal on Computing*, Vol. 26, No. 6, 1997, pp 1689-1713.

[2] Bangay, S., From virtual to physical reality with paper folding, *Computational Geometry*, Vol. 15, 2000, pp 161-174.

[3] Bern, M. and Hayes, B., The complexity of flat origami, in *Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms*, Atlanta, 1996, pp 175-183.

[4] Biedl, T., Demaine, E., Demaine, M., Lubiw, A., Overmars, M., O'Rourke, J., Robbins, S. and Whitesides, S., Unfolding some classes of orthogonal polyhedra, in *Proceedings of the 10th Canadian Conference on Computational Geometry*, Montreal, Canada, 1998.

[5] Dai, J. S. and Rees Jones, J., Mobility in metamorphic mechanisms of foldable/erectable kinds, *ASME Journal of Mechanical Design*, Vol. 121, 1999, pp 375-382.

[6] Gan, W. W. and Pellegrino, S., Closed-loop deployable structures, in *Proceedings of the 44th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, Norfolk, Virginia, U.S.A., 2003, Paper No. AIAA 2003-1450.

[7] Hull, T., On the mathematics of flat origamis, in *Proceedings of the Southeastern International Conference on Combinatorics, Graph Theory and Computing*, Boca Raton, Florida, U.S.A., 1994, pp 215-224.

[8] Jensen, F. and Pellegrino, S., Expandable structures formed by hinged plates, in *Proceedings of the Fifth International Conference on Space Structures*, Guildford, Surrey, U.K., 2002.

[9] Kling, D., Elsayed, E. A. and Basily, B. B., Manufacturing process for folded sheet materials, in *NSF Design, Service and Manufacturing Grantees and Research Conference*, San Juan, Puerto Rico, 2002.

[10] Lang, R. J., A computational algorithm for origami design, in *Proceedings of the 12th Annual ACM Symposium on Computational Geometry*, Philadelphia, 1996, pp 98-105.

[11] Lee, Y. T., Tor, S. B. and Soo, E. L., Mathematical modeling and simulation of pop-up books, *Computers and Graphics*, Vol. 20, No. 1, 1996, pp 21-31.

[12] Lin, Y. L. and Yang, D. C. H., Automatic development generation for thin-walled objects, in

Advances in Design Automation, Vol. 1, ASME, 1994, pp 367-377.

[13] Lin, Y. L. and Yang, D. C. H., Flat pattern generation of thin-walled objects based on a mechanism theory, *Proceedings of the Institution of Mechanical Engineers – Part B – Journal of Engineering Manufacture*, Vol. 212, No. 4, 1998, pp 325-334.

[14] Lipson, H. and Shpitalni, M., On the topology of sheet metal parts, *ASME Journal of Mechanical Design*, Vol. 120, 1998, pp 10-16.

[15] Liu, H. and Dai, J. S., Carton manipulation analysis using configuration transformation, *Proceedings of the Institution of Mechanical Engineers – Part C – Journal of Mechanical Engineering Sciences*, Vol. 216, 2002, pp 543-555.

[16] Liu, W. and Tai, K., Computational geometric modeling and unfolding of 3D folded structures, in *Proceedings of the ASME 2002 Design Engineering Technical Conferences* (*28th Design Automation Conference*), Montreal, Canada, 2002, Paper No. DETC2002/DAC-34046.

[17] Mantyla, M., *An introduction to solid modeling*, Computer Science Press, Rockville, 1988.

[18] Shioura, A., Tamura, A. and Uno, T., An optimal algorithm for scanning all spanning trees of undirected graphs, *SIAM Journal on Computing*, Vol. 26, No. 3, 1997, pp 678-692.

[19] Shpitalni, M. and Lipson, H., 3D conceptual design of sheet metal products by sketching, *Journal of Materials Processing Technology*, Vol. 103, 2000, pp 128-134.

[20] You, Z. and Pellegrino, S., Foldable bar structures, *International Journal of Solids and Structures*, Vol. 34, No. 15, 1997, pp 1825-1847.