To cut or to fill: a global optimization approach to topological simplification

DAN ZENG, Washington University in St. Louis, USA
ERIN CHAMBERS, St. Louis University, USA
DAVID LETSCHER, St. Louis University, USA
TAO JU, Washington University in St. Louis, USA

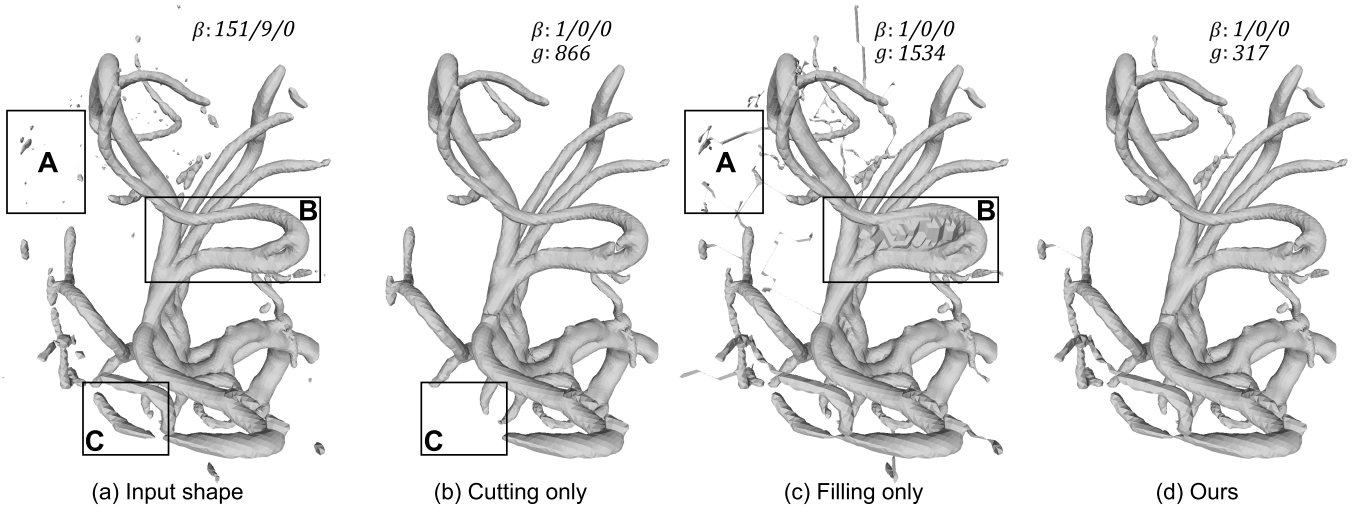


Fig. 1. To simplify the topology of a 3D shape (a), performing *cutting* alone (b) or *filling* alone (c) results in excessive changes, such as removing large components (box C in (b)), creating long bridges to distant islands (box A in (c)) and large patches to fill in a handle (box B in (c)). Given a set of pre-computed cuts and fills, our method optimally selects a subset of them to maximally simplify topology while minimizing the impact on the geometry (d). (β: number of connected components, handles, and cavities; g: geometric cost)

We present a novel algorithm for simplifying the topology of a 3D shape, which is characterized by the number of connected components, handles, and cavities. Existing methods either limit their modifications to be only cutting or only filling, or take a heuristic approach to decide where to cut or fill. We consider the problem of finding a globally optimal set of cuts and fills that achieve the simplest topology while minimizing geometric changes. We show that the problem can be formulated as graph labelling, and we solve it by a transformation to the Node-Weighted Steiner Tree problem. When tested on examples with varying levels of topological complexity, the algorithm shows notable improvement over existing simplification methods in both topological simplicity and geometric distortions.

Authors' addresses: Dan Zeng, Washington University in St. Louis, St. Louis, MO, USA; Erin Chambers, St. Louis University, St. Louis, MO, USA; David Letscher, St. Louis University, St. Louis, MO, USA; Tao Ju, Washington University in St. Louis, St. Louis, MO, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2020 Association for Computing Machinery.
0730-0301/2020/12-ART201 $15.00
https://doi.org/10.1145/3414685.3417854

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Volumetric models**.

Additional Key Words and Phrases: Topology simplification, graph labelling, Steiner tree, cell complexes, global optimization

ACM Reference Format:
Dan Zeng, Erin Chambers, David Letscher, and Tao Ju. 2020. To cut or to fill: a global optimization approach to topological simplification. *ACM Trans. Graph.* 39, 6, Article 201 (December 2020), 18 pages. https://doi.org/10.1145/3414685.3417854

1 INTRODUCTION

Shapes reconstructed from raw data often include many topological features, such as connected components, topological handles, and cavities (i.e., voids inside the shape). While some of these features are intended, many could be artifacts of the reconstruction (e.g., Figure 1 (a)). Excessive amount of topological features can be detrimental to many geometry processing tasks, including parameterization, shape analysis, and physical simulations.

A topological feature can be removed either by *cutting* contents from the shape or *filling* the shape with new contents (see Figure 2). For example, a handle can be removed by cutting open the handle body or filling in the handle hole. A connected component can be

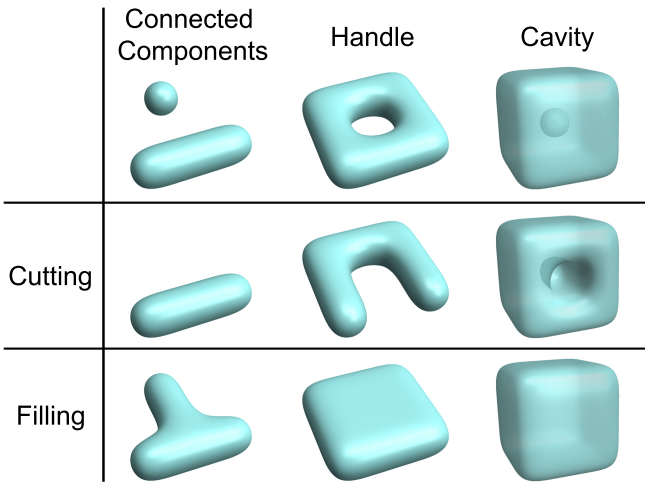


Fig. 2. A topological feature (e.g., a connected component, handle, or cavity) can be removed by either cutting or filling.

removed by either deleting that component (i.e., cutting) or connecting it with another component (i.e., filling). Similarly, a cavity can be removed by either filling the cavity or cutting a tunnel that connects the cavity with the exterior. While different operations may lead to the same topological outcome, they can have very different impact on the *geometry* of the shape. Ideally, a topological simplification algorithm should remove as many topological features as possible while changing the geometry as little as possible.

Many existing methods for topological simplification are *monotonic*, in that they perform only cutting or only filling to the entire shape. These methods can make excessive changes to the shape when there are features that are better to be cut (e.g., the small islands in Figure 1 (a) box A and the thin handle in box B) as well as features that are better to be filled (e.g., the big component in box C). Few methods are *non-monotonic*, in that they cut some features and fill others. In these methods, however, the decision of where to cut or fill is made heuristically to *locally* minimize the change to the geometry. Such heuristics are prone to make globally sub-optimal choices, particularly when the decision of cutting or filling one topological feature affects how other features can be removed (see more discussion in the next section).

In this paper, we attempt to find the *globally optimal* set of cuts and fills that maximally simplifies the topology while minimizing geometric changes. Leveraging existing *monotonic* simplification methods, we take as input a pre-computed set of cuts and fills, each associated with some geometric cost (e.g., total volume of the cut or fill). The problem then becomes selecting a subset of these cuts and fills with the minimal total geometric cost, such that applying them to the shape maximally reduces the number of connected components, handles, and cavities.

To solve the optimal selection problem, we make two technical contributions. First, we re-formulated the problem as graph labelling, where the given cuts and fills are nodes in the graph and a binary label indicates whether a cut or fill is selected (Section 5). While the graph formulation makes the problem more computation-friendly, finding the optimal labelling is still challenging, since the labelling energy involves the number of connected components in the labelled sub-graphs. Our second contribution is an algorithm for solving this labelling problem by transforming it to the well-known Node-Weighted Steiner Tree (NWST) problem (Section 6). The transformation allows us to leverage state-of-the-art solvers of NWST (e.g., [Leitner et al. 2018]) to find a near-optimal labelling.

We demonstrated our algorithm on cuts and fills produced by two popular monotonic simplification methods, morphological opening and closing [Nooruddin and Turk 2003], and topology-controlled inflation and deflation [Bischoff and Kobbelt 2002; Kriegeskorte and Goebel 2001; Szymczak and Vanderhyde 2003]. When tested on a suite of shapes with a wide range of topological complexity, our method consistently achieves lower geometric cost than existing simplification methods while being equally, if not more, effective in reducing topological complexity (e.g., Figure 1 (d), and more in Section 7).

2 RELATED WORK

We briefly review existing methods for simplifying the topology of 3D shapes, with an emphasis on how the decisions of where to cut or fill are made. In-depth discussions of many of these methods can be found in the survey [Attene et al. 2013]. We conclude with a brief discussion on topology-aware surface reconstruction methods.

Monotonic methods. These methods perform only cutting or only filling to the entire shape. A common monotonic approach to removing topological handles is to first represent the shape (resp. its complement) by a weighted connectivity graph, where the edge weight encodes the geometric cost of cutting (resp. filling), then identify edges to be removed from the graph using a minimum-weighted spanning tree (MST), and finally perform the corresponding cuts (resp. fills) to the shape. The graph can be created in various ways, such as an axis-aligned Reeb graph [Chen and Wagenknecht 2006; Shattuck and Leahy 2001], an adjacency graph of segmented parts [Han et al. 2002], and a curve skeleton [Zhou et al. 2007]. While this approach solves the handle-cutting or handle-filling problem optimally (due to optimality of MST), extending it to also make globally optimal choice between cutting and filling for each handle seems challenging. A remedy is to alternate between running the method on the shape and on the complement, each time cutting all handle bodies or filling all handle holes smaller than some user-specified parameter. However, the result can be sensitive to the choice of parameters (see Figure 10), and there is no guarantee of optimality. Furthermore, these methods cannot reduce the number of connected components or cavities.

To simplify all types of topological features (components, handles, cavities), one may apply morphological opening or closing to a shape represented by voxels [Nooruddin and Turk 2003]. Parameterized by a *structure element*, opening (resp. closing) cuts away (resp. fills in) small topological features where the structure element cannot fit inside (resp. outside) the shape. However, both operators may introduce new topological features, and the user has no direct control over the final topology. Several authors [Bischoff and Kobbelt 2002; Kriegeskorte and Goebel 2001; Szymczak and Vanderhyde 2003]

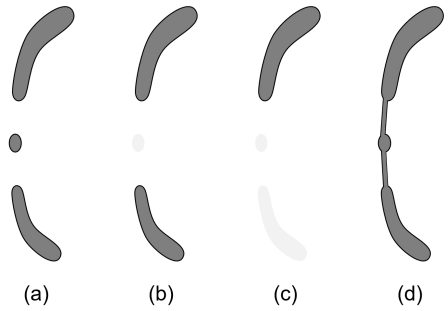


Fig. 3. A scenario where a greedy heuristic for choosing cutting or filling produces suboptimal results. See text for details.

achieved controlled simplification by deflating (resp. inflating) an initial *seed* towards the shape while forbidding, or conditionally allowing, topological changes. A different approach was presented recently [Chambers et al. 2018], which uses a heuristic to explore candidate fills with the goal of maximally simplifying topology. All of these methods work on voxelized shapes. However, since these methods cannot selectively apply cuts and fills to different parts of the shape, they may lead to excessive shape modifications (e.g., Figure 1 (b,c)).

Non-monotonic methods. These methods perform both cutting and filling on the shape. The majority of non-monotonic methods can only remove topological handles. They attempt to minimize different metrics of geometric changes, such as the total lengths of loops on the surface (*non-separating paths*) [Wood et al. 2004], the total volume of voxels to be cut or filled [Kriegeskorte and Goebel 2001], or the volume weighted by image intensity [Ségonne et al. 2007]. To the best of our knowledge, the MendIT method of [Ju et al. 2007] is the only non-monotonic method that can remove all three types of topological features. This method represents the topology of a voxelized shape and its complement by two graphs, which are updated as the shape is being modified, and uses information from both graphs to decide where to cut or fill next in order to minimize the total volume of change.

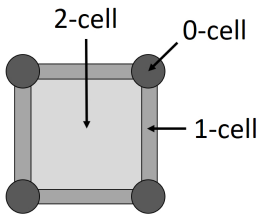
All these non-monotonic methods apply a greedy heuristic to decide which feature to remove next and whether it should be cut or filled. The heuristic favors the topological modification that results in the least geometric changes *at the current stage of the algorithm*. This greedy strategy often fails to make globally optimal decisions. A simple example is illustrated in Figure 3. To reduce the number of connected components of the shape in (a), a greedy heuristic would first remove the small island in the middle, as shown in (b), which results in a smaller change to the shape than connecting the island with the other two larger components. However, this move creates a large gap between the bottom and top components, which forces the heuristic to remove the bottom component in the next step, as shown in (c). A solution with a lower overall change to the shape would instead keep the middle island as a "link" to bridge the other two components, as shown in (d). We demonstrate many examples in Section 7 where the greedy heuristic becomes sub-optimal.

Topology-controlled surface reconstruction. A number of surface reconstruction methods allow the user to control the topology of the reconstructed surface and hence avoid the need for post-processing simplification. Some methods are guided by user interaction [Sharf et al. 2007; Yin et al. 2014], some allow the user to prescribe the number of handles [Huang et al. 2017; Lazar et al. 2018; Zou et al. 2015], and others take a deformable modeling approach and evolve a template while restricting topological changes [Han et al. 2003; Ségonne 2008; Sharf et al. 2006; Zeng et al. 2008]. However, since the majority of reconstruction methods do not offer direct topology control, topological simplification remains necessary.

3 BACKGROUND

Our algorithm is designed on shapes represented as cell complexes. We first briefly review their definitions and properties. In-depth discussions can be found on standard textbooks on algebraic topology such as [Hatcher 2002].

Intuitively, a cell complex represents a decomposition of space into topologically simple units, called *cells*. A k-dimensional cell, or *k-cell*, is an open set homeomorphic to an open k-dimensional ball. We call a cell x a *face* of cell y if x is contained in the boundary of y. The insert shows a quadrilateral 2-cell and its faces (four 1-cells and four 0-cells). A finite set of disjoint cells (like the one in the insert) is called a *cell complex* if, for each cell in the complex, all its faces are also in the complex. Two sets of cells (which may not form cell complexes) are *connected* if some cell in one set is the face of a cell in the other set.



The n-th *Betti number* β_n of a cell complex is the rank of the n-th homology group. Intuitively, for a 3-dimensional cell complex X, $\beta_0(X)$ is the number of connected components of X, $\beta_1(X)$ is the number of topological handles of X, and $\beta_2(X)$ is the number of cavities in X (i.e., number of connected components of the complement space $\overline{X}$ minus 1). The alternating sum of Betti numbers defines the *Euler characteristic* χ:

$$\chi(X) = \beta_0(X) - \beta_1(X) + \beta_2(X) \tag{1}$$

Alternatively, the Euler characteristic can be found by the alternating sum of the number of cells in X at different dimensions:

$$\chi(X) = k_0(X) - k_1(X) + k_2(X) - k_3(X) \tag{2}$$

where $k_d(X)$ is the number of d-dimensional cells in X.

4 PROBLEM STATEMENT

Topological simplification aims at minimally modifying a shape to remove as many topological features (connected components, handles, cavities) as possible. However, since the number of possible modifications to the shape is virtually infinite, the problem can be untractable. Leveraging existing topological simplification methods, we consider a limited space of shape modifications in the form of a pre-defined set of *cuts* (contents to be removed from the shape) and *fills* (contents to be added to the shape). These cuts or fills can

be obtained by running a monotonic simplification method in a "cut-only" mode or "fill-only" mode (as reviewed in Section 2). In addition, we assume that each cut or fill is associated with some geometric cost. Given these inputs, the topological simplification problem reduces to *selecting* a subset of the cuts and fills with the least total geometric cost such that the resulting shape has the simplest topology.

Formally, we represent the space by a 3D cell complex $\mathbf{\Omega}$ (e.g., a hexahedral or tetrahedral mesh). Let $\mathbf{O}$ be the set of all 3-cells (e.g., hexahedra or tetrahedra). As input, we are given a set of *shape cells* $T \subset \mathbf{O}$, *cut cells* $C \subset T$ and *fill cells* $F \subset \mathbf{O} \setminus T$, and a geometric cost $g(v)$ for each cut or fill cell v. For topological analysis, we define the shape as the closure $\Omega(T)$, which is a cell complex made up of 3-cells in T and their (lower-dimensional) faces.

We seek a binary labelling of the cut cells and fill cells, indicating whether they belong to the shape (label 1) or not (label 0), such that the modified shape has the least total number of topology features and, secondarily, the sum of geometric costs over all cut cells removed from the shape and all fill cells added to the shape is minimal. Given a labelling L, we denote the set of cut cells C (resp. fill cells F) that are labelled $\delta = 1, 0$ under L as $C_{L,\delta}$ (resp. $F_{L,\delta}$). The cut cells selected to be removed from the shape, and the fill cells selected to be added to the shape, are therefore $C_{L,0}$ and $F_{L,1}$, respectively. The modified shape is the closure of all 3-cells that remain in the shape, $X = \Omega((T \setminus C_{L,0}) \cup F_{L,1})$. Our two objectives can be expressed by minimizing the following vector *lexicographically*,

$$\left\{ \beta_0(X) + \beta_1(X) + \beta_2(X), \sum_{v \in C_{L,0} \cup F_{L,1}} g(v) \right\} \tag{3}$$

where β_n is the n-th Betti number. Since the first term of Equation 3 is an integer, we can equivalently minimize the following scalar value,

$$\lambda * (\beta_0(X) + \beta_1(X) + \beta_2(X)) + \sum_{v \in C_{L,0} \cup F_{L,1}} g(v), \tag{4}$$

where λ is any constant greater than $\sum_{v \in C \cup F} g(v)$. To further simplify the computation of this energy, we replace $\beta_1(X)$ (the number of handles in X) by the Euler characteristic $\chi(X)$, which can be computed by counting cells in X (see Equation 2). Substituting Equation 1 into the equation above, we arrive at the following scalar energy of the labelling L,

$$E(L) = 2\lambda * (\beta_0(X) + \beta_2(X)) - \lambda * \chi(X) + \sum_{v \in C_{L,0} \cup F_{L,1}} g(v), \tag{5}$$

We will discuss in Section 7 how we obtain the cut cells C, fill cells F, and geometric cost g from existing monotonic topological simplification algorithms [Bischoff and Kobbelt 2002; Chambers et al. 2018; Kriegeskorte and Goebel 2001; Nooruddin and Turk 2003; Szymczak and Vanderhyde 2003]. Since these algorithms all operate on voxelized shapes, our implementation is specialized to cell complexes $\mathbf{\Omega}$ made up of hexahedral cells $\mathbf{O}$. However, our algorithm, as described in the next two sections, applies to any type of 3D cell complex.

5 GRAPH FORMULATION

To solve the labelling problem formulated above, we will first reformulate it as a graph labelling problem. The graph formulation allows us to utilize existing graph optimization techniques, as we shall see in the next section. Our graph formulation transforms the energy in Equation 5, which is expressed in terms of the topology and geometry of the shape, into a labelling energy on the graph. In particular, the number of connected components and cavities of the modified shape ($\beta_0(X)$, $\beta_2(X)$) are captured by the number of connected components in the 1-labelled and 0-labelled subgraphs, respectively. The Euler characteristic ($\chi(X)$) and the geometric cost ($g(v)$) are transformed into the sum of per-node labelling costs.

5.1 Graph construction

To faithfully capture the topology of the shape, our graph represents not only the cut and fill cells, but also the remaining 3-cells in the space. The latter consists of two types of 3-cells, *kernel cells* $T \setminus C$ and *neighborhood cells* $\mathbf{O} \setminus (T \cup F)$. Regardless of the labelling of the cut or fill cells, kernel cells are always in the shape whereas neighborhood cells are always outside the shape.

The graph is denoted by $G = \{V, E\}$, where V, E are the sets of nodes and edges. Each node $s \in V$ represents a group of 3-cells of the same type (i.e., kernel, cut, fill, or neighborhood), denoted by O_s. We call s a *kernel*, *cut*, *fill*, or *neighborhood node* if O_s consists of the respective type of 3-cells. There are a number of reasons for defining O_s as a group of cells instead of a single cell. First, it usually takes a collection of cut or fill cells to achieve the desired topological change (e.g., cutting open a handle body or filling in the handle hole). Second, defining the graph at the level of cell groups makes the problem more efficient to solve than labelling individual cells. More precisely, O_s is a connected component of 3-cells of the same type, by some notion of connectivity that will be defined below. Similarly, two nodes $s, t \in V$ are connected by an edge in E if their cells O_s, O_t are connected. A 2D illustration of graph construction is shown in Figure 4 (a,b).

The key to the definition of nodes and edges is the notion of *connectivity* between two 3-cells. Our connectivity rule is chosen so that the topology of the shape can be captured by graph labelling. More precisely, consider a binary labelling L of the nodes V, so that setting $L(s) = \delta$ for $\delta = 0, 1$ and node $s \in V$ assigns label δ to all 3-cells in O_s. Note that $L(s) = 1$ (resp. $L(s) = 0$) for any kernel (resp. neighborhood) node s. We define the *δ-labelled subgraph* of G for $\delta = 1, 0$, denoted as $G_{L,\delta}$, as the set of all i-labelled nodes and all edges connecting two i-labelled nodes. In the 2D example of Figure 4 (c), the 1-labelled and 0-labelled subgraphs are colored magenta and cyan, respectively. Ideally, for any labelling L, *each connected component of $G_{L,1}$ should correspond to a connected component of the shape, and each connected component of $G_{L,0}$ should correspond to either a cavity of the shape or the infinite background.*

Straight-forward connectivity rules often fail to achieve this goal. We illustrate with a 2D quadrilateral cell complex in Figure 5. Consider the input in (a), and specifically the three fill cells u, v, w. Suppose we adopt the connectivity rule that two 2-cells are connected if they share any common face (e.g., a 1-cell or a 0-cell). Then u, v, w belong to a single connected component. This component is

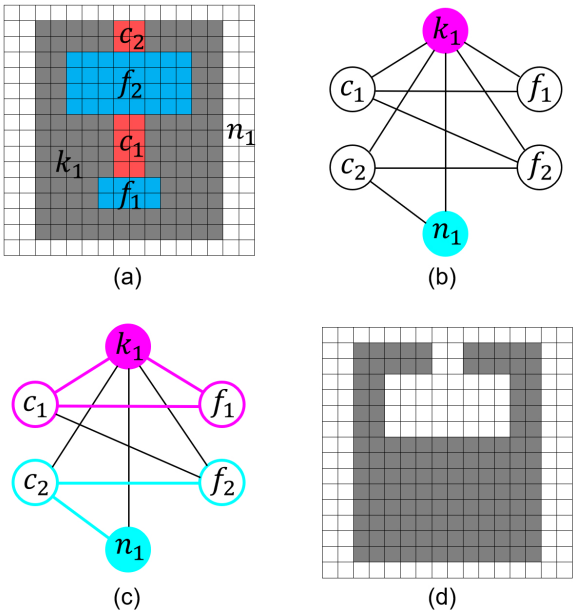


Fig. 4. Graph construction and labelling on a quadrilateral cell complex in 2D. (a): Input cell complex with kernel cells (gray), cut cells (red), fill cells (blue), and neighborhood cells (white). (b): The graph with kernel node k_1, cut nodes c_1, c_2, fill nodes f_1, f_2, and neighborhood node n_1 (the cells represented by each node are marked in (a)). The kernel and neighborhood nodes are colored magenta and cyan, respectively, to indicate that they are always labelled 1 and 0. (c): A binary labelling of the cut nodes and fill nodes as 1 (magenta) and 0 (cyan). Edges in each labelled subgraph are colored accordingly. (d): The modified shape defined by the labelling in (c).

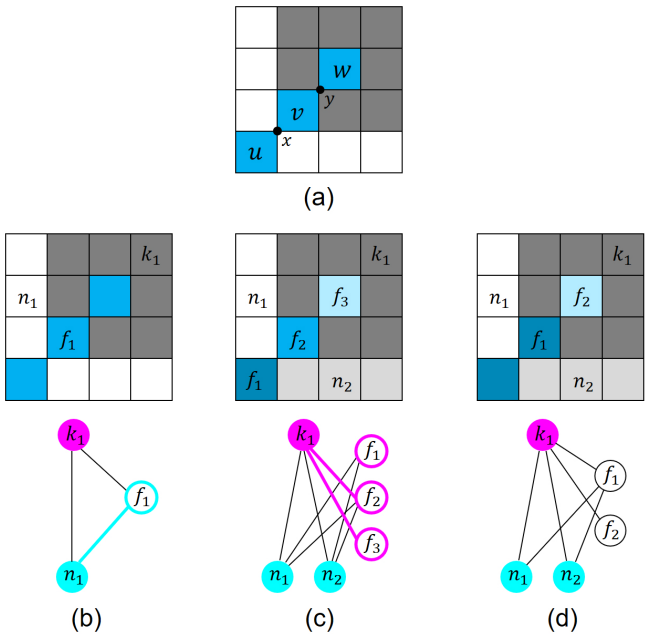


Fig. 5. Comparing graphs constructed from the same input (a) by assuming that two 2-cells are connected if they share any common face (b), if they share a common 1-dimensional face (c), and if they are R-connected (d). See text for details.

represented by one fill node, noted as f_1, which is connected to both the kernel node k_1 and neighborhood node n_1, as shown in (b). If f_1 is labelled 0, the shape (consisting of only kernel cells) would have one cavity (left behind by w) and one background. However, the 0-labelled subgraph (colored cyan in the graph of (b)) has only one connected component. On the other hand, suppose we adopt the connectivity rule that two 2-cells are connected only if they share a common 1-cell. Then u, v, w are disconnected from each other. They are each represented by a fill node, as shown in the graph in (c). If these fill nodes are all labelled 1, the shape (consisting of all kernel and fill cells) would have one connected component. However, the 1-labelled subgraph (colored magenta in the graph of (c)) consists of two connected components, since the fill node f_1 (representing cell u) is not connected to other fill nodes or the kernel node k_1.

We propose a connectivity rule that allows our graph to correctly capture the topology of the shape, at the cost of slightly restricting the space of labellings. The new connectivity rule considers not only the spatial configuration of two 3-cells, but also their types and types of their surrounding cells. We *rank* the four types of 3-cells in the order of kernel, cut, fill, and neighborhood, so that the kernel type has the highest rank and neighborhood type has the lowest rank. We define a rank-based connectivity (or *R-connectivity*) as:

Definition 5.1. Two 3-cells u, v are *R-connected* if they share a common face that is not the face of another 3-cell whose rank is higher than both u, v.

We again use the 2D example of Figure 5 (a) to illustrate R-connectivity. Note that the two fill cells u, v are R-connected, because they share a common face (x) that is a face of only neighborhood cells (colored white), which are at a lower rank than fill cells. On the other hand, fill cells v, w are not R-connected, because their only common face (y) is also the face of some kernel cells (colored gray), which are at a higher rank than fill cells. As a result, the graph, shown in Figure 5 (d), consists of two fill nodes, one representing u, v (noted as f_1) and the other representing w (noted as f_2). One can verify that, for any labelling of f_1, f_2, the 1-labelled subgraph has the same number of connected components as the shape, while the number of connected components in the 0-labelled subgraph is the number of cavities of the shape plus 1.

The following statement formalizes how our graph, based on R-connectivity, captures the topology of the shape:

Proposition 5.2. *A labelling L is called* proper *if it satisfies the following constraint: if a cut node s and a fill node t are connected by an edge, than either $L(s) = 1$ or $L(t) = 0$. Denote the modified shape as $X = \Omega(\cup_{L(s)=1} O_s)$. For any proper labelling L,*

(1) $\beta_0(X) = \beta_0(G_{L,1})$.
(2) $\beta_2(X) = \beta_0(G_{L,0}) - 1$.

where $\beta_0(S)$ counts the number of connected components in a graph S.

The proof is given in the Appendix A. The properness constraint essentially forbids a connected pair of cut and fill to be applied to the shape at the same time. We argue that this constraint does not impose a significant restriction on the space of labelling that we are interested in. Note that such a pair usually acts on the same topological feature (e.g., c_1, f_1 or c_2, f_2 in Figure 4). Since using both the cut and the fill would likely lead to increased geometric cost without any further reduction in topology, it is reasonable to exclude such labelling in our solution space.

5.2 A graph labelling problem

We shall re-formulate the problem posed in Section 4 to a labelling problem on the graph constructed above. The key is to express the energy E of Equation 5 as an energy on the labelled graph. The previous section showed that the first part of E is captured by the connected components in the 0- and 1-labelled subgraphs for any proper labelling (Proposition 5.2). In this section, we show that the remaining part of E can be written as the sum of labelling costs defined at each graph node.

We first re-write the Euler characteristic so that it can be expressed as the sum of node-wise quantities. We do so by decomposing the modified shape $X = \Omega(\cup_{L(s)=1} O_s)$ into groups of cells, each associated with a graph node. Specifically, we define Ω_s as the subset of cells in the closure $\Omega(O_s)$ that are not faces of any 3-cell ranking higher than O_s. The set Ω_s consists of all 3-cells in O_s as well as some of their lower-dimensional faces. As we show in Appendix A (Lemma A.1 (1) and equality 11), for any proper labelling, Ω_s of all 1-labelled nodes $s \in V$ form a decomposition of the shape X. As a result, $\chi(X)$ is simply the sum of $\chi(\Omega_s)$ over all 1-labelled nodes s. This equality can be further transformed into

$$\chi(X) = \sum_{s \in V_C \cup V_K} \chi(\Omega_s) - \sum_{\substack{s \in V_C \\ L(s)=0}} \chi(\Omega_s) + \sum_{\substack{s \in V_F \\ L(s)=1}} \chi(\Omega_s)$$

where V_K, V_C, V_F denote the sets of kernel nodes, cut nodes and fill nodes. Note that the first term is constant regardless of the labelling.

We can now express the minimization of the last two terms of E in Equation 5 on the graph as minimizing the sum of a node-wise labelling cost $h(s, L(s))$ over all nodes $s \in V$, as defined below:

$$h(s, \delta) = \begin{cases} \lambda * \chi(\Omega_s) + \sum_{v \in O_s} g(v), & \text{if } s \in V_C \text{ and } \delta = 0 \\ -\lambda * \chi(\Omega_s) + \sum_{v \in O_s} g(v), & \text{if } s \in V_F \text{ and } \delta = 1 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

Intuitively, h captures the *change* in the Euler characteristic and the geometric cost with respect to the input shape. As a result, labelling a cut node as 1 or a fill node as 0 does not incur any cost. Also, observe that h can be either positive or negative.

Finally, we reformulate the problem posed in Section 4 as a graph labelling problem, which we call *topological labelling* (or *TL*):

Definition 5.3 (Topological Labelling). Find a proper labelling L of graph G that minimizes:

$$E_G(L) = 2\lambda * (\beta_0(G_{L,0}) + \beta_0(G_{L,1})) + \sum_{s \in V_C \cup V_F} h(s, L(s)). \tag{7}$$

We note that the solution space of TL is smaller than that of the problem posed in the previous section, since a labelling of the graph only adds or subtracts a group of (R-connected) cells at a time. However, the grouping of cells in TL yields a smaller problem to solve. Moreover, as we shall see in the next section, TL lends itself well to existing graph optimization tools.

6 OPTIMIZATION

Topology labelling (TL) is a challenging graph optimization problem, as the energy E_G involves the number of connected components in the labelled subgraphs. To the best of our knowledge, no existing graph optimization tool can directly minimize such an energy. For example, the popular graph-cut framework [Boykov et al. 2001] minimizes the sum of node-wise and edge-wise labelling costs, which are inadequate to capture graph connectivity.

We start with a greedy strategy to minimize the energy (Section 6.1). While simple to implement, the strategy can easily produce sub-optimal results. Next, we introduce a variant of TL, which we call As-Connected-As-Possible (ACAP) TL, and show that it can be transformed to the Node-Weighted Steiner Tree (NWST) problem (Section 6.2). We then present our complete solution that combines the NWST solver with the greedy strategy (Section 6.3). The section concludes with ways to improve scalability of the algorithm on large graphs (Section 6.4).

6.1 Greedy strategy

We first consider a greedy, iterative heuristic to solve TL. Note that a similar strategy is used in all existing non-monotonic topology simplification methods [Ju et al. 2007; Kriegeskorte and Goebel 2001; Ségonne et al. 2007; Wood et al. 2004]. We initialize the labels by labelling all cut nodes 1 and all fill nodes 0 (i.e., the input shape). At each iteration, we evaluate the "benefit" of each node as the decrease in the energy E_G if the label of the node is flipped, and flip the label of the node with greatest benefit that does not violate the properness constraint in Proposition 5.2. The process is repeated until no node has a positive benefit.

As illustrated in Figure 6, this straight-forward strategy can easily produce sub-optimal results. In the initial labelling (a), flipping the label of the cut node c_1 or either of the two fill nodes f_1, f_2 would reduce the number of connected components in the shape by 1. Assuming that cut cells O_{c_1} have a smaller geometric cost than those of the fill cells O_{f_1} or O_{f_2}, c_1 would have a higher benefit than f_1 and f_2, and therefore it will be labelled 0 in the next step, as shown in (b). Doing so, however, prevents the algorithm from proceeding any further, because the properness constraint forbids either f_1 or f_2 to be labelled 1. This produces a suboptimal result that contains two connected components. The optimal labelling for this input would label all three nodes c_1, f_1, f_2 as 1, which results in a single connected component as shown in (c).

6.2 As-Connected-As-Possible Topology Labelling

To improve upon the greedy heuristic, we hope to transform TL into some existing graph optimization problem for which mature, global optimizers have been developed. As our energy concerns graph connectivity, the Steiner Tree (ST) problems naturally came

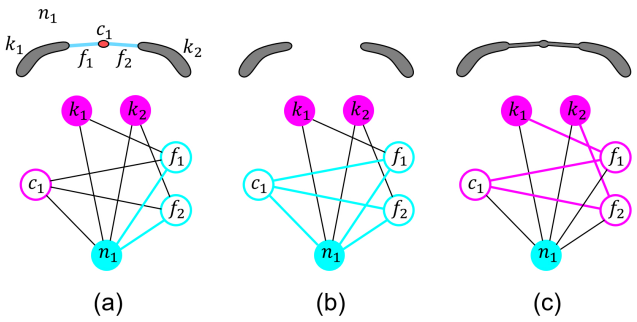


Fig. 6. (a): An input shape (top) and the graph with two fill nodes and one cut node (bottom) shown with the initial labelling. (b): The greedy strategy labels the cut node c_1 as 0, producing a sub-optimal result with two connected components. (c): The optimal labelling assigns 1 to all three nodes c_1, f_1, f_2 so that the shape consists of a single connected component.

to our minds. While many flavors of ST exist, they generally look for a subgraph in a given graph that connect up a given set of "terminal" nodes while minimizing the sum of node-wise or edge-wise costs over the subgraph.

The key difference between our problem, TL, and ST is that the former maximizes graph connectivity as part of its energy, whereas the latter imposes connectivity as a constraint. To leverage solvers of ST, we will consider a variant of TL that treats graph connectivity as a hard constraint instead of a soft energy term. As we will show, this TL variant can be reduced to the Node-Weighted Steiner Tree (NWST) problem.

6.2.1 A variant of TL. Our idea is to impose the constraint on a labelling L that the two labelled subgraphs, $G_{L,0}$ and $G_{L,1}$, are as connected as they can possibly be. Consider the labelling L_1 (resp. L_0) that labels all cut and fill nodes as 1 (resp. 0). Since all kernel nodes are labelled as 1, the least number of connected components in the 1-labelled subgraph $G_{L,1}$ for any labelling L is the number of those connected components in $G_{L_1,1}$ that contain some kernel nodes. Symmetrically, since all neighborhood nodes are labelled as 0, the least number of connected components in the 0-labelled subgraph $G_{L,0}$ for any L is the number of those connected components in $G_{L_0,0}$ that contain some neighborhood nodes. We call two kernel (resp. neighborhood) nodes *reachable* if they lie in the same connected component of $G_{L_1,1}$ (resp. $G_{L_0,0}$). A maximum set of mutually reachable neighborhood (resp. kernel) nodes is called a *reachable set*. We call a labelling L *As-Connected-As-Possible (ACAP)* if each connected component of $G_{L,1}$ spans a reachable set of kernel nodes, and each connected component of $G_{L,0}$ spans a reachable set of neighborhood nodes.

We formulate the following variant, called ACAP-TL, that removes the connected components from the energy of TL and imposes the ACAP constraint instead:

Definition 6.1 (As-Connected-As-Possible Topological Labelling). Find a proper and ACAP labelling L of graph G that minimizes:

$$E^*_G(L) = \sum_{s \in V_C \cup V_F} h(s, L(s)). \tag{8}$$

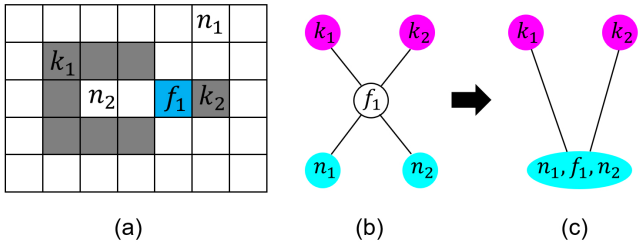


Fig. 7. (a): Adding the fill cell (blue) to the shape would reduce one connected component but create one cavity. (b): The corresponding fill node f_1 in the graph is a double articulation node. (c): f_1 is resolved by labelling it as 0 and merging with connected neighborhood nodes.

Observe that the energy of ACAP-TL, E^*_G, is a simple sum of node-wise labelling costs, which makes the problem closer to an ST problem. We note that ACAP-TL is not the same as TL, in two important ways. First, the solution to ACAP-TL may not be the solution of TL. This is because a labelling that minimizes the TL energy may not have to be ACAP. For example, the solution of TL may produce more connected components than the solution of ACAP-TL but with many fewer handles. Second, unlike TL, a solution of ACAP-TL may not exist, if no labelling can satisfy the ACAP constraint. A typical example is when a cut or fill node is an articulation node in both $G_{L_1,1}$ and $G_{L_0,0}$. That is, labelling the node as either 1 or 0 would disconnect a reachable set of neighborhood or kernel nodes. We call such nodes *double articulation nodes*. This situation is illustrated in Figure 7 (a,b), where the fill node f_1 is the articulation node. As we shall discuss in Section 6.3, we resolve these two differences by combining greedy heuristics, like the one introduced in the previous section, with solving ACAP-TL.

6.2.2 Reduction to NWST. We can reduce the ACAP-TL problem to the *Node-Weighted Steiner Tree (NWST)* problem. Consider a graph $H = \{U, A, R, \omega\}$ with nodes U, edges A, terminal nodes $R \subseteq U$, and weights $\omega : U \setminus R \rightarrow \mathbb{R}$ on the non-terminal nodes. The NWST problem seeks a connected subgraph of H that spans R and minimizes the sum of weights over non-terminal nodes in the subgraph.

At a first glance, ACAP-TL differs from NWST in several ways. First, while NWST only asks the subgraph to be connected, an ACAP labelling requires *both* the 1-labelled and 0-labelled subgraphs to be connected. Second, NWST requires *all* terminals to be connected, whereas only nodes (whether kernel or neighborhood) within each reachable set need to be connected in ACAP-TL. Third, ACAP-TL has an additional constraint - the labelling has to be proper.

We shall construct a new graph H from G, such that a solution of NWST on H corresponds to a solution of ACAP-TL on G. As we explain below, the graph H is designed to address the differences between ACAP-TL and NWST mentioned above. The construction is also illustrated in Figure 8.

First, we create one terminal node in H for each kernel node and neighborhood node in G. These two sets of terminals are denoted as U_K, U_N respectively. In addition, for each group of terminals in U_K (resp. U_N) representing a reachable set of kernel (resp. neighborhood) nodes, we select an arbitrary terminal in that group and connect it with an auxiliary terminal node π. This ensures that all

terminals are connected by a subgraph of H as long as each group of terminals representing a reachable set is connected.

Next, we create two types of non-terminal nodes in H, denoted by U_C, U_F. Each node p in U_C or U_F represents a group of cut and fill nodes in G, denoted by $V(p)$, whose definition will be given shortly. Importantly, each cut or fill node will be represented by at least one node in U_C and one node in U_F. Intuitively, selecting a node p in U_F (resp. U_C) corresponds to labelling all cut and fill nodes in $V(p)$ as 1 (resp. 0) in G. This allows a subgraph S of H to encode a labelling of G, as long as each cut or fill node in G is represented by exactly one non-terminal node in S. We connect U_F, U_C and the terminal nodes U_K, U_N following the connectivity in G. Specifically, each node p of U_F (resp. U_C) is connected with a terminal q of U_K (resp. U_N) if any cut or fill node in $V(p)$ is connected with the kernel (resp. neighborhood) node represented by q.

While not every subgraph of H encodes a labelling of G, we certainly hope that the NWST does. To do so, we first add a new set of terminal nodes, denoted as U_{CF}, one for each cut or fill node of G. A terminal representing a cut or fill node s is connected to all nodes of U_F and U_C that also represent s. This ensures that, in a subgraph of H that connects all terminal nodes (e.g., the NWST), each cut or fill node of G must be represented by *at least* one non-terminal node. Next, we assign each node of U_C, U_F a large weight to penalize a cut or fill node being represented by more than one non-terminal nodes. Specifically, the weight ω at each node $p \in U_F$ (resp. $\in U_C$) is the cost of labelling all nodes of $V(p)$ as 1 (resp. 0) plus a large constant:

$$\omega(p) = \sum_{s \in V(p)} (h(s, \delta_p) + D) \tag{9}$$

where h is the labelling cost defined earlier (Equation 6), $\delta_p = 1$ for $p \in U_F$ and 0 for $p \in U_C$, and D satisfies:

$$D > \sum_{s \in V_C \cup V_F} ||h(s, 0) - h(s, 1)|| - \min_{s \in V_C \cup V_F, \delta = 0, 1} h(s, \delta). \tag{10}$$

As we shall prove later, this lower bound of D gaurantees that the NWST of H contains exactly one non-terminal node representing each cut or fill node of G (unless the solution of ACAP-TL does not exist).

To further ensure that the NWST of H encodes a *proper* labelling of G, we define the nodes of U_C, U_F as follows. We call a set of cut and fill nodes of G *fillable* (resp. *cuttable*) if the set is connected and, for any fill (resp. cut) node in the set, all of its connected cut (resp. fill) nodes are included in the set as well. Observe that, given a collection of mutually disjoint cuttable and fillable sets whose union covers all cut and fill nodes, the labelling that assigns 1 to the fillable sets and 0 to the cuttable sets is always proper. Based on this observation, we create one node in U_F (resp. U_C) representing each fillable (resp. cuttable) set of G.

We now formalize the equivalence between ACAP-TL on G and NWST on H. Given a subgraph S of H, we denote its non-terminal node set as U_S. We say that S is *CF-Disjoint* if $V(p) \cap V(q) = \emptyset$ for any pair of nodes $p \in U_C \cap U_S$ and $q \in U_F \cap U_S$. That is, U_S cannot contain a node from U_C and a node from U_F that represent the same cut or fill node of G. We show that:

Proposition 6.2. *Let S be a solution of NWST on H,*

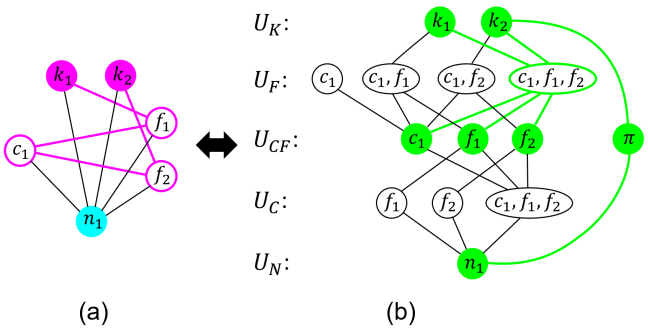


Fig. 8. The graph H (b) constructed for the NWST problem from the original graph G (a). Each node of H is annotated by the corresponding nodes in G, and terminal nodes are shaded green. The highlighted subgraph of H (green nodes and edges) encodes the labelling of G.

(1) If S is CF-Disjoint, then the following labelling on G is a solution to ACAP-TL: a cut or fill node s is labelled 1 (resp. 0) if $s \in V(p)$ for some $p \in U_F \cap U_S$ (resp. $p \in U_C \cap U_S$).
(2) If S is not CF-Disjoint, then a solution of ACAP-TL on G does not exist.

We prove the proposition in Appendix B. As an example, the labelling in Figure 8 (a) (same as the optimal labelling in Figure 6 (c)) can be obtained from the subgraph in (b) (highlighted in green), which connects all terminals and is CF-Disjoint. With this proposition, we can solve the ACAP-TL problem on G by solving the NWST problem on H and checking the result for CF-Disjointness.

6.3 Algorithm

The previous section presented a globally optimal strategy to solve ACAP-TL. To solve the original TL problem, we need to address two issues stemming from the NWST-based approach and the differences between ACAP-TL and TL as mentioned earlier. First, the subgraph produced by the NWST solver may not be CF-Disjoint. This may happen either because a solution to ACAP-TL does not exist, or because the NWST solver failed to solve to optimality. Second, since the energy of ACAP-TL is different from that of TL, a labelling that minimizes the former may not be minimal for the latter.

To address the first issue, we propose several greedy heuristics to modify the graph G to improve the odds of solving ACAP-TL. As discussed in the previous section, an ACAP labelling does not exist if there is a *double-articulation node* (e.g., f_1 in Figure 7 (b)). Once detected, a double-articulation node s is resolved by "sending" it to the kernel (or neighborhood); that is, s is labelled 1 (or 0) and merged with its connected kernel (or neighborhood) nodes (see Figure 7 (c)). The choice of kernel or neighborhood is made to result in a greater decrease in the energy E_G. If multiple double-articulation nodes are found, the one that would result in the greatest decrease in energy is chosen to be resolved. After resolving one node, we check the graph again for new double-articulation nodes, and the process repeats until no more such nodes are found.

Even with all double articulation nodes resolved, the NWST solver may still return a subgraph S of H that is not CF-Disjoint. In this case, we need to further modify the graph G to make ACAP-TL

```
TopologyLabelling (G)
    Repeat:
        Resolve all double-articulation nodes from G
        Construct H from G
        S ← Solve NWST on H
        If S is CF-Disjoint
            Break
        Else
            Resolve a doubly-labelled node from G
    L ← Labelling obtained from S
    Improve L by greedy strategy
    Return L
```

Fig. 9. Pseudo-code of the labelling algorithm.

easier to solve. Since S is not CF-disjoint, there exists some cut or fill node s of G that is represented by a node in $U_F \cup U_S$ and a node in $U_C \cup U_S$. We call s a *doubly-labelled node*. We solve a doubly-labelled node in the same way that we resolve a double-articulation node - by picking the doubly-labelled node that results in the greatest decrease in E_G and sending it to the neighborhood or kernel. Once a doubly-labelled node is resolved, we check and resolve double-articulation nodes again (since the graph is changed), and solve for ACAP-TL. The process is repeated until a CF-disjoint subgraph of H is found.

To alleviate the second issue (that the energies of ACAP-TL and TL are different), we take the solution of ACAP-TL as the initial labelling and apply the greedy strategy mentioned in Section 6.1 to further reduce the TL energy. In our tests, however, we have found that the greedy strategy rarely improves upon the labelling produced by ACAP-TL, but we include this step nevertheless for the occasional improvement.

The complete algorithm is summarized in the pseudo-code in Figure 9. Note that the iterative graph modification is guaranteed to terminate, because each iteration sends at least one doubly-labelled node to the neighborhood or kernel, and hence the number of cut and fill nodes to be labelled is monotonically decreasing.

6.4 Improving scalability

The optimality of our algorithm largely depends on the optimality of the NWST solver. Since the NWST problem is NP-hard, the chance of a solver finding an optimal (or close-to-optimal) solution decreases with the graph size. We describe two strategies to reduce the graph size, which improves both the efficiency and optimality of our algorithm on large graphs.

Cluster simplification. The first strategy is to solve the labelling problem on smaller subgraphs whenever possible, before solving it on the entire graph. Consider a maximal set of connected cut and fill nodes W, which we call a *cluster*. There are two types of clusters that can be simplified before solving ACAP-TL on G. If W is connected to only neighborhood (resp. kernel) nodes, then all nodes in W must be labelled 0 (resp. 1) for the labelling to be ACAP. In this case, we simply send the entire set W to the neighborhood or kernel, as we did for double-articulation and doubly-labelled nodes. Another interesting scenario is when W is connected to exactly one kernel node k and one neighborhood node n. It is easy to show that, if labelling L is a solution of ACAP-TL on the whole graph G, then the restriction of L to the subgraph S spanning nodes $W \cup \{k, n\}$ is a solution of ACAP-TL on S. We therefore apply the same iterative solver (without the final greedy step) to S, and afterwards send the 1-labelled (resp. 0-labelled) nodes of W to the kernel (resp. neighborhood). If enabled, this cluster-simplification step takes place at the beginning of each iteration in our solver (right after the line "Repeat" in the pseudocode of Figure 9). Note that processing of different clusters are independent of each other, and hence they can be trivially parallelized.

Node pruning. While cluster simplification does not affect the optimality of our algorithm, we propose another strategy that trades theoretical optimality for practical optimality and efficiency. We have observed that the bulk of the transformed graph H consists of nodes U_C, U_F (representing all cuttable and fillable sets) and their incident edges. In Appendix C, we show that solving NWST with a reduced set of U_C, U_F still leads to a proper and ACAP labelling of G. Even though such labelling may not minimize the ACAP-TL energy, the reduced graph size could make the NWST solver more successful in finding a close-to-optimal solution in practice. We have found the following pruning scheme to be quite effective in our experiments. We measure the *hops* between two fill (resp. cut) nodes s, t as the least number of cut (resp. fill) nodes on any path from s to t in G. We keep a node $p \in U_F$ (resp. $p \in U_C$) only if the maximum hops between two fill (resp. cut) nodes in $V(p)$ is no more than K, a user-specified number.

7 RESULTS

We evaluate our algorithm on cuts and fills produced by existing monotonic simplification methods. In particular, we consider morphological opening/closing [Nooruddin and Turk 2003] and topology-controlled inflation/deflation [Bischoff and Kobbelt 2002; Kriegeskorte and Goebel 2001; Szymczak and Vanderhyde 2003]. These methods are simple to implement and capable of removing all types of topological features. Note that these methods all operate on a voxel grid, which is a cell complex with hexahedral cells (i.e., voxels). In the following, we shall refer to a cut or fill cell as a *cut or fill voxel*. We compare our method to the TopoMender method [Zhou et al. 2007], a monotonic simplification method that removes topological handles using a minimum spanning tree; the MendIT method [Ju et al. 2007], which is, to the best of our knowledge, the only non-monotonic method capable of removing all three types of topological features; and the greedy strategy in Section 6.1 without the full algorithm in Section 6.3.

We solve the NWST problem using the recent branch-and-bound algorithm of [Leitner et al. 2018], which performed well on public benchmarks (e.g., the 11th DIMACS Challenge [DIM 2014]) as well as on our own data. A nice feature of the implementation is that it allows the user to set a maximum running time, which is useful when the problem size is large. We found that the solver typically returns an optimal solution within seconds for small or medium-sized graphs (e.g., containing up to thousands of nodes). For larger

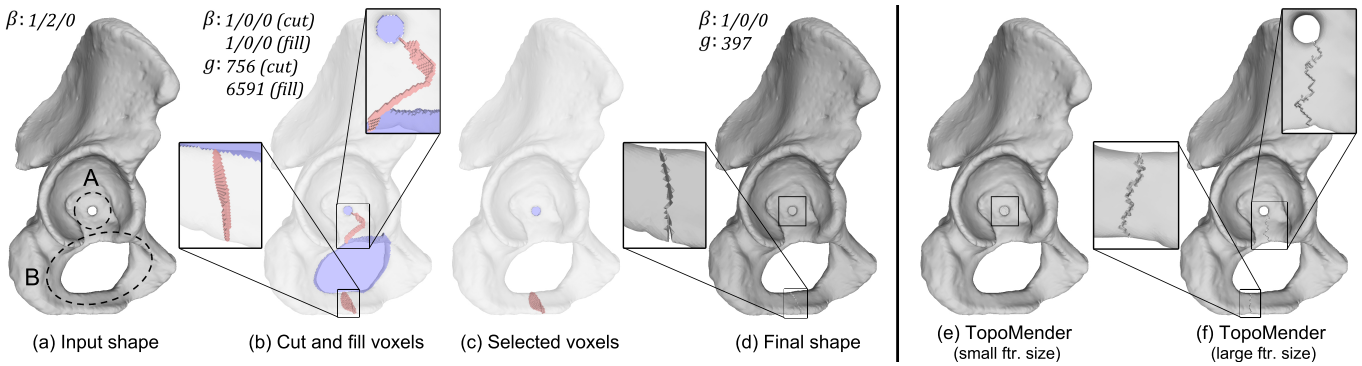


Fig. 10. (a): The Hip model with two handles (marked A and B). (b): Cut voxels (red) and fill voxels (blue) produced by topology-controlled inflation and deflation. (c): Selected fill voxels (filling in the small handle hole of A) and cut voxels (cutting open the narrow part of the handle body of B). (d): The modified shape. (e,f): Results of TopoMender using a small parameter (e), which leaves the handle B in the shape, or using a large parameter (f), which cuts open the wide handle body of A. (β: number of connected components, handles, and cavities; g: geometric cost)

graphs, the optimality of the solution often does not improve after a few seconds. As a result, we set the time limit to be 8 seconds regardless of graph size. Unless stated otherwise, we apply the basic algorithm described in Section 6.3 without employing the optional heuristics described in Section 6.4.

7.1 Distance-based cuts and fills

We first consider the cuts and fills produced by inflating or deflating an initial *seed* towards the shape [Bischoff and Kobbelt 2002; Kriegeskorte and Goebel 2001; Szymczak and Vanderhyde 2003]. Starting with all voxels in a bounding box as the seed, this approach iteratively removes voxels that lie outside the shape while preventing topological changes. The removal is prioritized by the Euclidean distance function from the shape's boundary, so that voxels further away from the shape are removed before those closer to the shape. The deflation results in a minimal set of fill voxels whose addition to the shape removes all topological features. Cut voxels can be created by the reverse process, inflating from a seed inside the shape (we use the voxel that is furthest away from the shape's boundary). Each cut or fill voxel v is assigned a constant cost, i.e., $g(v) = 1$.

Figure 10 shows a simple example (the Hip). The shape in (a) has two handles, one with a small handle hole and a wide handle body (A), and another with an expansive handle hole and a narrower handle body (B). As shown in (b), removing all cuts would make a long opening in the handle body of A, while adding all fills would create a large patch in the handle hole of B. Our method picks a subset of both cut and fill voxels to take the less invasive operation for each handle, leading to a lower geometric cost as shown in (c,d). We also show the results, in (e,f), of the monotonic method TopoMender. TopoMender takes in a user-given feature size parameter, and cuts (resp. fills) all handles whose handle body (resp. handle hole) has a smaller radius than the parameter. To perform both cutting and filling, we ran TopoMender twice, first cutting all small handle bodies and next filling all small handle holes. However, it can be challenging to choose the right feature size to get desirable results: a too-small parameter leaves the handle B in the shape, while a too-large parameter tears open the wide handle body of A.

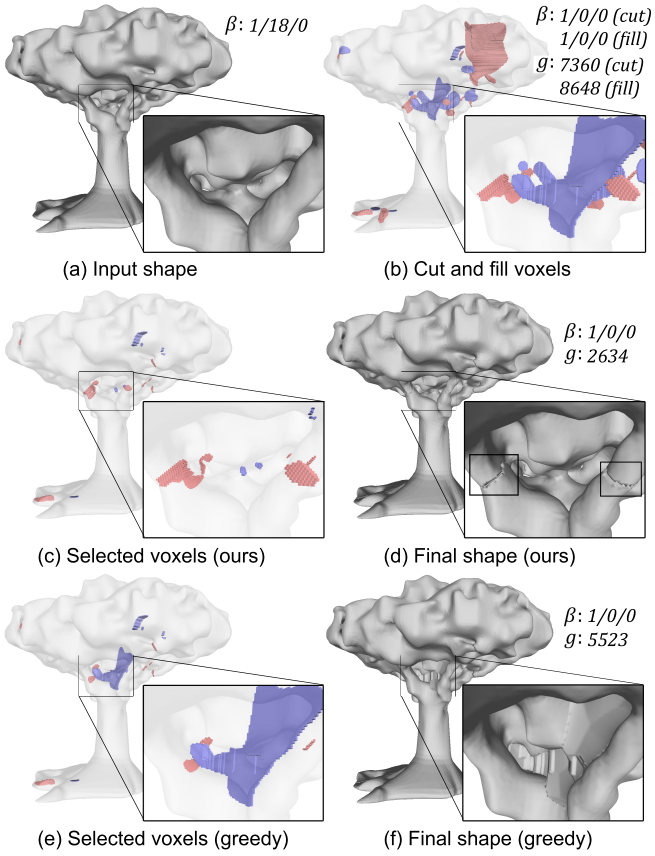


Fig. 11. (a): The Tree shape with 18 handles. (b): Input cut (red) and fill (blue) voxels. (c): Cut and fill voxels selected by our algorithm. (d): The modified shape. (e): Cut and fill voxels selected by the greedy strategy, which include a large group of fill voxels. (f): Resulting shape of the greedy strategy.

In contrast, our algorithm automatically picks the least costly way to remove each feature.

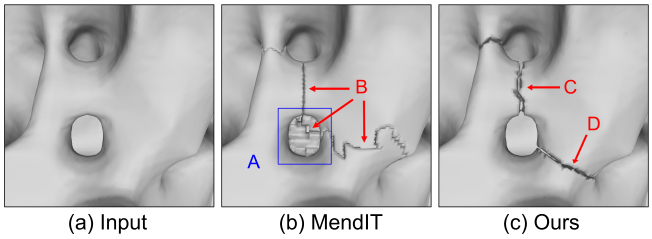


Fig. 12. Comparing result of our method (c) and MendIT (b) on a region of the Tree with two handles (a). While MendIT fills a handle hole (A) and makes a long cut (B) on the handle body, our method makes two short cuts (C,D) without filling the handle hole.

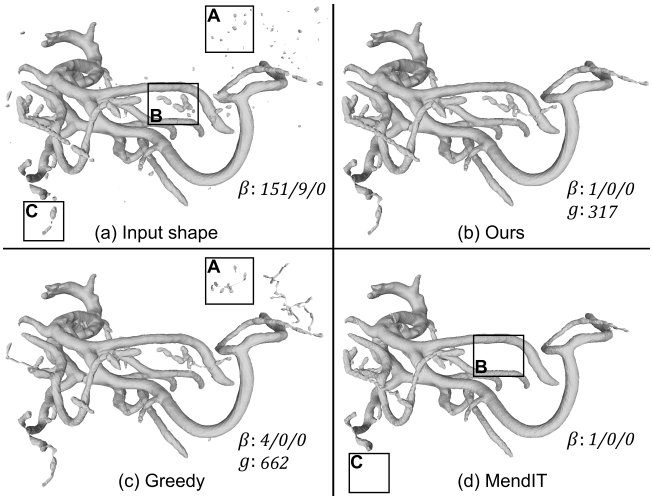


Fig. 13. Comparing the result of our algorithm (b), the greedy labelling strategy (c), and MendIT (d) on the Vessel example from Figure 1. To reduce the number of components, our method removes small and distant islands (box A) while connecting to larger ones (box B and C). The other (greedy) methods either fails to remove all components in the result (c) or misses non-trivial components (d).

Figure 11 shows a more complex example (the Tree). The input shape in (a) contains 18 handles. Either removing all cuts or adding all fills would remove all handles, but at the cost of significant geometric changes, as evident in (b). Our algorithm selects a small subset of cut and fill voxels to achieve the same topological goal but with a much lower geometric cost, as shown in (c,d). As a comparison, running the greedy strategy of Section 6.1 alone also removes all handles, but at a higher geometric cost, as shown in (e,f). The close-up view examines a region where our full algorithm based on global optimization differs with the greedy strategy. The greedy strategy selected a large group of fill voxels, which fills in two handles at once and thus was prioritized for labelling, despite its large geometric cost. In contrast, our algorithm selected two smaller groups of cut voxels that remove the same handles at a lower cost. To further demonstrate the advantage of global optimization, Figure 12 examines our method on a different region of the Tree and compares with the greedy method of MendIT. MendIT starts by

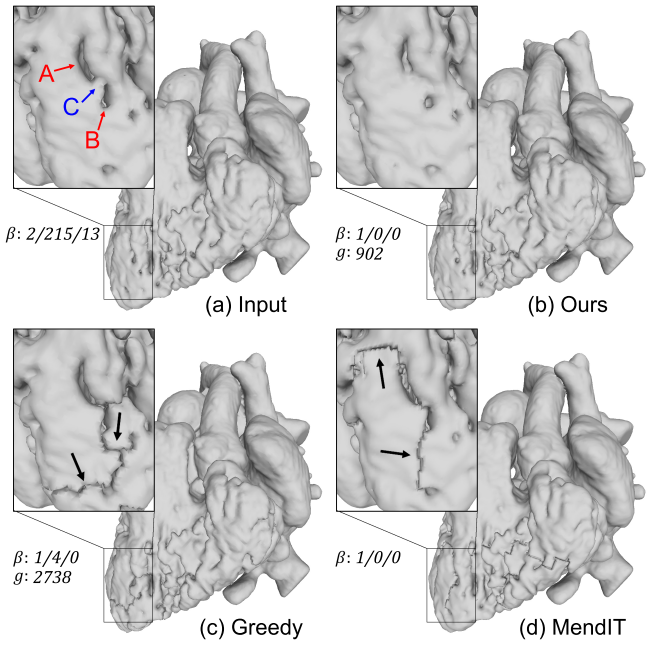


Fig. 14. Comparing the result of our algorithm (b), the greedy labelling strategy (c), and MendIT (d) on the Heart example. Both our method and MendIT fully simplify the topology, while the greedy strategy leaves 4 handles in the shape. The inserts show a region where both the greedy strategy and MendIT make excessive geometric changes.

filling the handle hole (A), which has a low geometric cost. However, filling that hole creates a thick handle body that has to be removed by a long cut (B), which un-does the earlier filling. Our algorithm removes the same number of handles with two short cuts (C,D) and no fills.

The Vessel in Figure 1 contains multiple connected components and handles. Once again, while performing only cutting or only filling results in excessive geometric changes (highlighted in the boxes A,B,C), our algorithm selects a geometrically minimal set of cut and fill voxels to achieve the same amount of topological reduction. Figure 13 compares our method with the greedy strategy of Section 6.1 and MendIT on the same example. Due to their greedy nature, these two methods can make globally sub-optimal choices. The greedy strategy left a few islands in the result (box A in (c)), while MendIT removed a few non-trivial components from the input (boxes B,C in (d)), similar to the scenario illustrated in Figure 3.

The Heart in Figure 14 is an even more complex example with multiple connected components, handles, and cavities. Observe that our method is able to fully simplify the topology, while the greedy strategy leaves a few handles behind and incurs a larger geometric cost. The inserts compare our method with the greedy strategy and MendIT in a region where two large handle holes (A,B) are separated by a narrow handle body (C). Both greedy approaches choose to first cut the handle body C, as it has a low geometric cost. However, the cut merges the two handles holes A and B into an even larger handle hole. This forces the two methods to subsequently make several large cuts (indicated by arrows) to avoid filling the

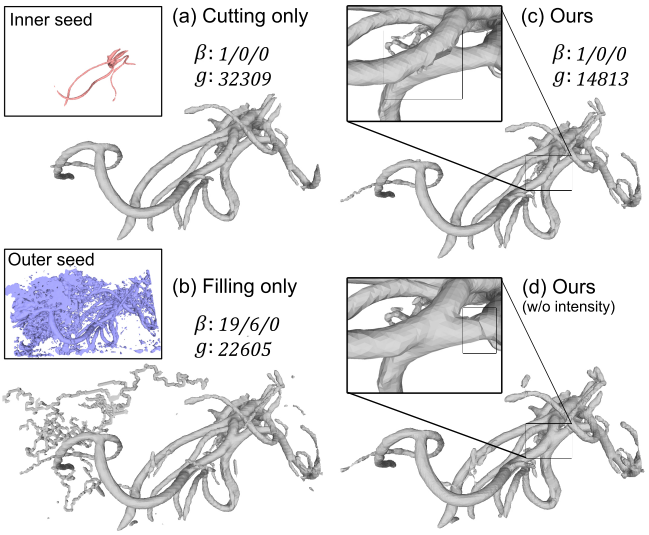


Fig. 15. The shape after subtracting intensity-aware cut voxels (a) or adding intensity-aware fill voxels (b), and after running our algorithm with the intensity-aware cuts and fills (c) or distance-based cuts and fills (d). The inserts in (a,b) show the inner and outer seeds used for inflation and deflation. The inserts in (c,d) examine a region where the same handle is removed differently in the two results, either separating the two vessels (c) or breaking a vessel in the middle (d).

merged handle hole. Thanks to the global optimization approach, our algorithm avoids making these large cuts by filling the handle holes A and B without cutting the handle body C.

7.2 Intensity-aware cuts and fills

A 3D shape is often represented as the iso-surface of an intensity function, such as a signed distance function reconstructed from point clouds or a 3D MRI or CT scan. When the intensity function is available, it can be used to guide the inflation and deflation process to create *intensity-aware* cuts and fills [Bischoff and Kobbelt 2002]. Basically, the Euclidean distance field is replaced by the intensity function to prioritize voxels during inflation or deflation process. That is, voxels whose intensity values are further from the iso-value of the shape are removed earlier than voxels whose intensity values are closer to the iso-value. Accordingly, we assign a non-uniform geometric cost to a cut or fill voxel as the magnitude of the intensity gradient. This cost penalizes using voxels on strong intensity edges. The cost reduces to a constant when the intensity function is the Euclidean distance field, making our choice consistent with the earlier scenario when the intensity function is not available.

We make two more changes to the inflation and deflation process to make it more practical. First, since the intensity values are often only available (or reliable) within a certain range, using a bounding box or an interior voxel as the seed may not work if the seed contains voxels whose values are outside that range. Instead, for a user-specified intensity range $[I_{low}, I_{high}]$, and assuming the interior of shape has higher intensity than the iso-value, we define an intensity-based seed as the set of all voxels whose values are above I_{low} (for deflation) or I_{high} (for inflation). Second, unlike the bounding box or a single voxel, the intensity-based seed may have a complex topology. To create cut or fill voxels that simplify, and not complicate, topology, we modify the inflation and deflation routine to allow any topological changes that reduces the number of components, handles or cavities.

We first apply the intensity-based inflation and deflation to the Vessel shape from Figure 1. This shape was originally obtained as an iso-surface of a 3D CT scan, which we use as the intensity function. As shown in Figure 15 (c,d), compared with distance-based cut and fill voxels, the use of intensity-aware cuts and fills allows our method to make a "nicer" cut that separates two close-by vessels instead of breaking a vessel in the middle. Note that the intensity-based outer seed has a significant amount of topological noise. As a result, the fill voxels alone do not fully simplify the topology. By utilizing both cut and fill voxels, our algorithm achieves full reduction of topology at a lower geometric cost than performing either cutting or filling.

We next perform intensity-aware cutting and filling on two large examples (the Panicle and the Root) in Figures 16,17. Both shapes are obtained as iso-surfaces of CT scans, and contain thousands of topological features including handles, islands, and cavities. Due to the topological noise present in the seeds used for inflation and deflation, neither the set of cut voxels nor the set of fill voxels can simplify the topology effectively by themselves, and hundreds of topological features still remain after performing only cutting or only filling (a few are highlighted in the inserts). By combining both sets of voxels, our method achieves significantly more reduction in topology than either cutting or filling. Figure 18 zooms in on a complex region of the root that has handles, cavities and islands, which results in a cluster of cut and fill voxels. Our method selects both cut and fill voxels to remove these features.

In both of these examples, the initial graph G contains thousands of nodes and edges, and the transformed graph for NWST, H, is too large to hold in memory. As a result, we use the strategies discussed in Section 6.4 to reduce the graph size. Table 1 examines the effect of cluster simplification and node pruning (with different parameter K) on the size of H, the overall running time, and the resulting energy (both topology and geometry) using the Root example. Observe that cluster simplification is particularly effective in reducing the graph size, which leads to both significantly faster runtime and more

Cluster?	K	$\|U\|/\|A\|$	Time (sec)	$\beta_0/\beta_1/\beta_2$	g
Yes	1	745/29990	35.1	32/6/0	38637
Yes	2	1089/32841	39.8	32/6/0	38692
Yes	3	5586/49596	52.8	32/9/0	38431
Yes	4	45130/420373	126.0	32/13/0	41242
No	1	24791/258512	23573.3	32/11/0	51343
No	2	31261/302879	30748.4	32/13/0	47192
No	3	54604/499427	31200.6	32/11/0	47836

Table 1. Effect of using clustering simplification (first column) and choices of K for node pruning (second column) on the number of nodes and edges of H (taken as the maximum size among all clusters and over all iterations of the algorithm), overall runtime of our algorithm, and the Betti numbers and the geometric cost of the solution.

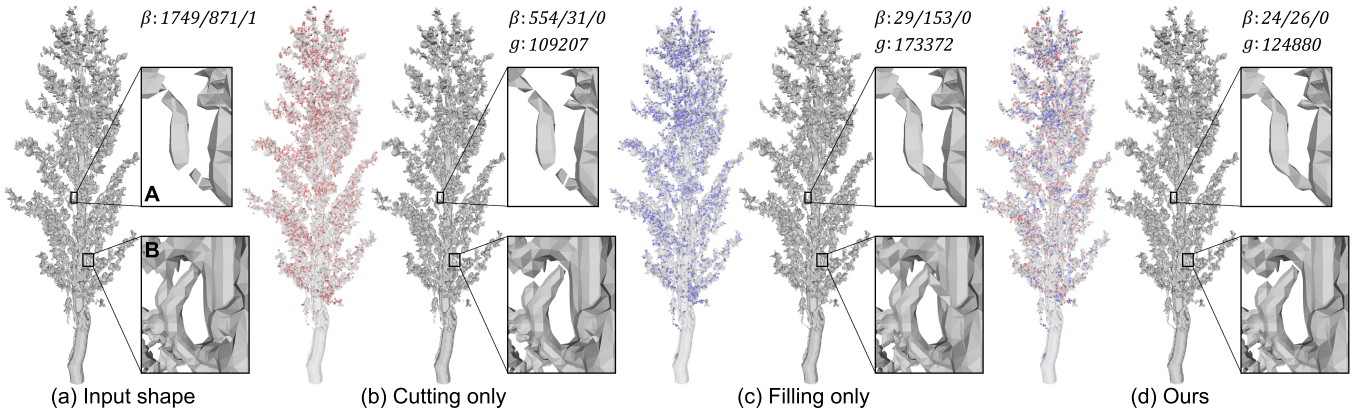


Fig. 16. (a): An iso-surface from a CT scan of a sorghum panicle. (b,c): Results of intensity-aware cutting and filling (cut/fill voxels shown on the left). (d): Our result. Box A highlights a few islands that are only connected by filling, and box B highlights two handles that are only removed by cutting. Our method resolves both features.

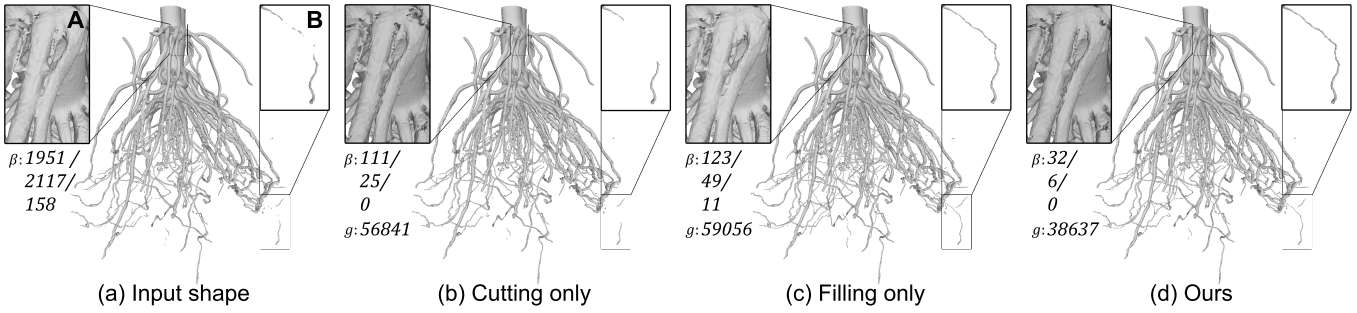


Fig. 17. (a): A highly complex iso-surface from a CT scan of a corn root. (b,c): Results of intensity-aware cutting and filling. (d): Our result. Box A highlights a handle that is only removed by cutting (which separates the two root branches), and box B highlights islands that are only connected by filling. Our method resolves both features.

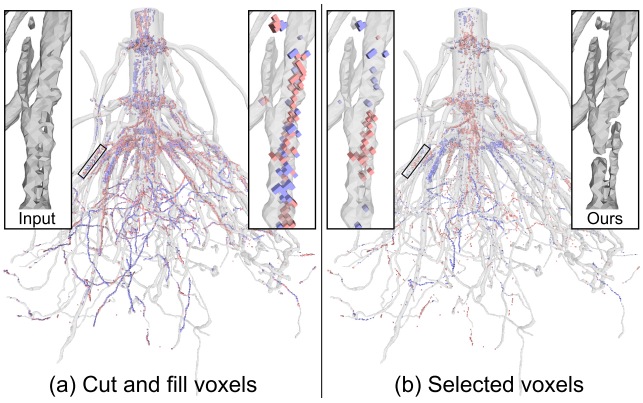


Fig. 18. Cut and fill voxels (a) and those selected by our algorithm (b) on the Root. The inserts highlight a cluster of cut and fill voxels in a region with complex topology.

optimal solutions by the NWST solver. While larger K should lead to improved optimality in theory (by pruning fewer nodes of H), it also results in larger graph sizes and hence deteriorating solutions. The results shown in Figures 16,17 are produced with $K = 1$.

7.3 Cuts and fills from opening and closing

Morphological opening and closing are common ways to reduce topological noise on a voxelized shape T. Given a structure element B (e.g., a box or voxelized ball), opening computes the largest union of B that can fit inside T, and closing computes the complement of the largest union of B that can fit outside T. An example is shown for another Heart example in Figure 19. Observe that the small handle on the shape shown in the insert of (a) is removed by either opening (b) or closing (c).

There are several limitations of opening and closing. First, they may introduce new topological features that are not present on the original shape. For example, opening may create new islands (e.g., box A in (b)) while closing may merge different shape parts to form new handles (e.g., box B in (c)). Second, both operators make expansive modification of the shape, most of which do not contribute to reduction in topology (as seen from the cut and fill voxels at the top of (b,c)). Lastly, both operations are monotonic, since opening only cuts while closing only fills.

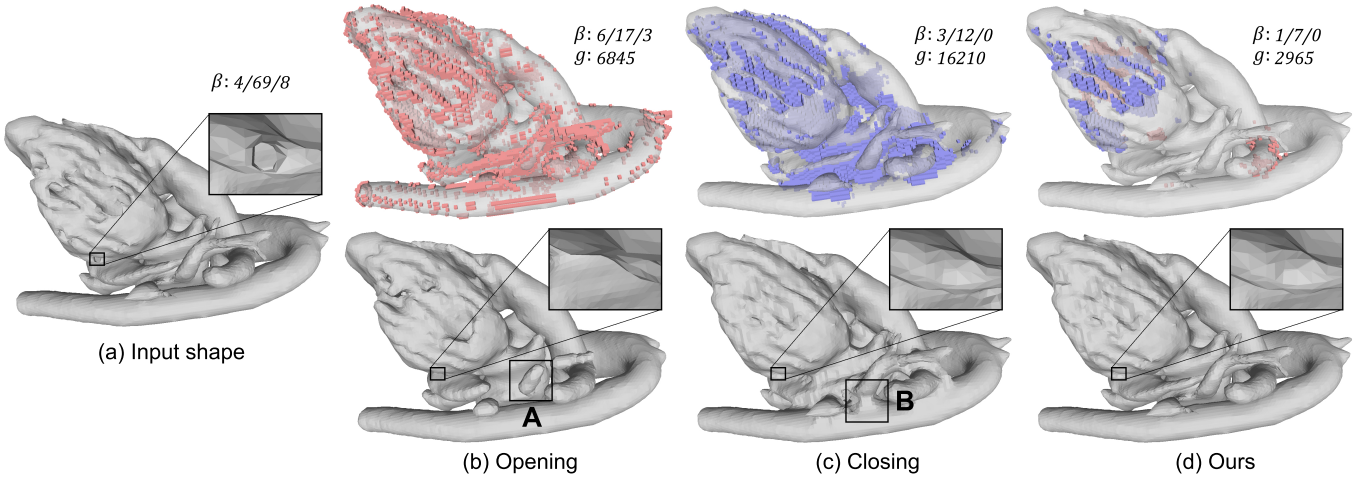


Fig. 19. (a): A heart segmentation. (b): Result of opening (bottom), which contains several new islands (box A), and the corresponding cut voxels (top). (c): Result of closing (bottom), which merges nearby vessels (box B) and thereby creating new handles, and the corresponding fill voxels (top). (d): Voxels selected by our algorithm (top) and the modified shape (bottom). The insert examines a handle that is removed in all three methods.

These limitations can be addressed by running our algorithm on the results of both opening and closing. We take the difference between the shape and its opening (resp. closing) as the cut (resp. fill) voxels, and assign each voxel a unit geometric cost. Since the majority of the changes made by opening or closing are not topology-related, and to reduce graph size, we pre-process the graph G to send a cut (resp. fill) node s to the kernel (resp. neighborhood) if $\chi(\Omega_s) = 0$ and if labelling s as 0 (resp. 1) does not change the number of connected components of either the 0-labelled or 1-labelled subgraph. As shown in Figure 19 (d), our algorithm removes more topological features and incurs a lower geometric cost than either opening or closing.

7.4 Performance

We report detailed statistics of all examples in this paper in Table 2. These include the graph sizes (G and H), running time, and the objective energy (both topology and geometry) of the results produced by our algorithm and other variants. All experiments are performed on a Windows PC with 3.47Hz CPU and 24G RAM.

We make several notes about the table. First, we did not report the energy of our method before applying the greedy improvement step (second-to-last line in Figure 9), because that step did not improve the overall energy for any of the examples in this table. The only experiments that we noticed some improvement was when the graph size is so large that NWST returns a highly sub-optimal solution (e.g., when cluster simplification is turned off on the Root or Panicle). Second, observe that our NWST-based labelling algorithm almost always outperforms the greedy strategy. The difference in the topological complexity between these two strategies becomes significant for large graphs, which shows the importance of a global optimization approach for the labelling problem.

Lastly, although we do not know the optimal energy for the labelling problem, we can provide a lower bound on the Betti numbers as the *persistent Betti numbers* of the inclusion map $\Omega(T \setminus C) \rightarrow \Omega(T \cup F)$. These numbers describe the least possible number of topological features for any shape sandwiched between $\Omega(T \setminus C)$ (cutting only) and $\Omega(T \cup F)$ (filling only). Note that, in 3D, a shape realizing the persistent Betti numbers may not exist, and finding the sandwiched shape with the simplest topology is an NP-hard problem (homological simplification) [Attali et al. 2015]. Nevertheless, these numbers offer a theoretical lower bound of how much topology we can simplify. We compute the persistent Betti numbers using the DIPHA software [Bauer et al. 2014], which conveniently handles cubical complexes. Observe from the table that our algorithm realizes this lower bound on simple examples while getting close to it on large examples. In particular, we are 20 features away from the lower bound on the Panicle, and only 3 handles away from the lower bound on the Root example, which initially contains over 4000 topological features.

8 DISCUSSION

We presented an algorithm for maximally reducing topological complexity of a 3D shape while minimizing geometric changes by selecting from a given set of cuts and fills. To the best of our knowledge, this is the first attempt at solving topological simplification in its full generality (treating all types of topological features and allowing both cutting and filling) as a global optimization problem.

A key limitation of our work is that the choice of the input set of cuts and fills can have a major impact on how the solution of our graph labelling problem, formulated in Section 5, approximates the solution to the cell selection problem, stated in Section 4. Since graph labelling operates at the level of R-connected components of cells, the granularity of such components directly affects the solution space. In particular, the labelling algorithm would return a poor approximation if adding (resp. subtracting) an R-connected component of fill (resp. cut) cells simultaneously removes some topological features and introduce new ones. We have observed

Example	$\beta_0/\beta_1/\beta_2$ (shape)	$\beta_0/\beta_1/\beta_2$ (persistent)	$\beta_0/\beta_1/\beta_2; g$ (cutting)	$\beta_0/\beta_1/\beta_2; g$ (filling)	$\beta_0/\beta_1/\beta_2; g$ (greedy)	$\beta_0/\beta_1/\beta_2; g$ (ours)	$\|V\|/\|E\|$	$\|U\|/\|A\|$	Time (sec)
Figure 10	1/2/0	1/0/0	1/0/0; 756	1/0/0; 6591	1/0/0; 397	1/0/0; 397	5/7	12/16	3.1
Figure 11	1/18/0	1/0/0	1/0/0; 7360	1/0/0; 8648	1/0/0; 5523	1/0/0; 2634	37/94	138/390	3.8
Figure 19	4/69/8	1/0/0	6/17/3; 6845	3/12/0; 16210	2/10/1; 12198	1/7/0; 2965	46/104	147/262	0.01
Figure 1	151/9/0	1/0/0	1/0/0; 866	1/0/0; 1534	4/0/0; 662	1/0/0; 317	202/413	4556/53188	24.3
Figure 15	151/9/0	1/0/0	1/0/0; 31309	19/6/0; 22605	6/0/0; 30735	1/0/0; 14813	207/413	3202/42404	45.4
Figure 14*	2/215/13	1/0/0	1/0/0; 3070	1/0/0; 3264	1/4/0; 2738	1/0/0; 902	376/1076	753/29680	3.1
Figure 16*	1749/871/1	20/10/0	554/31/0; 109207	29/153/0; 173372	97/25/0; 118948	24/26/0; 124880	3341/7391	169/585	5.5
Figure 17*	1951/2117/158	32/3/0	111/25/0; 56842	123/49/11; 59056	116/169/1; 47341	32/6/0; 38637	6070/14881	745/29990	35.1

Table 2. Statistics for all examples in the paper (ordered by topological complexity): Betti numbers of the shape; persistent Betti numbers of the inclusion map from the all-cut shape to the all-filled shape; Betti numbers and geometric cost for cutting only, filling only, greedy labelling, and our method; number of nodes and edges of G and H; and running time. The * indicates the use of cluster simplification and node pruning (with $K = 1$).

many such components in the cuts and fills generated by morphological opening and closing, where our algorithm produced a less simplified result (see Table 2). We would like to explore ways to segment cells (voxels) at a finer level than R-connectivity, while still being able to formulate a graph labelling problem that is amenable for optimization.

While our current implementation is specialized to hexahedral cell complexes in the form of a voxel grid, our algorithm applies to any cell complexes. In the future, we plan to develop implementations for general cell complexes in 3D, such as tetrahedral meshes. We would also like to explore extension of the algorithm to allow for more user control. For example, prescribing Betti numbers (e.g., a single connected component with k handles) would be useful for shapes that have a known topological structure. Finally, an orthogonal direction of investigation is how to obtain better geometric costs for cuts and fills to better capture the semantics of the shape and produce more natural-looking shape modifications.

ACKNOWLEDGMENTS

This work is supported by NSF grants CCF-1614562, DBI-1759836, CCF-1907612, EF-1921728, and NIH grant CA233303-1. Dan Zeng is supported by an Imaging Science Pathway fellowship at Washington University in St. Louis. We would like to thank Christopher Topp for providing the Sorghum and Root data.

REFERENCES

2014. 11th DIMACS Implementation Challenge in Collaboration with ICERM: Steiner Tree Problems. http://dimacs11.zib.de/

Dominique Attali, Ulrich Bauer, Olivier Devillers, Marc Glisse, and André Lieutier. 2015. Homological reconstruction and simplification in R3. *Computational Geometry* 48, 8 (2015), 606–621.

Marco Attene, Marcel Campen, and Leif Kobbelt. 2013. Polygon Mesh Repairing: An Application Perspective. *ACM Comput. Surv.* 45, 2 (March 2013), 15:1–15:33.

Ulrich Bauer, Michael Kerber, and Jan Reininghaus. 2014. Distributed computation of persistent homology. In *2014 proceedings of the sixteenth workshop on algorithm engineering and experiments (ALENEX)*. SIAM, 31–38.

Stephan Bischoff and Leif Kobbelt. 2002. Isosurface Reconstruction with Topology Control.. In *Pacific Conference on Computer Graphics and Applications*. 246–255.

Yuri Boykov, Olga Veksler, and Ramin Zabih. 2001. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence* 23, 11 (2001), 1222–1239.

Erin W Chambers, Tao Ju, David Letscher, Mao Li, and Christopher Topp. 2018. Some Heuristics for the Homological Simplification Problem. In *CCCG*.

Lin Chen and Gudrun Wagenknecht. 2006. Automated topology correction for human brain segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 316–323.

Xiao Han, Chenyang Xu, Ulisses Braga-Neto, and Jerry L. Prince. 2002. Topology Correction in Brain Cortex Segmentation Using a Multi-Scale, Graph-Based Algorithm. *IEEE Trans. Med. Imaging* 21, 2 (2002), 109–121.

Xiao Han, Chenyang Xu, and Jerry L. Prince. 2003. A topology preserving level set method for geometric deformable models. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25, 6 (2003), 755–768.

Allen Hatcher. 2002. *Algebraic Topology*. Cambridge University Pres.

Zhiyang Huang, Ming Zou, Nathan Carr, and Tao Ju. 2017. Topology-controlled Reconstruction of Multi-labelled Domains from Cross-sections. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–12.

Tao Ju, Qian-Yi Zhou, and Shi-Min Hu. 2007. Editing the Topology of 3D Models by Sketching. *ACM Trans. Graph.* 26, 3, Article 42 (July 2007). https://doi.org/10.1145/1276377.1276430

N. Kriegeskorte and R. Goebel. 2001. An efficient algorithm for topologically correct segmentation of the cortical sheet in anatomical mr volumes. *Neuroimage* 14, 2 (August 2001), 329–346.

Roee Lazar, Nadav Dym, Yam Kushinsky, Zhiyang Huang, Tao Ju, and Yaron Lipman. 2018. Robust optimization for topological surface reconstruction. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–10.

Markus Leitner, Ivana Ljubić, Martin Luipersbeck, and Markus Sinnl. 2018. A dual ascent-based branch-and-bound framework for the prize-collecting steiner tree and related problems. *INFORMS Journal on Computing* 30, 2 (2018), 402–420.

Fakir S. Nooruddin and Greg Turk. 2003. Simplification and Repair of Polygonal Models Using Volumetric Techniques. *IEEE Trans. Vis. Comput. Graph.* 9, 2 (2003), 191–205.

Florent Ségonne. 2008. Active contours under topology control - genus preserving level sets. *International Journal of Computer Vision* 79, 2 (2008), 107–117.

Florent Ségonne, Jenni Pacheco, and Bruce Fischl. 2007. Geometrically accurate topology-correction of cortical surfaces using nonseparating loops. *IEEE transactions on medical imaging* 26, 4 (2007), 518–529.

Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. 2006. Competing fronts for coarse–to–fine surface reconstruction. In *Eurographics*. Vienna, 389–398. http://www.mat.puc-rio.br/~tomlew/competing_fronts_eg.pdf

Andrei Sharf, Thomas Lewiner, Gil Shklarski, Sivan Toledo, and Daniel Cohen-Or. 2007. Interactive Topology-aware Surface Reconstruction. *ACM Trans. Graph.* 26, 3 (July 2007).

David W. Shattuck and Richard M. Leahy. 2001. Automated Graph Based Analysis and Correction of Cortical Volume Topology. *IEEE Trans. Med. Imaging* 20, 11 (2001), 1167–1177.

Andrzej Szymczak and James Vanderhyde. 2003. Extraction of topologically simple isosurfaces from volume datasets. In *IEEE Visualization*. 67–74.

Zoë J. Wood, Hugues Hoppe, Mathieu Desbrun, and Peter Schröder. 2004. Removing excess topology from isosurfaces. *ACM Trans. Graph.* 23, 2 (2004), 190–208.

Kangxue Yin, Hui Huang, Hao Zhang, Minglun Gong, Daniel Cohen-Or, and Baoquan Chen. 2014. Morfit: Interactive Surface Reconstruction from Incomplete Point Clouds with Curve-driven Topology and Geometry Control. *ACM Trans. Graph.* 33, 6 (Nov. 2014), 202:1–202:12.

Yun Zeng, Dimitris Samaras, Wei Chen, and Qunsheng Peng. 2008. Topology cuts: A novel min-cut/max-flow algorithm for topology preserving segmentation in N-D images. *Computer Vision and Image Understanding* 112, 1 (2008), 81–90.

Qian-Yi Zhou, Tao Ju, and Shi-Min Hu. 2007. Topology Repair of Solid Models Using Skeletons. *IEEE Transactions on Visualization and Computer Graphics* 13, 4 (July 2007), 675–685.

Ming Zou, Michelle Holloway, Nathan Carr, and Tao Ju. 2015. Topology-constrained surface reconstruction from cross-sections. *ACM Trans. Graph.* 34, 4 (2015), 128.

A PROOF OF PROPOSITION 5.2

We start by associating a set of cells (at all dimensions) to each node and showing some properties of these sets. Define Ω_s as the subset of cells in the closure $\Omega(O_s)$ that are not faces of any 3-cell ranking higher than O_s. Note that $O_s \subseteq \Omega_s \subseteq \Omega(O_s)$ by definition. Recall that two sets of cells are *connected* if some cell in one set is the face of a cell in the other set. We can show that,

Lemma A.1. *The following properties hold:*

(1) For any cell $c \in \Omega$, there exists a unique node $s \in V$ such that $c \in \Omega_s$. That is, Ω_s of all nodes $s \in V$ form a disjoint decomposition of Ω.
(2) For any node $s \in V$, Ω_s is connected.
(3) For any two nodes $s, t \in V$, they are connected by an edge if and only if Ω_s and Ω_t are connected.

Proof. We prove each property in turn.

(1) Consider the set H of highest-ranking 3-cells that c is a face of. If H contains more than one 3-cell, then all 3-cells in H are R-connected, because c is not a face of any other higher-ranking 3-cells. Hence all 3-cells in H belong to the same R-connected component O_s for some node $s \in V$.
(2) Since $O_s \subseteq \Omega_s$ and the remaining cells $\Omega_s \setminus O_s$ are faces of (and hence connected to) some 3-cell in O_s, we only need to show that any two 3-cells in O_s are connected via a path of cells in Ω_s. Since any two 3-cells in O_s are connected via a path of R-connected 3-cells, we only need to show that two R-connected 3-cells $u, v \in O_s$ share a common face that is in Ω_s. By R-connectivity, u, v share a common face c that is not the face of any 3-cell with rank higher than that of u, v. Hence $c \in \Omega_s$, and the property holds.
(3) If s, t are connected, then there exists 3-cells $u \in O_s$ and $v \in O_t$ such that they share a common face c that is not a face of another 3-cell ranking higher than u or v. Hence c belongs to either Ω_s or Ω_t, implying that Ω_s and Ω_t are connected. Conversely, if Ω_s and Ω_t are connected, and without loss of generality, there exists a cell $c \in \Omega_s$ that is a face of a 3-cell $v \in O_t$. Let u be the 3-cell in O_s that c is a face of. Since $c \in \Omega_s$, there is no other 3-cell with rank higher than u that has c as a face. Hence u, v are R-connected, and hence nodes s, t are connected by a graph edge.

□

We now prove Proposition 5.2:

Proof. (**Proposition 5.2**) We first show that the following statement is true for any *proper* labelling L: if two nodes $s, t \in V$ are connected such that s has a higher rank than t, and if $L(t) = 1$, then $L(s) = 1$. Note that s can be either a kernel node or cut node. If s is a kernel node, then $L(s) = 1$ regardless of L. If s is a cut node, then t can only be a fill node, and therefore $L(s) = 1$ since L is proper.

To prove the Proposition, it suffices to show that the union of Ω_s over all 1-labelled nodes s is the modified shape X, or:

$$\cup_{L(s)=1} \Omega_s = \Omega(\cup_{L(s)=1} O_s) \tag{11}$$

for any proper labelling L. If this is true, then the three properties in Lemma A.1 ensure that the connected components of X, and of its complement $\Omega \setminus X$, have one-to-one correspondence with the connected components of the 1-labelled and 0-labelled subgraphs.

To show that equality 11 holds, we need to show that for any node $s \in V$ such that $L(s) = 1$, and any 3-cell $u \in O_s$, all faces of u (at all dimensions) must have been included in the union $\cup_{L(s)=1}\Omega_s$. We consider two cases for each face c of u. If $c \in \Omega_s$, then obviously $c \in \cup_{L(s)=1}\Omega_s$. Otherwise, by Lemma A.1, there exists some other node $t \in V$ such that $c \in \Omega_t$, t has higher rank than s, and s, t are connected. As shown at the beginning of this proof, we conclude that $L(t) = 1$, and hence $c \in \cup_{L(s)=1}\Omega_s$. □

B PROOF OF PROPOSITION 6.2

We start with a few lemmas before proving the main proposition. In the following discussion, we consider a graph $H = \{U, A, R, \omega\}$ as defined in Section 6.2. For any node $p \in U \setminus \{\pi\}$, we denote the node(s) in V that p represents by $V(p)$. For a set of nodes $P \subseteq U$, $V(P) = \cup_{p\in P} V(p)$. Conversely, we denote by $U(s)$ the node in U representing a node $s \in V$, whenever $U(s)$ exists and is unique. Given a subgraph S of H, we denote its non-terminal node set by U_S and the total weights of U_S as $\omega(S)$. We also denote the union of all cut and fill nodes in V as V_{CF}.

A subgraph S of H is called *C-Disjoint* (resp. *F-Disjoint*) if for any two nodes p, q in $U_C \cap U_S$ (resp. $U_F \cap U_S$), $V(p) \cap V(q) = \emptyset$. The following Lemma shows that any solution to NWST is both C-Disjoint and F-Disjoint:

Lemma B.1. *If S is a connected subgraph of H that spans R such that $\omega(S)$ is minimized, then S is both C-Disjoint and F-Disjoint.*

Proof. By symmetry, we only need to show that S is C-Disjoint. Suppose, to the contrary, that S is not, and there exist $p, q \in U_C$ such that $V(p) \cap V(q) \neq \emptyset$. By definition of cuttable sets, the union of two cuttable sets is also cuttable. Therefore there exists some node $r \in U_C$ such that $V(r) = V(p) \cup V(q)$. Furthermore,

$$\begin{aligned}&\omega(p) + \omega(q) - \omega(r)\\ &= \textstyle\sum_{s\in V(p)}(h(s,0) + D) + \sum_{s\in V(q)}(h(s,0) + D) - \sum_{s\in V(r)}(h(s,0) + D)\\ &= \textstyle\sum_{s\in V(p)\cap V(q)}(h(s,0) + D)\end{aligned} \tag{12}$$

On the other hand, by definition of D (Equation 10), the following holds for any node $t \in V$ and $\eta \in \{0, 1\}$:

$$h(t,\eta) + D > h(t,\eta) - \min_{s\in V_{CF}, \delta=0,1} h(s,\delta) \geq 0 \tag{13}$$

We conclude from Equations 12,13 that $\omega(p) + \omega(q) > \omega(r)$. By construction of graph edges A, any node in U connected to either p or q is also connected to r. We create another subgraph, S', by replacing p, q in S with r (if it is not already in S). The arguments above show that S' is connected and spanning R, and $\omega(S') < \omega(S)$, which contradicts the assumption that S is minimal. □

The next lemma shows that if a C-, F-, and CF-Disjoint subgraph exists in H, then the solution to NWST on H has to be CF-Disjoint.

Lemma B.2. *If there is a connected subgraph of H that spans R and is C-Disjoint, F-Disjoint, and CF-Disjoint, then any minimum-weight, R-spanning, connected subgraph of H must be CF-Disjoint.*

Proof. We first show that a connected, R-spanning, C-, F-, and CF-Disjoint subgraph S of H corresponds to a labelling of V. Since S is R-spanning and connected, every terminal node $r \in U_{CF}$ must be connected to some node in p in $U_C \cap U_S$ or $U_F \cap U_S$, implying that $V(r) \in V(p)$. Hence the union of sets $V(p)$ for all $p \in U_S$ covers all nodes in V. By C-, F-, and CF-Disjointness, no two sets in this union overlap. As a result, for any node $s \in V_{CF}$, its corresponding node $U(s) \in U_S$ exists and is unique. We define a labelling L such that s is labelled as 0 (resp. 1) if $U(s)$ is in U_C (resp. U_F). We can therefore express the weight of S as:

$$\omega(S) = \sum_{s \in V_{CF}} (h(s, L(s)) + D) \tag{14}$$

Suppose, to the contrary, that there exists some connected, R-spanning subgraph S' of H that has minimal weight, but S' is not CF-Disjoint. By the same argument above for S, the union of sets $V(p)$ for all $p \in U_{S'}$ still covers all nodes in V, but these sets are no longer disjoint (since S' is not CF-Disjoint). In particular, there exists some node $s_0 \in V_{CF}$, $p_0 \in U_C \cap U_{S'}$ and $q_0 \in U_F \cap U_{S'}$ such that $s_0 \in V(p_0) \cap V(q_0)$. Consider a labelling L' of V such that each node s is labelled as 0 (resp. 1) if there is some $p \in U_C \cap U_{S'}$ (resp. $p \in U_F \cap U_{S'}$) such that $s \in V(p)$. For nodes like s_0, we will arbitrarily pick a label. By inequality 13, we have:

$$\begin{aligned} \omega(S') &= \textstyle\sum_{p \in U_{S'}} \sum_{s \in V(p)} (h(s, \delta_p) + D) \\ &\geq \textstyle\sum_{s \in V_{CF}} (h(s, L'(s)) + D) + h(s_0, 1 - L'(s_0)) + D \end{aligned} \tag{15}$$

where δ_p is 0 (resp. 1) if $p \in U_C$ (resp. $p \in U_F$). The equality holds if s_0 is the only overlapping node among sets $V(p)$ for all $p \in U_{S'}$. Combining Equations 14,15, and by definition of D, we have:

$$\begin{aligned} &\omega(S') - \omega(S) \\ &\geq \textstyle\sum_{s \in V_{CF}} (h(s, L'(s)) - h(s, L(s))) + h(s_0, 1 - L'(s_0)) + D \\ &\geq -\textstyle\sum_{s \in V_{CF}} \|h(s, 0) - h(s, 1)\| + \min_{t \in V_{CF}, \delta = 0,1} h(t, \delta) + D \\ &> 0 \end{aligned}$$

which contradicts the assumption that S' has minimal weight. □

The next two lemmas establish the relation between a solution to the ACAP-TL problem and a connected, R-spanning subgraph of H that is C-, F- and CF-Disjoint.

Lemma B.3. *Let L be a proper and ACAP labelling on G. There is a connected, R-spanning subgraph S of H that is C-Disjoint, F-Disjoint, and CF-Disjoint. In addition, $\omega(S) = E^*_G(L) + D * |V_{CF}|$.*

Proof. Let $V_{L,0}$ (resp. $V_{L,1}$) denote the set of 0-labelled (resp. 1-labelled) nodes of V_{CF} under labelling L. Since L is proper, each connected component of $V_{L,0}$ (resp. $V_{L,1}$) is cuttable (resp. fillable). We define S to include all terminal nodes R, all $p \in U_C$ such that $V(p)$ is a connected component of $V_{L,0}$, and all $q \in U_F$ such that $V(q)$ is a connected component of $V_{L,1}$. By this construction, all sets $V(p)$ for $p \in U_S$ are disjoint, and hence S is C-, F-, and CF-Disjoint.

We next show that S is connected. First, by our construction of U_S, the union of sets $V(p)$ for all $p \in U_S$ covers all nodes in V. As a result, each terminal in U_{CF} is connected to some node in U_S. Next, consider a group of terminals $I \subseteq U_N$ such that $V(I)$ is a reachable set of neighborhood nodes in V. Since L is ACAP, $V(I)$ lies in a connected component of $G_{L,0}$, denoted by G_I. Let U_I be the set of nodes in $U_C \cap U_S$ such that $V(U_I)$ cover all the cut and fill nodes of G_I. Since $p \in I$ is connected to $q \in U_I$ if and only if $V(p) \in V(q)$, and since each node of U_I represents a connected component of the remainder of G_I after removing nodes $V(I)$, we conclude that terminal nodes I and nodes U_I are connected. Similarly, let $J \subseteq U_K$ be a group of terminals such that $V(J)$ is a reachable set of kernel nodes in V, this group is connected to nodes U_J in $U_F \cap U_S$ such that $V(U_J)$ cover all the cut and fill nodes of G_J, the connected component of $G_{L,1}$ containing $V(J)$. Furthermore, by definition of ACAP, each connected component of $G_{L,0}$ or $G_{G,1}$ must contain some neighborhood or kernel nodes, and hence any node of U_S must belong to U_I or U_J for some group I or J. That is, every node of U_S lies in some connected subgraph of S containing a group of terminals of U_N or U_K that represents a reachable set in V. Lastly, since each such group of terminals is connected to the terminal node π, we conclude that S is connected.

Finally, since the sets $V(p)$ for all $p \in U_S$ form a complete and non-overlapping cover of V_{CF}, we derive:

$$\begin{aligned} \omega(S) &= \textstyle\sum_{p \in U_S} \sum_{s \in V(p)} (h(s, \delta_p) + D) \\ &= \textstyle\sum_{s \in V_{CF}} (h(s, L(s)) + D) \\ &= E^*_G(L) + D * |V_{CF}| \end{aligned}$$

□

Lemma B.4. *Let S be a connected, R-spanning subgraph of H that is C-Disjoint, F-Disjoint, and CF-Disjoint. There is a proper and ACAP labelling L on G. In addition, $E^*_G(L) = \omega(S) - D * |V_{CF}|$.*

Proof. Using a similar argument as in the proof of Lemma B.2, the sets $V(p)$ for all $p \in U_S$ form a complete and non-overlapping cover of V_{CF}. For any $s \in V_{CF}$, let $U(s)$ denote the unique node $p \in U_S$ such that $s \in V(p)$. We define the labelling L so that $L(s)$ is 0 (resp. 1) if $U(s)$ is in either U_N (resp. U_K) or U_C (resp. U_F). Since each node of U_C (resp. U_F) represents a cuttable (resp. fillable) set in V_{CF}, and by definition of cuttability and fillability, L is proper.

We next show that L is also ACAP. We first show that any two reachable neighborhood (resp. kernel) nodes $s, t \in V$ must belong to a connected subgraph of $G_{L,0}$ (resp. $G_{L,1}$). Due to symmetry, we only discuss the case that s, t are reachable neighborhood nodes. Since S is connected, nodes $U(s)$ and $U(t)$ are connected by a simple path P in S. That is, P does not contain the same node twice. We will show that P only contains nodes from U_N and U_C. If this is true, $V(P)$ would form a connected subgraph in $G_{L,0}$ that contains s, t. First, P cannot contain any node from U_{CF}. This is because sets $V(p)$ for all $p \in U_S$ form a complete and non-overlapping cover of V_{CF}, and so each node of U_{CF} is connected to exactly one node of U_S and hence cannot be used in a path. Second, and because P avoids U_{CF}, it cannot contain any node of U_K or U_F without using the π node at least twice. Due to simplicity of P, it avoids U_K, U_F as well. Finally, we need to show that P does not contain π. Suppose to the contrary that it does, and π divides P into two segments P_1, P_2 such that P_1 connects $U(s)$ to some node $p \in U_N$, P_2 connects $U(t)$ to some other node $q \in U_N$, and π is connected to both p and q. Since both P_1 and P_2 contain only nodes from U_N and U_C, s and $V(p)$ are both contained in some connected subgraph in $G_{L,0}$, implying that they are reachable. Similarly, t and $V(q)$ are reachable as well. However, since π connects to only one node from each group of nodes of U_N representing a reachable set, $V(p)$ and $V(q)$ cannot be

reachable. This leads to the contradicting conclusion that s, t are not reachable.

To complete the argument that L is ACAP, we need to show that any 0-labelled (resp. 1-labelled) node $s \in V_{CF}$ is contained in a connected subgraph of $G_{L,0}$ (resp. $G_{L,1}$) that also contains some neighborhood (resp. kernel) node. We shall only discuss the case of a 0-labelled node s. Consider the node $U(s)$ in U_C. Note that $U(s)$ is only connected to some nodes from U_N or U_{CF}. Since S is connected, and since each node in U_{CF} is connected to only one node from U_S, $U(s)$ must be connected with some node $p \in U_N$. As a result, the connected subgraph of $G_{L,0}$ spanning $V(U(s))$ contains both s and neighborhood node $V(p)$.

Finally, by construction of L, we have:

$$\begin{aligned} E^*_G(L) &= \textstyle\sum_{s \in V_{CF}} h(s, L(s)) \\ &= \textstyle\sum_{p \in U_S} \sum_{s \in V(p)} h(s, \delta_p) \\ &= \omega(S) - D * |V_{CF}| \end{aligned}$$

□

We are now ready to prove Proposition 6.2:

Proof. (**Proposition 6.2**) Let S be a connected, R-spanning subgraph of H with minimal weight. We prove the two statements of the proposition.

(1) Suppose S is CF-Disjoint. By Lemma B.1, S is also C-Disjoint and F-Disjoint. Therefore, by Lemma B.4 and the arguments within its proof, the labelling L obtained by the rule in Proposition 6.2 (1) is a proper and ACAP labelling on G, and $E^*_G(L) = \omega(S) - D * |V_{CF}|$. It remains to show that L is minimal. Suppose, to the contrary, that there is some other labelling L' that is also proper and ACAP, such that $E^*_G(L') < E^*_G(L)$. By Lemma B.3, there is some connected, R-spanning subgraph S' of H such that $\omega(S') = E^*_G(L') + D * |V_{CF}|$. Therefore,

$$\begin{aligned} \omega(S) &= E^*_G(L) + D * |V_{CF}| \\ &> E^*_G(L') + D * |V_{CF}| \\ &= \omega(S') \end{aligned}$$

which contradicts the assumption that S is minimal.

(2) Suppose S is not CF-Disjoint. Suppose, to the contrary, there is some proper and ACAP labelling L on G. By Lemma B.3, there exists some subgraph S' of H that is connected, R-spanning, C-Disjoint, F-Disjoint, and CF-Disjoint. Since S is minimal, S must be CF-Disjoint by Lemma B.2, which reaches a contradiction.

□

C ACAP-TL AND NWST ON PRUNED GRAPH

To support the node-pruning strategy in Section 6.4, we show that the NWST solved on the pruned graph H still results in a proper and ACAP labelling (although it may not be optimal). Consider a subset of nodes $U'_C \subseteq U_C$ and $U'_F \subseteq U_F$, and the subgraph H' of H spanning nodes $U' = \{U_K, U_N, U_{CF}, U'_F, U'_C, \{\pi\}\}$. We show:

Proposition C.1. *Let S be a connected subgraph of H' that spans R and is CF-Disjoint, and define labelling L on G such that $L(s)$ is 1 (resp. 0) for any cut or fill node $s \in V$ if $s \in V(p)$ for some $p \in U'_F \cap U_S$ (resp. $p \in U'_C \cap U_S$). Then L is proper and ACAP.*

Proof. Since S is connected, R-spanning, and CF-Disjoint, the same argument in the proof of Lemma B.2 shows that L as defined in Proposition C.1 gives each node $s \in V_{CF}$ a unique label. Note that, since S may not be C-Disjoint or F-Disjoint, there could be multiple nodes of U'_C (resp. U'_F) representing the same node in V_{CF}. Nonetheless, CF-Disjointness ensures that the same node will not be given *different* labels. Since each node in U'_C (resp. U'_F) represents a cuttable (resp. fillable) set of V_{CF}, L is proper.

To show that L is ACAP, we closely follow the arguments in the proof of Lemma B.4. The key difference is that, since S may not be C-Disjoint or F-Disjoint, a node in U_{CF} may connect to multiple nodes of U_S, although these nodes are either all in U'_C or all in U'_F (due to CF-Disjointness of S). We make the following changes to the proof of Lemma B.4 to reflect this difference:

- In the argument for why two reachable neighborhood nodes $s, t \in V$ belong to a connected subgraph of $G_{L,0}$, we make the following change. We will show that the simple path P connecting $U(s), U(t)$ contains only nodes from U_N, U_C, U_{CF} (instead of just U_N and U_C). If this is true, $V(P)$ forms the desired subgraph of $G_{L,0}$. To show that P avoids U_K and U_F, note that any node of U_{CF} cannot serve as a "bridge" from $U_N \cup U_C$ to $U_K \cup U_F$, and hence π will still be present in P (at least twice) if P contains a node from U_K or U_F.
- We use a different argument to show that a 0-labelled node $s \in V_{CF}$ is contained in a connected subgraph of $G_{L,0}$ that also contains a neighborhood node. Consider any node $p \in U'_C \cap U_S$ such that $s \in V(p)$. Since S is connected, there is some terminal node $q \in U_K$ that is connected to p via a simple path P in S, such that P does not contain any other terminal node in U_K than q. Since nodes in U'_C are only connected to terminals in U_K and U_{CF}, and the latter are only connected to U'_C, P consists solely of p, q, and nodes of U'_C and U_{CF}. Thus $V(P)$ forms the desired subgraph of $G_{L,0}$.

□