Visual Analytics of Cascaded Bottlenecks in Planar Flow Networks

Tobias Post*, Christina Gillmann*, Thomas Wischgoll†, Bernd Hamann‡ and Hans Hagen*

*Computer Graphics and HCI Group, University of Kaiserslautern, Kaiserslautern, Germany
E-mail: tpost@rhrk.uni-kl.de, c_gillma@cs.uni-kl.de, hagen@cs.uni-kl.de

†Advanced Visual Data Analysis, Wright State University, Dayton, USA
E-mail: thomas.wischgoll@wright.edu

‡Department of Computer Science, University of California, Davis, USA
E-mail: hamann@cs.ucdavis.edu

***Abstract*—Finding bottlenecks and eliminating them to increase the overall flow of a network often appears in real world applications, such as production planning, factory layout, flow related physical approaches, and even cyber security. In many cases, several edges can form a bottleneck (cascaded bottlenecks). This work presents a visual analytics methodology to analyze these cascaded bottlenecks. The methodology consists of multiple steps: identification of bottlenecks, identification of potential improvements, communication of bottlenecks, interactive adaption of bottlenecks, and a feedback loop that allows users to adapt flow networks and their resulting bottlenecks until they are satisfied with the flow network configuration. To achieve this, the definition of a minimal cut is extended to identify network edges that form a (cascaded) bottleneck. To show the effectiveness of the presented approach, we applied the methodology to two flow network setups and show how the overall flow of these networks can be improved.**

***Index Terms*—Visual Analysis, Bottleneck Visualization, Flow Networks, Planar Graphs, Maximum Flows, Minimum Cuts**

I. Introduction

The analysis of flows is an important topic in various applications, such as cyber-security [14], biological pathway analysis [32], and cyber-physical manufacturing systems [15]. An important aspect in the context of optimizing such networks is the identification and elimination of bottlenecks [18].

The analysis of bottlenecks in such networks can be accomplished via flow networks, with entities represented as nodes/vertices and their relation as edges of the network. Depending on a network's setting, each of these edges has a specific capacity, describing the maximum flow between two connected entities. To identify a bottleneck of a network, the corresponding flow network is analyzed. Several requirements need to be fulfilled for an analysis to be meaningful (see Section II). Contrary to intuition, the bottleneck of a flow network is not a single edge between two nodes. Instead, a bottleneck is a set of edges. The minimum cut of a flow network can help with describing these bottlenecks. This cut separates the nodes of the flow network into two groups: one that can be reached by the network's source, and the other defined by the remaining nodes. In this graph-theoretical setup, the question arises how to identify the true bottleneck edge in the group of minimum-cut edges, how to visually encode this bottleneck, and how to increase the maximum flow in a flow network, when there are no nodes that can be reached by the source and sink simultanously (e.g., cascaded bottlenecks).

We introduce a visual analysis methodology for cascaded bottlenecks in planar flow networks. It is based on the work of Post et al. [21] and extends the definition of a minimum cut in a flow network by separating the nodes in a network into three groups: nodes that can be reached from the source, nodes that can reach the sink of the network, and the remaining nodes. This definition allows an enhanced classification of edges crossing these regions to identify those edges that are the bottlenecks of the network. When these regions are not attached to each other, the presented methodology allows users to identify potential bottlenecks, high-lighted for effective communication. To define an intuitive visualization of bottlenecks in a network, we present a visualization based on the Voronoi diagram [11] derived from the underlying graph's node layout. Color-coded regions indicate bottleneck transition in a flow network. Our visualization supports user interaction enabling users to manually adapt cascaded bottlenecks and increase the overall flow throughout a network. To study the influence of user defined changes, our approach involves a visual feedback allowing users to refine network settings until the network's flow properties are as desired, see Section III.

In summary, this paper makes the following contributions:

- An extended definition of the minimum cut for flow networks
- An intuitive visualization of a minimum cut in a flow network and cascaded bottlenecks
- An interactive visual analytics approach that allows users to increase overall flow in a network

To show the effectiveness of the presented approach, we improve the overall flow in two networks and discuss how it satisfies the defined requirements for bottleneck analysis (see Section IV).

LEVIA'18: Leipzig Symposium on Visualization in Applications 2018

II. Related Work

This section summarizes the state of the art in minimum cut visualization as well as techniques that help to increase the overall flow in a network. Further, this section describes the requirements that must be fulfilled to perform meaningful bottleneck analysis.

A. Comparative Visualization of Minimum Cuts

Vehlow et al. [30] presented a state-of-the-art report summarizing available network drawing methods with the goal of grouping the nodes of graphs. Although they presented a large variety of graph-drawing algorithms, an intuitive visual mapping of the minimum cut itself was not presented. In contrast, our approach introduces a visual encoding for the minimum cut based on Voronoi diagram tiles.

Brandes et al. [4] presented a planar visualization of the minimum cut in flow networks by arranging a network in a rectangular manner and adding a poly-line to indicate the cut. This method is widely used in open-source solutions discussed by [13], [16], [17]. Although this method provides a suitable first indication of the minimum cut, it cannot indicate edges in a flow network that represent a bottleneck in the network. Our approach utilizes the method of Brandes et al. [4] as a foundation and refines the definition of a minimum cut to enhance transitions that form the bottleneck of the considered system.

B. Cascaded Bottleneck Analysis

The identification and visualization of bottlenecks is an important problem when considering overall flow in a network [31]. This Section discusses the state of the art.

Alstott et al. [3] presented a method that supports the identification and adaptation of cascaded bottlenecks through a computational approach. Although this approach produces a new flow network with an overall increased flow, it does not allow users to review different options to increase the flow. Therefore, our method visually communicates available options to a user, to increase the overall flow of a network; visually guides are presented to the user in this process.

Scholz-Reiter et al. [27] presented an analytical approach to model dynamic bottlenecks in manufacturing systems. Their approach can be used to classify edges in a flow network as (cascaded) bottleneck edges. Although this approach allows one to classify bottleneck edges, it does not allow one to visually inspect and adapt them. Our approach therefore includes a visual tool to adjust bottleneck edges.

Qi et al. [23] presented a visual analytics approach to identify bottlenecks in a road network by examining the average speed of cars on a road segment. Although this technique shows where traffic is currently moving slowly, it does not provide insight to the real bottleneck edges in the flow network. Thus, we present an approach that permits the identification of bottleneck edges and allows users to adapt them.

Methods to examine and increase the overall flow in a water distribution system [24], [28] typically utilize graph theory concepts, such as minimum cuts, to compute the optimal setting of a water distribution network when considering specific parameters. Although this approach provides decision makers with a suitable overview of an optimal setting in a flow network, it does not indicate which edges in the flow network require adaptation to achieve the desired behavior. Our visual approach makes it possible to identify bottlenecks in a flow network and allows users to interactively adjust them.

C. Requirements for Bottleneck Analysis

In order to provide an effective analysis of cascaded bottlenecks in planar flow networks, the following requirements have to be satisfied.

R1: Identification of the bottleneck edges [25]

In order to increase the overall flow in a flow network, bottleneck edges must be identified. This includes single bottlenecks that consist of one edge as well as bottlenecks that are composed of multiple edges in the network.

R2: Identification of potential improvements [22]

In a variety of cases, more than one bottleneck can exist. Multiple edges in a flow network can be classified as bottleneck edges that can obtain a higher capacity in order to increase overall flow in a network. The goal is to identify different options.

R3: Communication of bottlenecks [5]

As users from different domains need to determine the edges in a flow network where capacity needs to be increased, the identified bottleneck edges need to be visually communicated to users in an intuitive manner.

R4: Interactive adaptation of flow [26]

A flow network usually corresponds to specific physical objects. Users need to be enabled to select the edge (objects) that require a larger capacity.

R5: Feedback loop [7]

After altering the capacity of edges, the overall flow and bottleneck edges can change. A feedback loop is required that permits users to examine the effects of new settings, enabling them to make further changes when necessary.

The overall goal is to provide a visual analytics tool that is capable of satisfying these requirements.

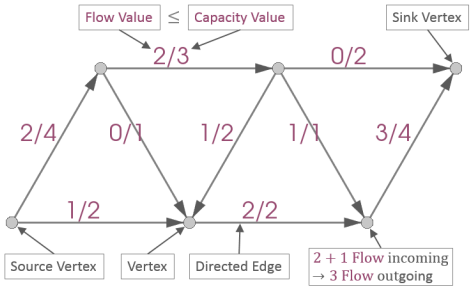


Fig. 1: Flow network consisting of vertices, directed edges, a source and sink vertex, and a flow and capacity value per edge. The capacity limits the flow. Except the source and sink, all vertices preserve the flow [21].

III. Methods

The analysis of bottlenecks in flow networks is an essential task for many real world applications in planning and engineering. This section presents a method to visually inspect single bottleneck fronts in planar flow networks developed by Post et al. [21]. Based on this, a method to analyze the propagation of bottlenecks for a network with an ensemble of different configurations is demonstrated.

A. Flow Networks

This section relies on the general definition of flow networks with a single source and sink, which is presented below. Figure 1 shows an example for such a flow network. A *network* $N = (G, c, s, t)$ consists of a directed graph $G = (V, E)$ with a finite set of vertices V and a set of directed edges $E \subseteq V \times V$. Here, the edges should not include self loops or multiple edges in the same direction between any two nodes. The capacity function $c : E \to \mathbb{R}_+$ assigns a non-negative capacity value to every edge in the network. The vertices $s, t \in V$ with $s \neq t$ should be the only source and sink in the network, respectively.

A *flow* $f : E \to \mathbb{R}_+$ is a function assigning a non-negative flow value to each edge in the network. Hence, a *flow network* is a network together with a specific flow on it. There are several constraints that apply to such a flow. The flow should be limited by the capacity, thus, $\forall e \in E : f(e) \leq c(e)$, i.e., the flow along an edge is never larger than the edge's capacity. Also, all vertices except the source and sink should preserve the flow, thus, $\forall v \in V \setminus \{s, t\} : \sum_{(w,v)\in E} f(w, v) = \sum_{(v,w)\in E} f(v, w)$, i.e., the total incoming flow is equal to the total outgoing flow for a vertex. For the source, the total outgoing flow is larger than the total incoming flow, and for the sink this is reversed.

The *value of an s-t-flow* $|f| = \sum_{(s,w)\in E} f(s, w) - \sum_{(w,s)\in E} f(w, s)$ is the value of the outgoing flow of the source s minus its incoming flow. Since all vertices except the source s and sink t preserve the flow, this is the same as the value of the incoming flow of the sink s minus its outgoing flow. This work focuses on planar flow networks. To restrict the general definition of (flow) networks to planar (flow) networks, the respective graph G should be planar. This means that G can be plotted in a plane without edges crossing each other. Figure 1 shows an example for a planar flow network with a proper embedding in the image plane.

B. Maximum Flows

A *maximum flow* $\hat{f}$ on a network N has the largest value among all possible flows on N, thus, there exists no other flow f with $|f| > |\hat{f}|$. Maximum flows are interesting, since lower capacity constraints could be used to achieve flows with smaller values. This means that given capacity constraints limit maximum flows only. Thus, to evaluate the full potential of networks, the maximum flows have to be analyzed. This leads to the question of how to find a maximum flow for a given network. The method of Ford and Fulkerson [10] is a general approach to find such a maximum flow. To understand this approach, the definition of a residual network needs to be used.

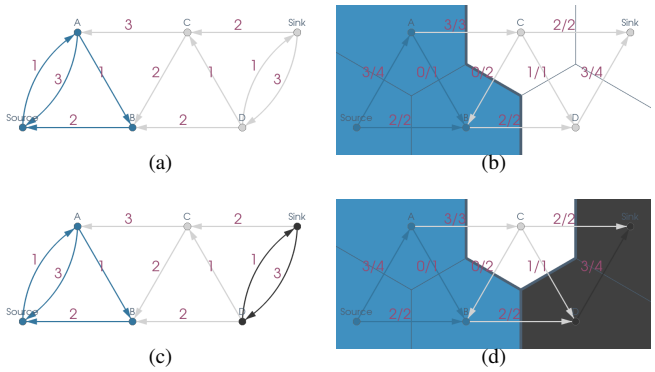


Fig. 2: Residual network of an exemplary network with maximum flow with vertices/edges reachable forwards from source or backwards from sink (left images), and the original network with minimum cut and classified Voronoi cells (right images). The classical construction of a minimum cut (upper images) suggests wrong bottleneck edges, while the new extended construction (lower images) shows the true bottleneck transitions (blue to black) [21].

For a given flow network $N = (G, c, s, t)$ with flow f the *residual network* is defined as $N_f = (G_f, c_f, s, t)$ with $G_f = (V, E_f)$. Thus, the vertices V and the source s and sink t of the residual network are the same as the ones of the given network, though the edges E_f and their capacities c_f change. The edges and capacities of the residual network are defined as follows. For each edge $(v, w) \in E$ a forward edge (v, w) is added to E_f if $f(v, w) < c(v, w)$. The capacity of such a new forward edge (v, w) is set to $c_f(v, w) = c(v, w) - f(v, w)$. For each edge $(v, w) \in E$ a backward edge (w, v) is added to E_f if $f(v, w) > 0$. The capacity of a new backward edge (w, v) is set to $c_f(w, v) = f(v, w)$. Following this definition, a residual network describes the amount of flow that can be added to an edge before the capacity limit is reached (forward edge), and the amount of flow that can be subtracted from an edge before a negative flow would arise (backward edge).

The method of Ford and Fulkerson operates on these residual networks. A directed path from the source to the sink is found in the residual network. This path is called an augmenting path, since the flow of the edges in the original network on this path can be improved thereby increasing the value of the overall flow in the network. So for a forward edge in the residual network, the flow of the original edge is increased and for a backward edge in the residual network, the flow of the original edge is decreased. This procedure is iterated as long as no more augmenting paths can be found in the residual network. It can be shown that the value of the resulting flow is maximal, so the resulting flow is a maximum flow.

The algorithm of Edmonds and Karp [8] uses a breadth-first-search from the source to always find a shortest augmenting path in the residual network. This ensures the termination of the algorithm as well as a polynomial bound of the algorithm's

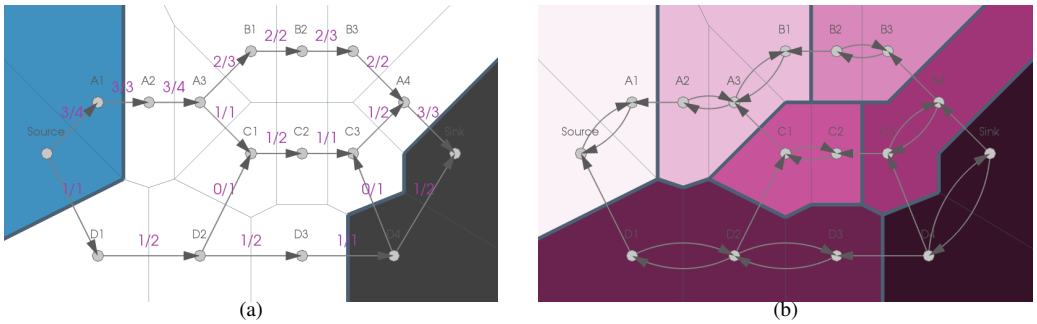


Fig. 3: A planar flow network and its extended minimum cut (image (a)). The spatial separation of the blue and black regions shows that the flow network does not have a single bottleneck front. A method to analyze cascaded bottlenecks is required. The strongly connected components for the residual network are shown color-coded (shades of purple), and their transitions form candidates for the cascaded bottlenecks of interest (image (b)).

run-time, leading to an efficient algorithm to find maximum flows. It can be shown that the run-time complexity of this algorithm is in $O(|V| \cdot |E|^2)$, i.e., the run-time is bounded asymptotically by $k \cdot |V| \cdot |E|^2$ for a fixed constant k, $|V|$ vertices and $|E|$ edges, and therefore is not dependent on the capacities. Although even lower-complexity algorithms with a complexity of nearly up to $O(|V| \cdot |E|)$ are known, the method of Ford and Fulkerson was demonstrated above, as the shown definitions like augmenting paths will be used in the following.

C. Minimum Cuts

To find bottlenecks in networks, maximum flows can be considered. For a given network with a path from source to sink, if all edges in the network were unsaturated the value of the flow could be increased. So for a maximum flow in this network there have to be saturated edges. For these edges the flow value equals the capacity value, so the flow cannot be increased any further. One could easily think that increasing the capacity of such an edge would result in a larger maximum flow, i.e., such an edge would be called a bottleneck edge in the following. It turns out that this intuition is incorrect and an increase of the capacity of such an edge is not guaranteed to increase the value of the maximum flow. To countervail this effect, this work focuses on cuts instead of flows.

An *s-t-cut* $C = (S, S')$ is a partition of the vertices V into the disjunct sets $S \subset V$ with $s \in S$ and $S' \subset V$ with $t \in S'$ such that $S \cup S' = V$. The *capacity of an s-t-cut* $|C| = \sum_{(v,w) \in E \,:\, v \in S \,\wedge\, w \in S'} c(v, w)$ is the sum of the capacities of edges from a vertex in S to a vertex in S'. A *minimum cut* $\check{C}$ of a network N has the smallest capacity among all possible cuts of N, so there exists no other cut C with $|C| < |\check{C}|$.

The max-flow min-cut theorem [9] from graph and optimization theory states $|\hat{f}| = |\check{C}|$, so the value of a maximum flow is equal to the capacity of a minimum cut and vice versa. This means that instead of considering maximum flows for the analysis of the performance and bottlenecks of networks, minimum cuts can be utilized.

The standard approach to find a minimum cut for a given network is to first calculate the maximum flow as described above, and then to collect all vertices that are reachable from the source vertex in the resulting final residual network. Those vertices form the set $\check{S}$ of the cut, with $\check{S}' = V \setminus \check{S}$ being the set of remaining vertices. The desired minimum cut is $\check{C} = (\check{S}, \check{S}')$.

Figure 2(a) shows the residual network of the maximum flow, the collection of vertices starting from the source in blue, and the remaining vertices in gray. To enhance the intuitiveness of the visualization and enable users to easily analyze minimum cuts, Figure 2(b) colors the Voronoi cells [11] of each vertex by a partition-specific color, blue for the vertices in S and gray for all other vertices in S'. The Voronoi cell of a vertex is the region of all points that are closer to this vertex then to all other vertices. By using Voronoi cells that share a common border to other cells of the same color, regions for both partitions of the minimum cut are formed.

As can be seen, the previously considered edge $(Source, B)$ starts and ends in the blue region and cannot be increased to increase the value of the maximum flow. Hence, this edge is not a bottleneck edge. In general, for all edges ending in S (blue region) by construction there exists a directed path in the residual network from the source to the endpoint of the edge. Thus, instead of increasing the capacity of such an edge, the flow along this path could be improved. An edge ending in S (blue region) cannot be a bottleneck edge. In contrast to this, one could investigate the behavior of an edge starting in the blue region and leading to the white region. As an example, the edge (A, C) with values "3/3" is considered. Again the intuition fails and the considered edge is not a bottleneck edge.

This shows that the general definition of a cut is not enough to find bottleneck edges. To compensate this shortcoming, this work extends the construction of a minimum cut by adding a third set $T \subset V$ to the partition. Figures 2(c) and 2(d) show the same visualizations as before, but this time all vertices that have a directed path to the sink in the residual network are collected in the set T and colored in black. All vertices that are not reached from the source or do not reach the sink form the set $R \subset V$ with $R = V \setminus (S \cup T)$ and are left white. The new partition is $P = (S, R, T)$ (blue / white / black regions)

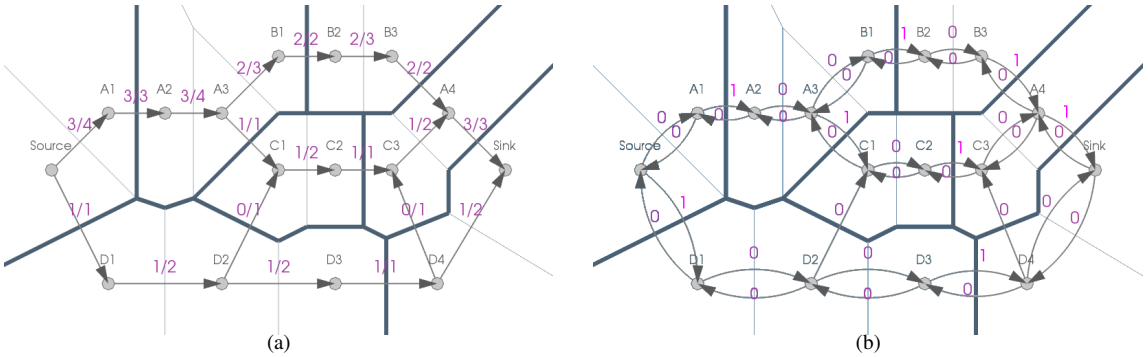


Fig. 4: The flow network from Figure 3(a) with its cascaded bottleneck candidates (image (a)). This flow network is used for the construction of the forward graph (image (b)).

with disjoint sets $S, R, T \subset V$, and $S \cup R \cup T = V$, and $s \in S$ and $t \in T$.

All edges starting in T (black region) cannot be bottleneck edges, since by construction there exists a directed path in the residual network from the starting point of the edge to the sink. All edges ending in R (white region) also cannot be bottleneck edges, since by construction they do not have a directed path in the residual network from their endpoint to the sink. Increasing the capacity of such an edge could increase the value of a flow from the source to the edge's endpoint, but not to the sink. The overall flow would not increase, hence the edge is no bottleneck. Analogously, edges starting in R (white region) also cannot be bottleneck edges.

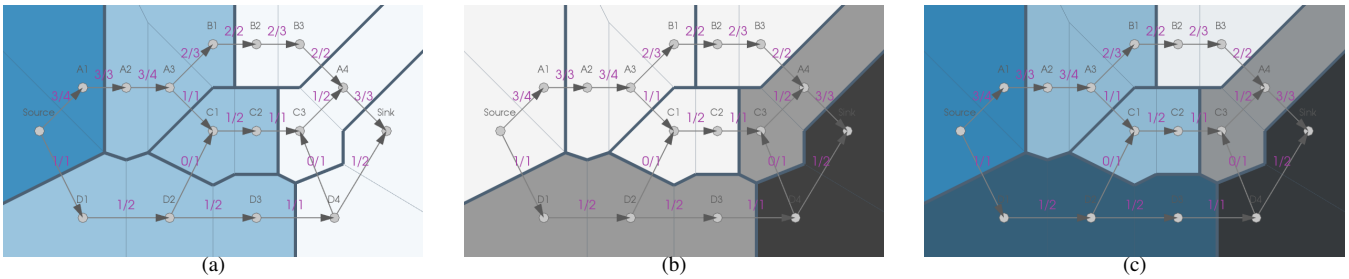
Proof: Let $(v, w) \in E$ with $v \in S$ and $w \in T$ be an edge leading from S (blue region) to T (black region). By construction, there exists a directed path $(v_1, v_2, ..., v_n)$ with $v_1, v_2, ..., v_n \in V$ and $v_1 = s$ and $v_n = v$ in the residual network from the source s to the starting point v of the edge. By construction there also exists a directed path $(w_1, w_2, ..., w_m)$ with $w_1, w_2, ..., w_m \in V$ and $w_1 = w$ and $w_m = t$ in the residual network from the endpoint w of the edge to the sink t. If both paths had a common vertex $v_i = w_j$, the path $(s = v_1, v_2, ..., v_{i-1}, v_i = w_j, w_{j+1}, ..., w_{m-1}, w_m = t)$ would be an augmenting path, and hence the given flow would not have been a maximum flow. Thus, both paths are disjoint and do not increase the overall flow without modifying $c(v, w)$. Also, the flow $f(v, w)$ of the given edge equals its capacity $c(v, w)$, because otherwise the edge (v, w) would be included in the residual network and the path $(s = v_1, v_2, ..., v_n = v, w = w_1, w_2, ..., w_n = t)$ would be an augmenting path. But by modifying the capacity $c(v, w)$ to a greater value $c'(v, w) > c(v, w)$ it holds that $f(v, w) < c'(v, w)$, thus, the edge (v, w) is included in the modified residual network. This leads to an augmenting path $(s = v_1, v_2, ..., v_n = v, w = w_1, w_2, ..., w_n = t)$ that can be used to increase the value of the overall flow. Hence, increasing the capacity of an edge from S to T increases the value of the maximum flow, thus, all edges from S (blue region) to T (black region) are bottleneck edges.

The overall approach works by first performing a max-flow calculation followed by two separate breadth-first-searches in the residual network starting forward from the source and backwards from the sink, respectively. Since the residual network has the same number of vertices and at most twice the number of edges than the original network, the run-time complexity of the breadth-first-searches is in $O(|V| + |E|)$, i.e., the complexity and limitations of the overall approach are dependent on the chosen max-flow algorithm, as described above.

We described a method to visually analyze single bottleneck fronts in planar flow networks. Here, transitions from S (blue region) to T (black region) were bottlenecks. As Figure 3(a) shows, there are cases where there are no direct edges leading from S to T since the blue and the black regions are separated spatially. These flow networks do not have a single bottleneck front but cascaded bottlenecks sequentially following one another. The question arises how the overall flow can be increased, since there is no single edge with a capacity limit that can be increased to do so.

To develop a method to analyze cascaded bottlenecks in planar flow networks, the strongly connected components (SCCs) [2] of the residual networks are evaluated (see Figure 3(b)). The SCCs can be calculated efficiently by Tarjan's algorithm [29] which has a run-time that is linear in the number of vertices and edges. The general definition of *strongly connected components* is a unique (except permutation) decomposition $C_1 \cup ... \cup C_k = V$ of the vertices V into a minimal number of disjunct sets $C_1, ..., C_k \subseteq V$ of mutually reachable vertices with $\forall\ v, w \in C_i : v\ reaches\ w$ for all $i \in \{1, ..., k\}$. The components have maximal size and there is a directed path from each vertex to each other vertex within the same SCC, and no directed path either to or from the vertices of another SCC. Since the residual graph is the graph of the residual flow, its SCCs indicate candidates for the bottlenecks. Additional flow can move freely within one SCC, while crossing the boundary between two neighboring SCCs might lead to the need to increase the capacity value of this particular edge (see Figure 3(b)).

The boundaries of the SCCs from Figure 3(b) are used to show the cascaded bottlenecks of interest in Figure 4(a).



(a) (b) (c)

Fig. 5: The forward/backward graphs are used to calculate the shortest forward/backward distance from the source/sink to each vertex, respectively (images (a) and (b)). The distances range from 0 (strongly saturated color) to 2 (weakly saturated color). By utilizing different color channels both distances can be displayed simultaneously in an unambiguous way (image (c)).

Since there is no single edge with a capacity value that could be increased to increase the overall flow, the question arises which capacity values to increase. To tackle this issue, the forward graph (see Figure 4(b)) is constructed. For a given flow network $N = (G, c, s, t)$ with directed graph $G = (V, E)$ and flow f the *forward graph* is defined as the weighted graph $G_F = (V, E_F, w_F)$. The vertices V are the same as the ones of the given network, though the edges E_F with their new weights w_F change. The edges and weights of the forward graph are defined as follows: For each edge $(v, w) \in E$ with $f(v, w) < c(v, w)$ a forward edge (v, w) with weight $w_F(v, w) = 0$ is added to E_F. For each edge $(v, w) \in E$ with $f(v, w) = c(v, w)$ a forward edge (v, w) with weight $w_F(v, w) = 1$ is added to E_F. For each edge $(v, w) \in E$ with $f(v, w) > 0$ a backward edge (w, v) with weight $w_F(w, v) = 0$ is added to E_F. The definition of the *backward graph* $G_B = (V, E_B, w_B)$ is analogous but with reversed edge orientations.

The forward and backward graphs can now be utilized to calculate the distance of the shortest weighted path from source and sink to each vertex, respectively (see Figure 5). These distances are now called *forward distance* and *backward distance*, respectively. This can be done efficiently by Dijkstra's algorithm [6] in $O(|E| + |V| \cdot log|V|)$ run-time. The construction of the forward and backward graphs ensure that only forward edges that are saturated in the flow network increase the distance. When those edges are used in a shortest path within the forward or backward graph, their capacity values need to be increased to transport additional flow. Since shortest paths are used, it is ensures that only a minimal number of these network edges have to be adapted to increase the overall flow. In the following it is demonstrated how this can be applied to develop a method to analyze cascaded bottlenecks.

IV. Results

To show the effectiveness of our approach, we tested it by applying it to two flow network examples.

A. Visual System for cascaded bottleneck analysis

The methods presented in Section III can be utilized to interactively analyze cascaded bottlenecks in planar flow networks, see Figure 6. In this case, for each strongly connected component (SCC) all combinations of one edge going in and one edge going out of the same component are connected by a minimal augmenting path in the residual network. Additional flow can travel freely on these paths, while the incoming and outgoing edges themselves can be bottlenecks. In contrast to that, by construction a transition from an SCC of one color to an SCC of another color always indicates a bottleneck, since the forward or the backward distance has changed between SSCs. This is the reason for the construction and visualization of the forward/backward distance rendered as color-coded components in Figure 6.

The augmenting paths within each SCC are shown as spline curve segments in Figure 6. Each segment can by selected by the user. Not all possible segments are shown. To enhance usability and restrict the selection to meaningful segments, segments are filtered and unwanted segments discarded. Here, all segments that start or end with an edge decreasing in forward distance or increasing in backward distance are omitted. These segments lead to SCCs that can be reached more efficiently by a different path and are discarded.

When a continuous path from source to sink is formed by the selected segments, this path is used to increase the capacities of bottleneck edges along the path. The capacities are increased by the minimal residual flow of a non-bottleneck edge along the path. After updating the maximum flow computations, at least one non-bottleneck edge on the path becomes saturated and the overall flow is increased as much as possible without adjusting non-bottleneck edges. This process describes one iteration shown in Figure 6 per column, demonstrating the effectiveness of our interactive method to analyze cascaded bottlenecks.

B. Multiple Sources and Sinks

In order to show the applicability of the presented approach to flow networks containing more than one source and sink, we applied the presented methodology to a exemplary water supply flow network.

The network can be reviewed in Figure 7(a). It contains 5000 nodes with 10000 edges. 10 of the nodes are source nodes and 10 are sink nodes. Nodes can be water reservoirs

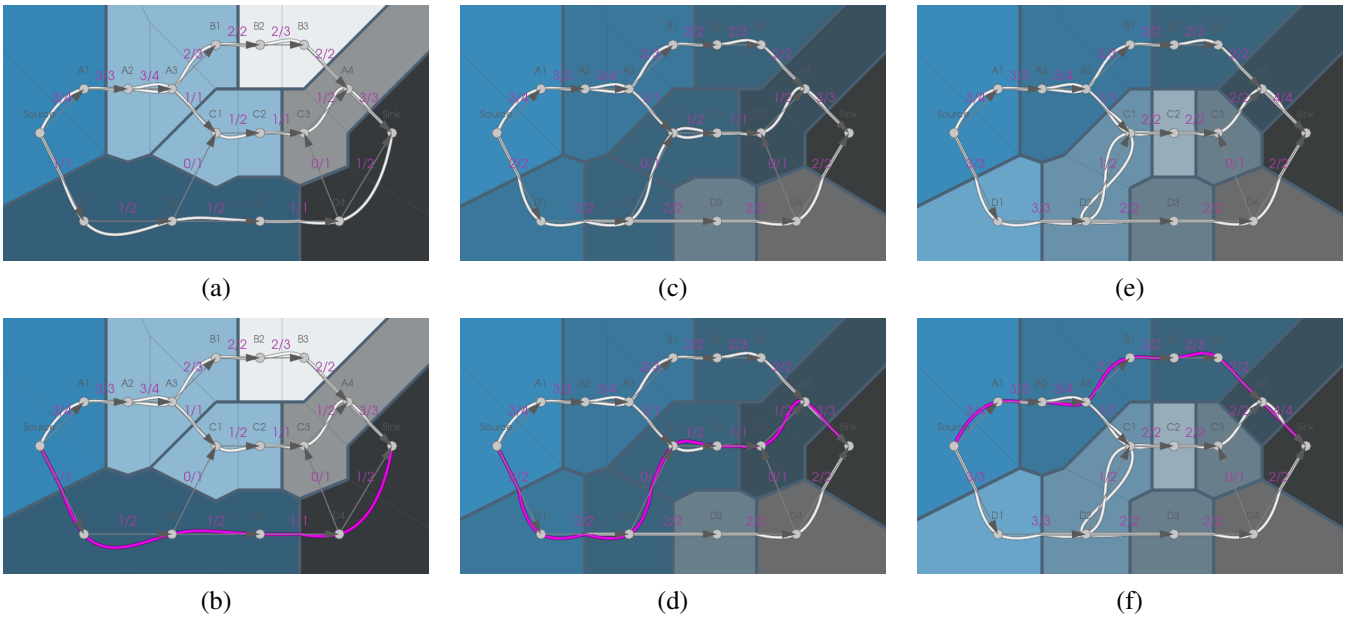


Fig. 6: Iterations of the feedback loop. The current flow network and all its filtered path segments are represented by spline curves (white) in the top images. The user can select path segments (magenta) and construct a path from source to sink (bottom images). This path is applied by increasing edge capacities on the path accordingly and calculating the increased overall flow for the next iteration. Only the capacities of edges leading from one to another component must be considered for potential adjustment. The component colors indicate the relative effort to send flow from the source to the component (blue), or from the component to the sink (black). Strongly saturated colors indicate relatively less effort and thereby fewer bottlenecks that must be overcome to increase overall flow.

(indicated by the blue tank icon), factories (indicated by the gray factories icon) or residential areas (indicated by the gray houses).

The resulting visualization of cascaded bottlenecks can be reviewed in Figure 7(b). The resulting cascaded bottleneck edge is highlighted in purple. In the examined network, the goal is to identify the bottleneck edges between water reservoirs and factories in order to promote a working economy system for the future.

In this example, it becomes clear, that the bottleneck edges would be hard to determine without utilizing the provided visualization. It can be seen, that one factory is the end of a bottleneck edge that connects this factory with a water reservoir.

In the presented visualization, users can select specific edges along the highlighted bottleneck edges and adjust them as shown in the previous example. Therefore, decision makers can create future restructuring plans for the water supply service.

V. Discussion

The following Section will discuss the ability of the presented approach to meet the defined requirements from Section II-C as well the requirements, that need to be fulfilled in order to gain user acceptance in real world applications.

A. Discussion of Requirements for Bottleneck Analysis

As discussed in Section II, an effective visual analytics tool for cascaded bottlenecks must satisfy specific requirements. In the following, we summarize how these requirements are satisfied by our methodology.

R1: Identification of bottlenecks

The presented methodology introduces a mathematical basis to identify bottlenecks in flow networks. In addition, direct bottleneck edges can be differentiated from cascaded bottlenecks by determining whether there exists an edge that can be reached by the sink and the source simultanously.

R2: Identification of potential improvements

To identify potential improvements, this research presents a methodology that classifies the different parts of a flow network where flow can circulate freely. The transition of these areas potentially improves overall flow in the network.

R3: Communication of bottlenecks

To communicate network characteristics effectively, we have devised a visual system that encodes bottlenecks of a flow network and visually highlights areas where flow can circulate freely. A visualization also indicates cascaded bottleneck edges.

R4: Interactive adaptation of flow

As the goal is to increase overall flow in a network, the presented methodology allows a user to interact with the flow network being analyzed. A user can increase cascaded bottle-

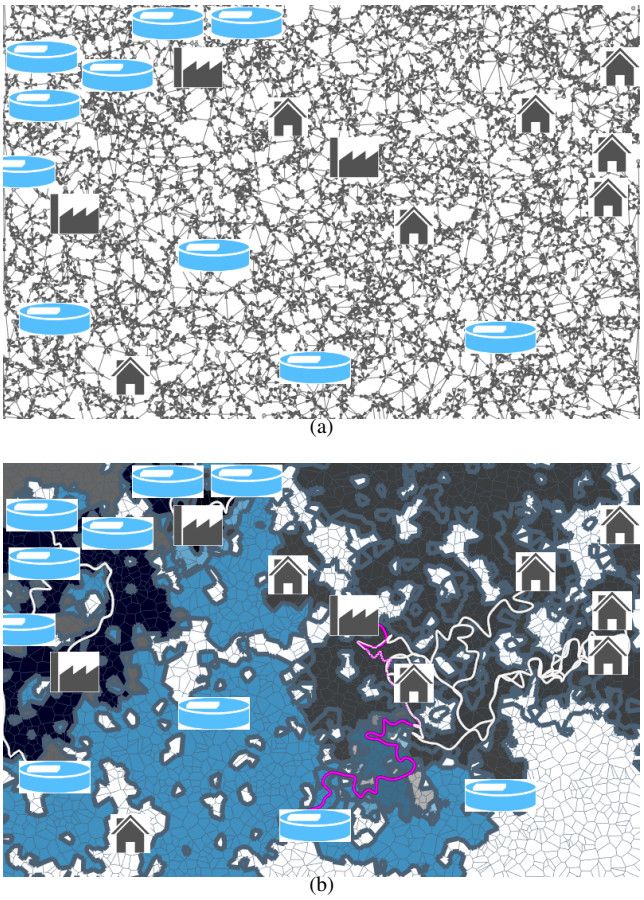


Fig. 7: The presented approach applied to a flow network with multiple sources and sinks. The network presents a water supply system in a town with residential areas, factories and water reservoirs. The purple line indicates a cascaded bottleneck between a water reservoir and a factory.

neck edges. This task is supported by an intuitive guidance provided to the user in the entire cascaded bottleneck.

R5: Feedback loop

When changing a flow network configuration, the overall flow and bottlenecks of the network can change. Our system communicates a newly designed flow network to the user and therefore supports a visual feedback loop. Users have the ability to improve a flow network until they are satisfied with its properties.

B. Discussion of Requirements for a Real World use of Visualizations

Gillmann et al. [12] formulated requirements, that need to be fulfilled to promote a real world use of a novel visualization technique. Namely they are usability, effectiveness, correctness, flexibility and intuitiveness. The ability to address the mentioned requirements is summarized below.

The usability of the presented approach is ensured by offering an interactive visualization, that directly encodes edges forming a bottleneck. Furthermore, users are enabled to adjust cascaded bottlenecks to achieve an overall increased flow in the underlying flow network.

The presented approach is effective in terms of computational and storage effectiveness. In addition, the visual guiding allows users to directly identify cascaded bottlenecks and how they can be improved. Still, with increasing complexity of the underlying flow network, the risk of visual clutter or overwhelming increases. This problem could be solved by applying focus and context techniques, such as hierarchical clustering of graphs. A survey of available techniques can be found in [1].

The description of the underlying utilized and defined mathematical concepts is based on well known and proved graph theoretical concepts such as minimal cuts, ensuring the correctness of the presented approach.

Furthermore, the presented approach is able to address a variety of different flow networks. Multiple sources, as well as arbitrary branching degrees can be computed. The underlying mathematical concepts are not restricted to planar flow networks. In fact, the concept of Voronoi cells is not restricted in terms of dimensionality as well. Still, a suitable interaction and visualization methodology would be required in such a case. The input flow networks can origin from a variety of application scenario reaching from traffic control over supply network to logistic tracking. Furthermore, flow networks can be extracted from medical image data [19], [20] to examine the flow in a vein system of a patient.

Finally, the intuitiveness of the approach is ensured, by utilizing a intuitive color scheme encoding cascaded bottlenecks and providing an easy to use interaction mechanism, that allows to adapt bottleneck edges, to achieve an increased flow in the considered flow network.

VI. Conclusion

We have introduced a novel approach to visualize bottlenecks (single and cascaded) in flow networks applicable to a variety of real-world applications. For example, product flows and constraints of a manufacturing system can be mapped to a network. We extended the definition of a minimum cut of a network to identify bottleneck edges. This extended definition was used as a basis to visualize minimum cuts and bottlenecks in production systems based on Voronoi regions. This approach supports a fast and intuitive identification of bottleneck transitions in a flow network. Based on this definition, cascaded bottlenecks can be identified. To improve them, users need to increase the capacity of multiple edges in a flow network. The presented work visually encodes all possible improvements and provides intuitive interaction for users.

Acknowledgments

This research was funded by the German Research Foundation (DFG) within the IRTG 2057 “Physical Modeling for Virtual Manufacturing Systems and Processes”.

References

[1] C. C. Aggarwal and H. Wang. *A Survey of Clustering Algorithms for Graph Data*, pages 275–301. Springer US, Boston, MA, 2010.

[2] A. V. Aho, J. E. Hopcroft, and J. Ullman. *Data Structures and Algorithms*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1983.

[3] J. Alstott, S. Pajevic, E. Bullmore, and D. Plenz. Opening bottlenecks on weighted networks by local adaptation to cascade failures. *Journal of Complex Networks*, 3(4):552–565, 2015.

[4] U. Brandes, S. Cornelsen, and D. Wagner. *How to Draw the Minimum Cuts of a Planar Graph*, pages 89–119. Springer Berlin Heidelberg, 2001.

[5] L. Braun, M. Volke, J. Schlamp, A. von Bodisco, and G. Carle. Flow-inspector: a framework for visualizing network flow data using current web technologies. *Computing*, 96(1):15–26, 2014.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[7] Z. Dong, Y. Pan, Z. Zhang, Y. Dong, and X. Huang. Modeling and control of fluid flow networks with application to a nuclear-solar hybrid plant. *Energies*, 10(11):1–21, 2017.

[8] J. Edmonds and R. M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, 1972.

[9] P. Elias, A. Feinstein, and C. Shannon. A note on the maximum flow through a network. *Information Theory, IEEE Transactions on*, 2(4):117–119, 1956.

[10] L. R. Ford and D. R. Fulkerson. Maximal Flow through a Network. *Canadian Journal of Mathematics*, 8:399–404, 1956.

[11] S. Fortune. Voronoi diagrams and delaunay triangulations. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 377–388. CRC Press, Inc., 1997.

[12] C. Gillmann, H. Leitte, T. Wischgoll, and H. Hagen. From Theory to Usage: Requirements for successful Visualizations in Applications. In *IEEE Visualization Conference (VIS) - C4PGV Workshop*, 2016.

[13] S. Halim. https://visualgo.net/maxflow. *online*, 2017.

[14] J. Jaffe. Bottleneck flow control. *IEEE Transactions on Communications*, 29(7):954–962, 1981.

[15] M. Kikolski. Identification of production bottlenecks with the use of plant simulation software. *Ekonomia i Zarzadzanie*, 8(4):103–112, 2017.

[16] S. Klamt, J. Saez-Rodriguez, and E. D. Gilles. Structural and functional analysis of cellular networks with cellnetanalyzer. *BMC Systems Biology*, 1:open access, 2007.

[17] S. Klamt and A. von Kamp. An application programming interface for cellnetanalyzer. *BioSystems*, 105:162–168, 2011.

[18] C. G. Lee and S. C. Park. Survey on the virtual commissioning of manufacturing systems. *Journal of Computational Design and Engineering*, 1(3):213 – 222, 2014.

[19] T. Post, C. Gillmann, T. Wischgoll, and H. Hagen. Fast 3D Thinning of Medical Image Data based on Local Neighborhood Lookups. In *EG/VGTC Conference on Visualization (EuroVis) - Short Papers*, 2016. doi: 10.2312/eurovisshort.20161159.

[20] T. Post, C. Gillmann, T. Wischgoll, and H. Hagen. OpenThinning: Fast 3D Thinning based on Local Neighborhood Lookups. In *IEEE Visualization Conference (VIS) - VIP Workshop*, 2016.

[21] T. Post, B. Hamann, H. Hagen, and J. C. Aurich. Ensemble Visualization of Bottlenecks in Planar Flow Networks. In *Physical Modeling for Virtual Manufacturing Systems and Processes*, volume 869 of *Applied Mechanics and Materials*, pages 234–243. Trans Tech Publications, 2017.

[22] A. P. Punnen and R. Zhang. Bottleneck flows in networks. *CoRR*, abs/0712.3858, 2007.

[23] H. Qi, M. Liu, L. Zhang, and D. Wang. Tracing road network bottleneck by data driven approach. *PLOS ONE*, 11(5):1–16, 05 2016.

[24] F. Rahmani, K. Muhammed, K. Behzadian, and R. Farmani. A graph theory based configuration of water distribution systems for optimum operation, 07 2016.

[25] C. Roser, M. Nakano, and M. Tanaka. A practical bottleneck detection method. In *Proceedings of the 33Nd Conference on Winter Simulation*, WSC '01, pages 949–953, Washington, DC, USA, 2001. IEEE Computer Society.

[26] J. Schlamp, R. Holz, Q. Jacquemart, G. Carle, and E. W. Biersack. HEAP: reliable assessment of BGP hijacking attacks. *IEEE Journal on Selected Areas in Communications*, 34(6):1849–1861, 2016.

[27] B. Scholz-Reiter, K. Windt, and H. Liu. Modelling dynamic bottlenecks in production networks. *International Journal of Computer Integrated Manufacturing*, 24(5):391–404, 2011.

[28] R. J. Shen, Q. G. Jia, Y. Y. Liang, and J. Zhang. Identify the bottleneck of water network by using graph theory. In *Materials Science and Information Technology*, volume 433 of *Advanced Materials Research*, pages 4794–4797. Trans Tech Publications, 2 2012.

[29] R. Tarjan. Depth first search and linear graph algorithms. *SIAM JOURNAL ON COMPUTING*, 1(2), 1972.

[30] C. Vehlow, F. Beck, and D. Weiskopf. Visualizing group structures in graphs: A survey. *Computer Graphics Forum*, pages n/a–n/a, 2016.

[31] Y. Wang, Q. Zhao, and D. Zheng. Bottlenecks in production networks: An overview. *Journal of Systems Science and Systems Engineering*, 14(3):347–363, Sep 2005.

[32] H. Yu, P. M. Kim, E. Sprecher, V. Trifonov, and M. Gerstein. The importance of bottlenecks in protein networks: Correlation with gene essentiality and expression dynamics. *PLoS Computational Biology*, 3(4), 2007.