The topological basis of function in flow networks

Jason W. Rocks, 1 Andrea J. Liu, 1 and Eleni Katifori 1

1 Department of Physics and Astronomy, University of Pennsylvania, Philadelphia, PA 19104, USA

The ability to reroute and control flow is vital to the function of many biological transport networks. By tuning the conductance of edges, many flow networks robustly control the propagation of inputs in order to achieve a wide variety of specific tasks. However, recent evidence suggests that network architectures used to achieve such tasks demonstrate significant design flexibility, blurring the relationship between structure and function. Here we seek to identify the structural features responsible for function in tuned networks. By adding and removing links, we optimize flow networks to perform a specific function, namely to achieve a minimum pressure difference along a chosen “target” edge when a pressure difference is applied to another predetermined “source” edge in the network. Using persistent homology, we show that networks tuned to perform such functions develop characteristic large-scale topological features that are similar for different networks that perform the same function, regardless of differences in the local link connectivity. These features correlate strongly with the tuned response, providing a clear topological relationship between structure and function.

I. INTRODUCTION

Many biological fluid transport networks are optimized to redirect flow as dictated by the needs of the system. By locally tuning the conductances of links (edges) of the network, venation networks in animals [1–3], plants [4, 5], fungi [6], and slime molds [7] control the spatial distribution of water, nutrients, oxygen, or metabolic byproducts. Flow networks have been shown to be remarkably tunable with the ability to support highly complex, multifunctional tasks. By adding or removing a small number of edges from a network, the pressure differences of a large number of edges can be tuned simultaneously to respond in a predetermined way to a pressure difference applied elsewhere in the network [8]. However, a basic question remains unresolved: What is the means by which a flow network performs a given task?

For simplicity, we focus on networks that we tune to perform a task composed of the response of a single “target” edge. By modifying the edge conductances of a network, we tune the pressure difference of this edge to respond by at least an amount $\Delta$ (chosen to be non-negative) when a unit pressure difference is applied to a separate, predetermined “source” edge. If two different networks are tuned to respond with the same value of $\Delta$, we say that they perform the same function. With this definition, the high degree of task complexity attainable in these networks (as demonstrated by Ref. [8]) implies that different networks with different choices of source and target edges can nevertheless be tuned to achieve the same $\Delta$. In other words, different structures can perform the same function. This raises the question of whether different networks that perform the same function share common aspects that depend on structure but transcend some of its details.

Here we use persistent homology to identify the underlying basis of function in a class of flow networks. We find that the structure-function relationship is topological in nature: as a network is tuned to achieve a target pressure difference $\Delta$, it separates into two domains of relatively uniform pressure. While this description is simple, we demonstrate it is robust even when $\Delta$ is small and the two domains cannot be identified by eye. The general approach we develop here could serve as the basis for studying the structure-function relationship for a wide variety of complex networks.

II. NETWORK TUNING PROTOCOL

We derive our flow networks from randomly-generated two-dimensional configurations of soft spheres with periodic boundary conditions, created using standard jamming algorithms. Flow networks are extracted from these configurations by placing nodes at the centers of each sphere and edges – with associated fluid conductances – between nodes corresponding to spheres that overlap. For each network, the values of the edge conductances are chosen randomly from the range 0.01 to 1.0 in discrete increments of 0.01. The source edge and target edge are each chosen at random in the network. To achieve a desired target pressure difference of at least $\Delta$ across the target edge, we use a greedy algorithm: in each step we increase or decrease the conductance of a single edge by 0.01 (staying within the range 0 to 1, inclusively), modifying the edge conductance that best optimizes the response at that step (for further details, see the Appendix, along with Ref. [8] and similar work on mechanical networks in Ref. [9]).

Fig. 1 shows examples of two different networks that have been tuned to perform the same function, namely to have a target pressure difference of $\Delta = 0.2$ relative to the source. For the purpose of comparison, the target edges have been chosen to be separated by similar distances from the source in each network. Figs. 1(A1) and (B1) depict the pressures on the nodes and the conductances of the edges, while Figs. 1(A2) and (B2) show the corresponding pressure differences on the edges. Clearly, the spatial distributions of edge conductances and pressures in the networks are noticeably different; it is unclear...
FIG. 1. Comparison of two flow networks with different structures tuned to perform the same function composed of a single target edge with a pressure difference of $\Delta = 0.2$. The source nodes are shown in red, while the target nodes are shown in green. (A1) and (B1): The pressures on the nodes are shown in black where the symbol denotes the sign of the pressure and the size denotes the magnitude. The thickness of the edges corresponds to the conductance. Edges that are shown as thick dashed blue lines have been fully removed (set to zero conductance) in the process of tuning. (A2) and (B2): The absolute values of the pressure differences on the edges of the tuned networks in (A1) and (B1) shown in shades of blue.

whether the two networks share anything in common beyond the pressure differences at their target edges, which have been tuned to the same value.

III. MAXIMUM TUNING LIMIT ($\Delta = 1$)

It is illuminating to first examine the extreme limit where $\Delta = 1$, the maximum achievable pressure difference at the target edge (see Figs. 2(A1) and (A2)). Here the network clearly separates into two domains of uniform node pressure, connected only by a single edge at the source. These two regions are separated by a crack-like structure with pressure differences of precisely 1.0 along edges that have been removed during the tuning process. The structural changes in the network architecture are purely topological: all edges connecting the two domains are removed (excluding the source edge, which could be removed with no change in the response), increasing the number of connected components in the network from one to two. The exact details of the local structure (which specific edges are modified) do not matter as long as this partitioning takes place. In this extreme case, the relationship between structure and function is clear: the presence of the two separate connected components, each associated with one source node and one target node, allows the desired target pressure difference to be achieved. Intuitively, this description should extend to all networks tuned to this same extreme limit, since adding any extra links between the two domains would allow current to flow between them and necessarily decrease the pressure difference. However, when $\Delta$ is less than 1, as in Fig. 1(B1) and (B2), the entire network is highly connected and the important aspects of the structure are not as well-defined.

Here we show that a persistent homology analysis allows us to generalize the domain description observed
in the extreme $\Delta = 1.0$ case to smaller pressure differences. The analysis is able to identify two analogous components, even in cases such as Fig. 1(B1) where the target pressure difference of $\Delta = 0.05$ is too small for the components to be discerned by eye. Consequently, this process allows us to relate the structures of different networks tuned to perform the same function.

**IV. TOPOLOGICAL SIGNATURE OF TUNING**

At its core, the process of tuning networks is local; it involves modifying the conductances of individual edges. However, the extreme example of Fig. 2(A1) provides evidence that coordinated, large-scale topological changes in the structure and response can arise from local edge tuning. To see if remnants of these topological changes are responsible for function when $\Delta < 1$, we use persistent homology, a technique that can detect and assign significance to the topological features of geometrically and/or topologically structured data [10, 11]. In this case, our data consists of the pressure response of tuned networks, along with the connectivities of the nodes and edges. In general, the types of topological features the persistence algorithm can detect include connected components, loops, voids, etc; for flow networks, only the first two are relevant. Since the extreme case for $\Delta = 1$ suggests that the network partitions into different domains, we focus solely on connected components. In the past, the persistence algorithm (or related techniques) has been used to study various topological aspects of flow networks [12, 13], along with their higher-dimensional analogs, mechanical networks [14, 15].

To apply the persistence algorithm, one needs an ordering of the network elements (vertices, edges) in terms of a quantity defined on the particular elements that are relevant to the tuned function. An obvious candidate is the pressures on the nodes. However, since the node pressures obey a discrete version of Laplace’s equation, local minima and maxima in the node pressures can only occur at the source. As a result, there can only be a single (global) minimum on one of the source nodes, and a single (global) maximum at the other source node. Since local extrema play an important role in defining topological features, their absence means that very few interesting features would be detected by the persistence algorithm (in fact, we would only detect a single connected component corresponding to the two global extrema at the source nodes). We therefore define our ordering on the nodes instead of the edges, sorting each edge according to the absolute value of the difference in pressure between its nodes. Given a network with $N_E$ edges, we label each edge with an integer $i$ according to this order, with $1 \leq i \leq N_E$, as illustrated in Fig. 3(A), and denote its corresponding pressure difference as $\Delta p_i$.

We then proceed as follows: starting with an empty network with no edges, we add each edge to the network in order of its pressure difference, one at a time. With each step $i$, we obtain a larger subset of our original network, consisting of the first $i$ edges. This sequence

![Figure 3](image-url)
of subnetworks corresponds to a filtration of the pressure differences on our original network. In the “ascending filtration,” we perform this process for each edge in order of the absolute value of the pressure differences from smallest to largest. Similarly, for the “descending filtration” we proceed in order of decreasing pressure difference.

At each step in the filtration, the persistence algorithm records any changes in the topological structure, i.e. any changes in the number of connected components. When an edge is added, there are three possibilities: (i) the new edge is not connected to any of the pre-existing edges, increasing the number of connected components by one; (ii) the new edge is shared between two of the pre-existing components, joining them together and decreasing the number of connected components by one, or (iii) the new edge is only connected to a single pre-existing component, incurring no change in the number of connected components. For the first case, in which a new component appears, we say that it is “born” and record the pressure difference at that step, \( \Delta p_b \), as its “birth pressure difference.” The new edge is the “birth edge.” In the second case, in which two components merge, we say that the component in the pair that was born most recently has “died,” and we record the pressure difference, \( \Delta p_d \), as its “death pressure difference.” The new edge is the “death edge.” In this way, each connected component that appears during the filtration is assigned a birth-death pair \((\Delta p_b, \Delta p_d)\). By carrying out the filtration in both ascending and descending order, we collect two sets of birth-death pairs, one for each filtration (the approach we have described here has been simplified for the sake of discussion, but is a sufficient version of the persistence algorithm. See the Appendix and Ref. [10] for a detailed explanation of the complete algorithm).

Figs. 3(B)-(F) illustrate this process for an example network. New components are born in Figs. 3(B), (C), and (D), colored green, orange, and blue, respectively, with corresponding birth pressures of \( \Delta p_1 \), \( \Delta p_4 \), and \( \Delta p_5 \). Figs. 3(E) and (F) show the deaths of two of the components. In Fig. 3(E), the blue component dies with a death pressure of \( \Delta p_7 \), resulting in the birth-death pair \((\Delta p_5, \Delta p_7)\), while in Fig. 3(F), the orange component dies with death pressure \( \Delta p_{16} \), resulting in the birth-death pair \((\Delta p_4, \Delta p_{16})\). The final component, consisting of the entire network, never dies, so we do not assign it a birth-death pair.

Once we have collected all birth-death pairs, \((\Delta p_b, \Delta p_d)\), we can construct a persistence diagram as depicted in Fig. 3(G). For the ascending filtration, the death pressure difference exceeds the birth pressure difference in each pair; these pairs are represented by points colored in blue. For the descending filtration, the death pressure difference is always smaller than the birth pressure difference in each pair; these pairs are represented by points colored in red. The complete set of points characterizes the topological structure of connected components in the network. Points associated with the ascending/descending filtration represent regions of the network with relatively low/high pressure differences.

Additionally, the vertical distance of a point from the black diagonal line in Fig 3(G), along which \( \Delta p_b = \Delta p_d \), is called the “persistence”: \( \tau = |\Delta p_d - \Delta p_b| \). This measures the lifetime of a feature during the filtration process, and provides a measure of its significance. Small fluctuations in pressure differences, for example, would yield birth-death pairs with low persistence. In the ex-
ample of Fig. 3, we see that the point \((\Delta p_4, \Delta p_{16})\) has a large persistence value. This means that the corresponding orange connected component survives, or persists, for a large range of pressure differences during the persistence algorithm. This high persistence suggests that this feature is important for characterizing the structure of the network.

We have carried out the persistence analysis for ensembles of tuned and untuned networks and collected the results for each ensemble into a persistence diagram. Fig. 4 (A) depicts a two-dimensional histogram of the average persistence diagram of over 30000 untuned networks, each composed of 256 nodes. For each network, the source and target edges are selected randomly. The histogram is calculated by dividing the persistence diagram into bins in \(\Delta p_b\) and \(\Delta p_d\) (shown as individual pixels) and counting the average number of points (birth-death pairs) that fall within each bin across all of the networks in the ensemble. We observe two different clusters of features for untuned networks, both of which correspond to fluctuations in the response due to the discrete nature of the initial networks. The features clustered near the origin are typically located far from the target edge where the pressure difference scale is relatively low. The band of features below the diagonal at birth pressure differences between about \(\Delta p_b = 0.35\) and 0.6 typically correspond to small numbers of isolated edges of relatively high pressure differences located near the source. In the continuum limit of Laplace’s equation with infinite system size, both sets of features would collapse towards a single point at the origin.

Fig. 4 (B) shows the equivalent histogram for an ensemble of networks tuned to a target pressure of \(\Delta = 0.8\). A comparison of Figs. 4 (A) and (B) shows that the histogram of the persistence diagrams changes drastically in two ways. First, a high concentration of features appears in the ascending diagram, located above the diagonal, concentrated in a thin vertical band at a birth pressure of \(\Delta p_b = 0\), with death pressures ranging from zero to our tuned response of \(\Delta = 0.8\). This indicates that tuned networks tend to develop regions of almost perfectly uniform node pressure (zero pressure difference), separated by boundaries of high pressure differences up to the tuned pressure difference. Most of these features are located far above the diagonal, indicating that they are of high significance. Similarly, for the descending diagram, a vertical band appears for the tuned networks that is absent for untuned networks. This band is concentrated at a birth pressure equal to our tuned response \(\Delta = 0.8\) with a death pressure ranging from zero to 0.8. This band corroborates our observations of the ascending filtration; it indicates that there are regions of pressure differences equal to our tuned response. These likely correspond to the boundaries between regions of uniform node pressures. Again, many of these features are of high significance because they are located far below the diagonal.

To understand how persistence diagrams evolve in more detail, we calculate the average persistence diagram for 11 target pressures ranging from \(\Delta = 0.0\) to 1.0. For each bin we find the value of \(\Delta\) whose average persistence diagram is most highly represented, with the largest average number of points in that bin compared to all \(\Delta\). We color each bin according to this representative value of \(\Delta\) as shown in Fig. 4 (C). We see that as networks are tuned for larger and larger target pressures \(\Delta\), the average ascending persistence diagram is steadily populated with points far above the diagonal and in a band \(\Delta p_d = 0\) to \(\Delta\), while the average descending diagram develops features at the tuned target pressure, in bands located at \(\Delta p_b = \Delta\). This confirms that the trends we see in Figs. 4 (A1) to (A2) generalize to all values of \(\Delta\).

V. TOPOLOGICAL CHARACTERIZATION

Now that we have identified persistent features that appear in the tuned network structures, namely the features in the vertical bands that appear at \(\Delta p_b = 0\) in the ascending filtration and at \(\Delta p_b = \Delta\) in the descending filtration, we associate these features with their actual components in the tuned networks. The obvious approach would be simply to identify the connected components that define each point in the persistence diagrams at either their birth or directly before their death as shown in Fig. 3. However, components can merge multiple times, forming a binary tree of component mergers. This results in identified regions that overlap with one another, with each node belonging to a large set of different com-

FIG. 5. (A) Skeletonized tree of the tuned network structure shown as thick lines, with death edges shown as thick dashed lines. The filtration index of birth and death edges are shown with circular and rectangular backgrounds, respectively. Using the death edges as the boundaries of components, three components shown in green, orange and blue can be identified. Each component (except the green component) has a corresponding persistence pair found by examining the index of that component’s birth edge. (B) Simplification of the topological structure by eliminating the component of lowest persistence, the blue component with associated persistence pair \((\Delta p_1, \Delta p_d)\). It is joined to the orange component because its death edge with index 7 connects the two components.
To accomplish this, we start by creating a skeletonized tree representation of our network, as shown in Fig. 5(A), which both encodes the topological changes we saw in our persistence algorithm and also allows us to uniquely divide our network in distinct components. To create this tree, we first perform the ascending filtration we defined in the previous section, keeping any edge which fits at least one of the following criteria: (i) the edge creates a new connected component (a birth edge), (ii) the edge merges two connected components (a death edge), or (iii) the edge adds a new vertex to the network.

Next, we mark any edges that fit the second criterion with a dashed line. As these edges denote merging events in our filtration, they naturally separate our network into different components. Using these edges as the boundaries between regions in the skeletonized network, we partition the network into different connected components, shown as the green, orange and blue regions in Fig. 5(A). Each component we find - except the first one to appear in the filtration - is associated with one of the birth-death pairs we found using the persistence algorithm. The birth edge is always the edge with the minimum filtration index in a component, while the death edge is always a dashed boundary edge that may or may not border the component. In the example, the orange and blue components are associated with the pairs \((\Delta p_4, \Delta p_6)\) and \((\Delta p_5, \Delta p_7)\), respectively. This process is analogous to the watershed transform often used in image segmentation [16]: each component we find is effectively a basin associated with a local minimum that corresponds to the component’s birth edge. The main drawback to this method is that it results in rampant over-segmentation, as is often the case when performing watershed transforms on noisy data. Each of the large number of points in our persistence diagram results in a new segment, no matter how small its persistence value. Figs. 1(A1) and (B1) show all of the individual components corresponding to birth-death pairs for the two networks from Fig. 1. Each component is colored arbitrarily in order to highlight individual regions. Clearly, the two networks are highly segmented, and the many individual connected components do not provide much structural intuition.

To remedy this, we use persistence-based simplification to effectively coarse-grain the components of our network [17, 18]. The goal is to eliminate components whose associated birth-death pairs have small persistence values \(\tau\), or low significance. To eliminate a component, we first find the birth-death pair with the smallest persistence value and its associated component. Next, we determine if the death edge is located on the boundary of that component, meaning it has one vertex located within the component. If it is does not, then we skip this component and find the birth-death pair with the next smallest persistence value, repeating this process until we find a component that satisfies our criterion. For example, in Fig. 5(A) the blue component with birth-death pair \((\Delta p_5, \Delta p_7)\) has the smallest persistence value, given by the point’s distance from the diagonal in Fig. 3(G). Once we have found a satisfactory component, we eliminate it by joining it with its unique neighboring component connected through its death edge, as seen in Fig. 5(B). We also stop marking the death edge with a dashed line, as it no longer separates two components. We can then perform this entire process repeatedly, eliminating the least significant feature at each iteration and thus simplifying the topological structure of our network (Again, for the sake of discussion, here we have presented an abridged version of the simplification algorithm. For a detailed description please see the Appendix and Refs. [17] and [18]).

We must still decide when to stop eliminating components. Instead of imposing a threshold by choosing a specific value for the persistence up to which we wish to simplify, we apply a physical criterion to choose a natural

\[
\Delta = 0
\]

...
FIG. 7. Evolution of the network structure and response with corresponding domain partitioning for (A) an untuned network and the same network tuned for target pressure differences of (B) $\Delta = 0.05$ (C) $\Delta = 0.20$, and (D) $\Delta = 1.00$. (First Row) Edges are colored on a white-to-blue scale according to the absolute value of the pressure differences on the edges. The source nodes are shown in red and the target nodes in green. The simplified domains containing each of the target nodes are highlighted in green and orange. (Second Row) The corresponding network structure and node pressures. The pressures on the nodes are shown in black where the symbol denotes the sign of the pressure and the size denotes the magnitude. The thickness of the edges corresponds to the conductance. Edges that are shown as thick dashed blue lines have been removed in the process of tuning. (Third Row) The associated histogram of node pressures with green and orange histograms showing the contributions of nodes in the green and orange domains, respectively, shown in the first and second rows.

stopping point. Since we have tuned the network to have a certain pressure differential between the target nodes, we continue to simplify components until the two target nodes join the same segment. At this point, we back off one step in the iteration to identify the two components that joined together. In this way we are able to uniquely identify significant regions of our tuned network structures associated with each target node without choosing an arbitrary simplification cutoff. Furthermore, we can measure the value of the persistence at which this simplification process stops, $\tau_{\text{max}}$, giving us a quantitative measure of the validity of our assumption that the two components corresponding to each target node are separate pressure domains. If $\tau_{\text{max}}$ is measured to be zero, then it is not possible to separate the network into two components in this way. But if $\tau_{\text{max}}$ is greater than zero, it measures the range of pressure differences over which the less persistent of the two components spans.

Figs. 6(A2) and (B2) demonstrate the results of the simplification procedure for the networks shown in Fig. 1. After coarse-graining via persistence, the topological structure of both networks has been greatly simplified compared to the initial components shown in Figs. 6(A1) and (B1), allowing us to identify two main regions (shown as green and orange), each associated with a single target node. These regions allow us to directly compare the structures of the two networks that have been tuned for the same function.
VI. STRUCTURE-FUNCTION RELATIONSHIP

Fig. 7 shows how the two components containing the two target nodes evolve with tuning. In Figs. 7(A1) and (A2), we see that before tuning, the two components associated with the target nodes (highlighted as green and orange) do not have any obvious structural significance, with a simplification threshold of \( \tau_{\text{max}} = 0.002 \). Fig. 7(A3) shows a histogram of the pressures on the nodes, highlighting the regions of the histogram associated with the two components. If we calculate the mean node pressure for the nodes in the green region and separately for the nodes in the orange region, we can take the absolute value of the difference between the two mean pressures, \( \Delta \bar{p} \), as an effective pressure difference between the two regions. We find that the untuned network has a very small effective pressure difference of \( \Delta \bar{p} = 0.002 \).

After tuning to a target pressure of \( \Delta = 0.05 \), Figs. 7(B1) and (B2) show that the network has more clearly segregated into two components bounded by a crack-like region of slightly larger pressure differences. These pressure differences are a result of edges that have been completely removed in that region. Examining the histogram in Fig. 7(B3), we see that the pressures of the nodes in the two components fall into different regions of the histogram with a difference in mean node pressures of \( \Delta \bar{p} = 0.06 \), close to the tuned pressure difference. The simplification threshold is \( \tau_{\text{max}} = 0.008 \).

Further tuning to a target pressure of \( \Delta = 0.2 \), we see in Figs. 7(C1) and (C2) that the two components partition the network even more clearly, and the areas of the histogram in Fig. 7(C3) associated with each component now comprise separate peaks. The nodes are almost completely partitioned into the two regions with node pressures of different signs. The difference in mean node pressures of the two regions is \( \Delta \bar{p} = 0.21 \), closely tracking the tuned pressure difference, and the simplification threshold is \( \tau_{\text{max}} = 0.04 \).

Finally, Figs. 7(D1) and (D2) show a complete partitioning of the network according to node pressure at a tuned pressure difference of \( \Delta = 1.0 \). The histogram in Fig. 7(D3) confirms this, as it shows two narrow peaks of node pressures with \( \Delta \bar{p} = 1.0 \) and a maximum possible simplification threshold of \( \tau_{\text{max}} = 1.0 \). In summary, Fig. 7 shows that as the target edge is tuned to larger and larger pressure differences, the network steadily partitions into two distinct domains. The node pressures within each domain are relatively uniform and the difference between the mean pressures in the two domains corresponds closely to the tuned target pressure difference. The separation of the two regions, as measured by the persistence simplification threshold \( \tau_{\text{max}} \), increases as well, to a maximum value of \( \tau_{\text{max}} = 1.0 \) in the limit \( \Delta = 1.0 \).

We have established the generality of this observation by tuning many networks to a variety of target pressure differences, obtaining the two components associated with the target nodes in each case. For each network, we calculate the difference in mean node pressures between the two regions \( \Delta \bar{p} \). Fig. 8(A) shows the average of \( \Delta \bar{p} \) for various system sizes \( N \) and target pressure differences \( \Delta \). Each point is calculated by averaging the results over an ensemble of 512 independent networks. We see that \( \Delta \bar{p} \) is close to \( \Delta \) for all system sizes and target pressure differences, with almost perfect agreement for larger systems. We also measure the average standard deviation \( \sigma_{\Delta \bar{p}} \) of the measured pressure difference for each system size and tuning threshold in Fig. 8(B). We see that the spread of each distribution is very narrow, especially for large \( N \).

In Fig. 9(A), we show that the average total fractional coverage of the two domains (the sum of the number of nodes in the two domains normalized by \( N \)) quickly approaches unity with increasing \( \Delta \), showing that the entire network is described by the domain structure, not just a small region near the target. Again, the results converge more rapidly for larger systems. Fig. 9(B) shows the average two-sample Kolmogorov-Smirnov (K-S) statistic between the node pressure distribution of the two domains (the green and orange histograms in Fig. 7) for each network. This quantifies the difference between the contributions to the total node pressure histograms of the two domains, with a value of one indicating no overlap between the two domains (as in Fig. 7(D3)) and zero indicating that the two domains overlap significantly (as in Fig. 7(B3)). We see that the K-S statistic quickly approaches unity with increasing \( \Delta \), especially for larger \( N \), indicating that the two domains rapidly segregate into regions with non-overlapping distributions of node...
pressures. Finally, in Fig. 9(C), we show the average simplification threshold \( \tau_{max} \), which indicates the significance of the topological separation of the two domains. We see that it starts at zero for low \( \Delta \) and increases to a maximum value of one for \( \Delta = 1.0 \). It is notable that even when \( \tau_{max} \) is small, the two domains still predict the tuned pressure difference quite accurately.

VII. DISCUSSION

In summary, we have established a quantitative characterization of function in flow networks using persistent homology. This analysis reveals the topological means by which function is tuned into these networks, providing a clear relationship between structure and function. As a network is tuned to larger and larger pressure differences at the target, it segregates into two domains of relatively uniform pressure which characterize and predict the tuned response. Although the local node connectivity and geometrical structure can differ between two networks tuned for the same function, the commonality in structure of the networks becomes apparent when viewed through a topological lens. This leads us to propose a refinement of the structure-function paradigm in the context of functional flow networks. Since the process of tuning is inherently topological, the aspect of structure that relates to function is also topological; it is the relationship between the topological structure and function that is important, not the relation between geometric structure and function. The fact that the structure-function relationship is topological contributes to the robustness of our results even in the case of small \( \Delta \), when the structures relevant to the tuned function are not discernible by eye.

We have also demonstrated that the techniques provided by persistent homology - both the persistence algorithm and persistence-based simplification - are powerful tools for quantifying network structures in a unique and threshold-independent manner. The persistence algorithm allows us to identify the general physics that distinguishes between systems (e.g. tuned and untuned networks), but is unable to uncover the precise features responsible. The persistence-based simplification allows the topological differences revealed by the persistence algorithm to be translated into concrete and unique features (in this case two connected components), even in the case of noisy data.

We note that our analysis differs from the minimum cut algorithm [19]. In the limit of high \( \Delta \), each of the two network components can be associated with one target node and one source node. In the context of network flow optimization (where flows are more broadly construed to allow for sub-maximal edge currents and unidirectional edge current constraints), an \( s \rightarrow t \) cut is a set of edges that when removed partitions the nodes of a flow network into two components, one containing the source node \( s \) (positive node pressure) and the other the sink node \( t \) (negative node pressure). The max-flow min-cut theorem states that the maximum possible value of the flow from a source node to a sink node is given by the total sum of the edge weights (conductances) defining the minimum cut, the \( s \rightarrow t \) cut with the minimum possible sum of edge weights. Various algorithms utilize this theorem to calculate maximum flows and, by extension, minimum cuts [19]. For large \( \Delta \), we expect that the two components obtained using persistence-based simplification should coincide with those that could be found by applying a minimum cut algorithm. However, for small \( \Delta \), the components we find do not encompass the entire network, differing from a minimum \( s \rightarrow t \) cut whose components should, by definition, always cover the entire network. Developing a formal connection between these two methods could provide further insight into the physical interpretation of the components we detect, along with
a deeper understanding of the topological properties of more general transport networks.

Our analysis also differs from the modularity analysis often applied to complex networks [20]. A modularity analysis requires a null model for the network connectivity to determine whether statistically significant communities exist and where they are [21]. The persistent homology analysis presented here does not require a null model. It is an interesting question for future work to assess the similarities and differences between the two analyses for different types of networks.

In our tuned flow networks, crack-like structures formed by edge removals partition the network into two components. Crack-like defects in resistor networks have been studied in some detail in the random resistor network literature, but not in the context of tuning [22]. An analytical theory of tuning would likely require providing a relationship between these crack structures, the segregation of the network into components, and the tuned response. This work has provided an important step towards relating the latter two, but has not explicitly explored the role of cracks.

It may be possible to ascertain what function has been tuned into a network, along with the relevant features, even if the location of the target is not known. Using the persistence algorithm, it is possible to calculate the distribution of all persistence values $\tau$ in a network. Since the relevant features typically have high values of $\tau$, as shown in Fig. 4, they will often show up as outliers in this distribution. By examining the properties of these outliers and looking at where features concentrate in the persistence diagrams, an educated guess could be made as to what function has been tuned into a network and perhaps even the location of the target nodes.

The analysis introduced here is general and could also be used to study the relationship between structure and function of flow networks tuned to perform other types of tasks such as displaying a specific current response or power loss through a target edge, or networks optimized to minimize global power loss, etc. Our analysis can be extended easily to networks tuned to perform more complex tasks composed of multiple targets such as those in Ref. [8]. Since these techniques do not depend on the local node connectivity, we would also expect our results to be robust to the overall network topology before tuning (e.g. non-planar, non-local edge structures or network with high degrees of modularity). As long as has a function has been successfully tuned into a network, we would expect qualitatively similar results. In addition, in this work we only explored the nature of connected components (0-cycles), but the persistence algorithm can also be used to identify significant cycles of edges (1-cycles) as well. Repeating our analysis to quantify the 1-cycle topology may prove useful in understanding the effects of the untuned network properties on what types of functions a network can be tuned to perform.

Biological flow networks employ a variety of mechanisms over a wide range of time scales in order to regulate local flow. On relatively short time scales, the vasculature systems of animals – notably that of the brain – and slime molds can dynamically control local flow by constricting and dilating vessels in order to support local activity. On longer time scales, animals, fungi, and slime molds can control flow to restructure the vasculature network. All these systems also undergo evolution on generational time scales to modify their network designs depending on the needs of the system or environmental changes. In all cases, our results suggest there may be a topological basis for function that could be uncovered by applying the analysis introduced here.

Obtaining an accurate and complete map of every single vessel of an entire organ or organism poses a difficult experimental challenge, as vasculature networks frequently consist of millions of nodes and span several length scales. In addition, it is known that small errors in the connectivity or conductances can be disastrous in determining function [23]. In spite of these obstacles, experimental researchers have directed their efforts to fully characterizing node connectivity and edge conductances (vessel diameters) [24]. Our results show that such detailed knowledge is not necessary. Rather, one must measure node pressures because function is encoded robustly in regions of nearly-uniform node pressure. Now that we understand this, such regions should not be difficult to identify, even with only partial measurement of node pressures. We hope that our results will inspire experimentalists to characterize network structures using a topologically-informed approach.

Given that flow networks are mathematically equivalent to one-dimensional mechanical networks [8], our results suggest that one could ask whether the structure-function relationship is also topological in mechanical networks that can perform mechanical functions, such as motor proteins or allosteric proteins. Mechanical networks can also be tuned to perform specific functions [9, 25, 26] and can undergo topological changes in structure, displaying responses ranging from hinge-like motions [27, 28] to more exotic “trumpet”-like responses [25]. The extreme case of a flow network segregated into two components with $\Delta = 1$ is analogous to the notion of a mechanical mechanism, or soft mode, since the response does not require any expenditure of energy (or power in the case of flow networks). The role of soft modes in function has been studied in proteins [cite Thorpe, protein flexibility predictions from graph theory]. Our analysis of flow networks provides a generalization of this idea to the case where the components are still connected with $\Delta < 1$. A similar analysis is therefore likely to be useful in identifying the generalization of a mechanical mechanism to the case where the deformation involved in the function is not a soft mode.

More generally, a topological description of function may help elucidate significant structures in other types of non-conservative or nonlinear functional networks. For example, one could ask whether characteristic topological features correspond to memories encoded in neural
networks trained via machine learning or in neuron networks.

ACKNOWLEDGEMENTS

We thank R. D. Kamien and S. R. Nagel for instructive discussions. This research was supported by the NSF through DMR-1506625 (J.W.R.) and PHY-1554887 (E.K.), the Simons Foundation through 454945 (J.W.R. and A.J.L.), 327939 (A.J.L.), and 568888 (E.K.), and the Burroughs Welcome Career Award (E.K.).

[1] Ronald F Tuma, Walter N Durán, and Klaus Ley. Handbook of Physiology: Microcirculation. Academic Press, San Diego, 2008.
[2] Marilyn J Cipolla. The Cerebral Circulation. Morgan & Claypool Publishers, second edition, 2016.
[3] Yu Rong Gao, Stephanie E. Greene, and Patrick J. Drew. Mechanical restriction of intracortical vessel dilation by brain tissue sculpts the hemodynamic response. Neuroimage, 115:162–176, jul 2015.
[4] J. Pittermann. The evolution of water transport in plants: An integrated approach, mar 2010.
[5] Lawren Sack and Christine Scifoni. Leaf venation: Structure, function, development, evolution, ecology and applications in the past, present and future. New Phytol, 198(4):983–1000, jun 2013.
[6] Luke Heaton, Boguslaw Obara, Vincente Grau, Nick Jones, Toshiyuki Nakagaki, Lynne Boddy, and Mark D. Fricker. Analysis of fungal networks. Fungal Biol Rev, 26(1):12–29, apr 2012.
[7] Atsushi Tero, Kenji Yumiki, Ryo Kobayashi, Tetsu Saiusa, and Toshiyuki Nakagaki. Flow-network adaptation in Physarum amoebae. In Theory Biosci, volume 127, pages 89–94. Springer-Verlag, may 2008.
[8] Jason W Rocks, Henrik Ronellenfitsch, Andrea J Liu, Sidney R Nagel, and Eleni Katifori. The limits of multifunctionality in tunable networks. Proc Natl Acad Sci U S A, jan 2019.
[9] Jason W. Rocks, Nidhi Pashine, Irminagd Bischoberger, Carl P. Goodrich, Andrea J. Liu, and Sidney R. Nagel. Designing allosterily-inspired response in mechanical networks. Proc Natl Acad Sci, 114(10):2520–2525, 2016.
[10] Herbert. Edelsbrunner and J. (John) Harer. Computational topology : an introduction. American Mathematical Society, 2010.
[11] Nina Otter, Mason A Porter, Ulrike Tillmann, Peter Grindrod, and Heather A Harrington. A roadmap for the computation of persistent homology. EPJ Data Sci, 6(1):17, dec 2017.
[12] Eleni Katifori and Marcelo O. Magnasco. Quantifying loopy network architectures. PLoS One, 7(6):e37994, jun 2012.
[13] Yuriy Mileyko, Herbert Edelsbrunner, Charles A. Price, and Joshua S. Weitz. Hierarchical ordering of reticular networks. PLoS One, 7(6):e36715, jun 2012.
[14] M. Kramar, A. Goullet, L. Kondic, and K. Mischaikow. Persistence of force networks in compressed granular media. Phys Rev E, 87(4):042207, apr 2013.
[15] Miroslav Kramár, Arnaud Goullet, Lou Kondic, and Konstantin Mischaikow. Quantifying force networks in particulate systems. Phys D, 283:37–55, aug 2014.
[16] J Roerdink and Arnold Meijster. The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. Fundam Informaticae, 41(1-2):187–228, jan 2000.
[17] Vanessa Robins, Peter John Wood, and Adrian P Sheppard. Theory and algorithms for constructing discrete morse complexes from grayscale digital images. IEEE Trans Pattern Anal Mach Intell, 33(8):1646–1658, aug 2011.
[18] Olaf Delgado-Friedrichs, Vanessa Robins, and Adrian Sheppard. Skeletonization and partitioning of digital images using discrete morse theory. IEEE Trans Pattern Anal Mach Intell, 37(3):654–666, mar 2015.
[19] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. Network flows: theory, algorithms, and applications. Prentice-Hall, Upper Saddle River, New Jersey, 1st edition, 1993.
[20] Michelle Girvan and M. E. J Newman. Modularity and community structure in networks. Proc Natl Acad Sci U S A, 103(23):8577–8582, jun 2006.
[21] M. E. J Newman. Finding community structure in networks using the eigenvectors of matrices. Phys Rev E, 74(3), 2006.
[22] Sidney Redner. Fractal and Multifractal Scaling of Electrical Conduction in Random Resistor Networks. In Robert A. Meyers, editor, Encyl Complex Syst Sci, pages 3737–3754. Springer, New York, New York, NY, 2009.
[23] Réka Albert, Hawoong Jeong, and Albert-László Barabási. Error and attack tolerance of complex networks. Nature, 406(6794):378–382, jul 2000.
[24] Antonino Paolo Di Giovanna, Alessandro Tibo, Ludovico Silvestri, Marie Caroline Müllcnbroich, Irene Costantini, Anna Letizia Allegra Mascaro, Leonardo Sacconi, Paolo Frasconi, and Francesco Saverio Pavone. Whole-Brain Vasculature Reconstruction at the Single Capillary Level. Sci Rep, 8(1):12573, dec 2018.
[25] Le Yan, Riccardo Ravasio, Carolina Brito, and Matthieu Wyart. Architecture and Co-Evolution of Allosteric Materials. Proc Natl Acad Sci, 2016.
[26] Holger Flechsig. Design of Elastic Networks with Evolutionary Optimized Long-Range Communication as Mechanical Models of Allosteric Proteins. Biophys J, 113(3):558–571, aug 2017.
[27] Sandipan Dutta, Jean-Pierre Eckmann, Albert Libchaber, and Tsvi Tlusty. Green function of correlated genes in a minimal mechanical model of protein evolution. Proc Natl Acad Sci, 115(20):E4559–E4568, may 2018.
[28] Le Yan, Riccardo Ravasio, Carolina Brito, and Matthieu Wyart. Principles for Optimal Cooperativity in Allosteric Materials. Biophys J, 114(12):2787–2798, jun 2018.
Appendix A: Tuning Protocol Notes

For this work, we follow the tuning procedure detailed in Ref. [8] with a small modification. Rather than tuning the relative change in the response of the target, here we tune the value of the target pressure difference directly. In addition, the pressure difference on a given edge is the difference between the pressures of the two nodes connected by that edge, with a sign that is arbitrary because the nodes are not ordered. This means that the sign of the target pressure difference before tuning can be negative. In such a case, a small amount of tuning is necessary even when $\Delta$ is zero.

Appendix B: Persistence and Simplification

Algorithm Details

In the main text, we describe simplified versions of the persistence and simplification algorithms. Here we provide the additional details that in conjunction with Refs. [10], [17], and [18], comprise the full versions of the algorithms used here. We will use the language of the aforementioned references to facilitate the merging of our approach with theirs.

We represent a network as a graph $G = (V, E)$ which is a tuple composed of a set of $N$ vertices (nodes) $V = \{v_1, \ldots, v_N\}$ and a set of $N_E$ edges $E = \{e_1, \ldots, e_{N_E}\}$. On each edge $i$ we define a function $g(e_i) = |\Delta p_i|$ which is the absolute value of the pressure difference on that edge.

We model a graph $G$ as a cell complex $K(G)$ composed of the collection of both the vertices (0-dimensional cells) and edges (1-dimensional cells). When necessary, we denote the dimension of a $p$-dimensional cell (or $p$-cell for short) by a superscript $\alpha^{(p)}$. We say a cell $\alpha^{(p)}$ is the face of another cell $\beta^{(q)}$ if $p \leq q$ and the vertices of $\alpha^{(p)}$ are a subset of the vertices of $\beta^{(q)}$. If $p$ is strictly less than $q$, we write this relationship as $\alpha^{(p)} < \beta^{(q)}$, while if $p \leq q$, we write $\alpha^{(p)} \leq \beta^{(q)}$.

1. Network Filtration

To perform both the persistence and simplification algorithms, we need to formally define a filtration on our cell complex $K(G)$. This requires prescribing an ordering on all cells, including both edges and vertices, with the requirement that a cell must always be ordered after its faces. In analogy to Ref. [17], we define the upper costar of an edge $x$ as the set of cells it introduces into the cell complex,

$$U(x) = \{ \alpha \in K \mid x \geq \alpha \quad \text{and} \quad g(x) = \min_{y \geq \alpha} g(y) \} \quad (B1)$$

These sets provide a unique non-overlapping partitioning of $K(G)$. Since we only have vertices and edges, these sets can only be composed of (i) a single edge, (ii) an edge and one of its vertices, or (iii) an edge and both of its vertices (In Ref. [17], function values are defined on the vertices, resulting in the use of lower stars. Here, however, we have function values that are defined on the edges, resulting in the use of upper costars). Now we can define level cuts of our cell complex, composed of all cells in upper costars whose defining edge has a function value less than $t$,

$$\mathcal{K}_t(G) = \{ \alpha \in K(G) \mid \alpha \in U(x) \quad \text{and} \quad g(x) \leq t, \quad \forall x \in E \} \quad (B2)$$

The resulting sequence of level cuts $\mathcal{K}_t(G)$ for increasing values of $t$ define the ascending filtration of $g(x)$ on $K(G)$ used to perform the standard persistence algorithm, which is described in detail in Ref. [10].

2. Discrete Gradient Vector Field

To perform our simplification process, we closely follow the procedure described in Refs. [17] and [18]. While we do not explain the entire procedure here, we will describe the modifications we had to make. The first step in the simplification process is to compute a discrete gradient vector field, $V$, composed of a collection of pairs of cells $(\alpha^{(p)} < \beta^{(p+1)})$ in $K_t(G)$ such that each cell is in at most one pair. Unpaired cells are called “critical cells” and represent essential topological features (analogous to critical points on a manifold). This vector field encodes the topological structure and is later used to determine which topological features to eliminate. In Ref. [17], constructing $V$ makes use of a lower star filtration which requires function values defined on the vertices. However, we define our function values on the edges and use an upper costar filtration. To accommodate this difference, we provide a new algorithm ProcessUpperCostars which is essentially the dual version of Algorithm 1. ProcessLowerStars in Ref. [17].

Similar to its counterpart, ProcessUpperCostars requires an ordering of all cells within each upper costar. Given a cell $\alpha \in U(x)$ with coface edges $\{x, y_{i_1}, \ldots, y_{i_k}\}$ (for a vertex, this list is composed of all adjacent edges, while for an edge it simply contains itself), define

$$G(\alpha) = (g(x), g(y_{i_1}), \ldots, g(y_{i_k}))$$

where

$$g(x) < g(y_{i_1}) < \cdots < g(y_{i_k}) \quad (B3)$$

Each cell is then ordered according to two criteria: (i) cell dimension ordered from smallest to largest (the faces of a cell must always appear before that cell) and (ii) the lexicographic ordering of these sequences from smallest to smallest. All other functions or objects in ProcessUpperCostars that we do not explicitly define are identical to those in Ref. [17] (or can be transparently inferred).
Algorithm 1 ProcessUpperCostars($\mathcal{E}, g$)

Input $\mathcal{E}$ set of edges in network
Input $g$ values on edges
Output $C$ critical cells
Output $V$ discrete vector field $V[\alpha^p] = \beta^{(p+1)}$

1: for $x \in \mathcal{E}$ do
2:    add all $\alpha \in U(x)$ to PQzero such that $\text{num}_{\text{unpaired cofaces}}(\alpha) = 0$
3:    add all $\alpha \in U(x)$ to PQone such that $\text{num}_{\text{unpaired cofaces}}(\alpha) = 1$
4:    while PQone $\neq \emptyset$ or PQzero $\neq \emptyset$ do
5:        while PQone $\neq \emptyset$ do
6:            $\alpha := \text{PQone.pop_front}$
7:            if $\text{num}_{\text{unpaired cofaces}}(\alpha) = 0$ then
8:                add $\alpha$ to PQzero
9:            else
10:               $V[\text{pair}(\alpha)] = \alpha$
11:               remove pair($\alpha$) from PQzero
12:               add all cells $\beta \in U(x)$ to PQone such that $(\beta < \alpha$ or $\beta < \text{pair}(\alpha))$ and $\text{num}_{\text{unpaired cofaces}}(\beta) = 1$
13:            end if
14:        end while
15:        if PQzero $\neq \emptyset$ then
16:            $\gamma := \text{PQzero.pop_front}$
17:            add $\gamma$ to $C$
18:            add all cells $\alpha \in U(x)$ to PQone such that $\alpha < \gamma$ and $\text{num}_{\text{unpaired cofaces}}(\alpha) = 1$
19:        end if
20:    end while
21: end for