VISUAL ANALYSIS FOR GRAPHS, NETWORKS AND FLOWS

Vom Fachbereich Informatik der
Technischen Universität Kaiserslautern
zur Verleihung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

genehmigte
DISSERTATION

von

TOBIAS M. POST

Datum der wissenschaftlichen Aussprache: 22.03.2018
Dekan: PROF. DR. STEFAN DESSLOCH
Berichterstatter: PROF. DR. HANS HAGEN
Berichterstatter: PROF. DR. BERND HAMANN

D 386

Abstract

Graphs and flow networks are important mathematical concepts that enable the modeling and analysis of a large variety of real world problems in different domains such as engineering, medicine or computer science. The number, sizes and complexities of those problems permanently increased during the last decades. This led to an increased demand of techniques that help domain experts in understanding their data and its underlying structure to enable an efficient analysis and decision making process.

To tackle this challenge, this work presents several new techniques that utilize concepts of visual analysis to provide domain scientists with new visualization methodologies and tools. Therefore, this work provides novel concepts and approaches for diverse aspects of the visual analysis such as data transformation, visual mapping, parameter refinement and analysis, model building and visualization as well as user interaction.

The presented techniques form a framework that enriches domain scientists with new visual analysis tools and help them analyze their data and gain insight from the underlying structures. To show the applicability and effectiveness of the presented approaches, this work tackles different applications such as networking, product flow management and vascular systems, while preserving the generality to be applicable to further domains.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Graphs, networks and flows are important concepts with an increased demand for visual analysis. This chapter motivates the need of research for this kind of visual analysis and explores potential application areas suitable for these concepts. Based hereon, this work presents different approaches for the visual analysis of graphs and flow networks that will be summarized in the following.

1.1 Application Areas of Graphs, Networks and Flows

Graphs, composed of nodes and edges, can be utilized to model and analyze a large variety of real world problems. Examples can be found in engineering, medicine, computer science and other domains. This list of examples can be continued extensively, covering all areas of research from molecular and genetic maps to biochemical pathways and protein functions [36].

Flows indicate the amount of movement and extend graphs to form flow networks. Examples for real world problems that can be modeled with flows or networks are network traffic [67], public traffic [104], product and material flows, computer networks, and data mining tasks for social media [76]. Flow networks are an important tool in these domains to identify main streams, bottlenecks, invariances and even security issues.

Although flows and networks are able to model a bigger variety of real world problems, their examination and analysis is more complex due to an increased degree of freedom and number of parameters. Therefore, domain scientists that are working with flow networks require suitable tools to model, adapt, control and analyze these networks. This effect is strengthened by the observation that datasets

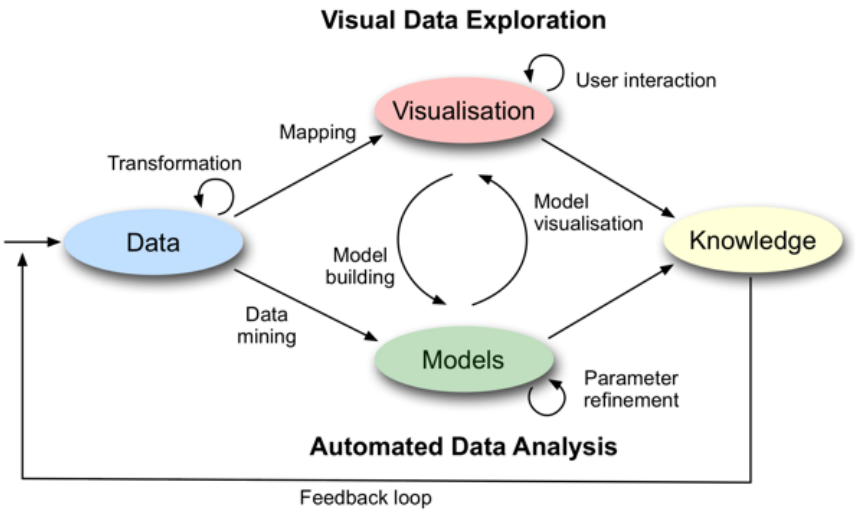


FIGURE 1.1: The process of visual analytics as shown by Keim et al. [54]. Visualization plays a central role in gaining new knowledge.

permanently became larger and more complex during the last decades. This led to an increased demand of analysts and decision makers, who need to be able to understand their data, identify critical aspects or situations and adapt the setting of their graphs or flow networks when required.

1.2 The Need of Visual Analysis for Graphs, Networks and Flows

The term **visual analytics** was formed by Keim et al. [54]. They proposed the visual analytics process as : *Analyse First - Show the Important - Zoom, Filter and Analyse Further - Details on Demand* (see Figure 1.1). This paradigm became a suitable solutions for a variety of problems covering physics, astronomy, business, environmental monitoring, security and medicine. As the visual analysis process shows, visualization plays a fundamental role to retrieve new insights from datasets and can be utilized to understand the underlying structures, thereby enabling an efficient analysis and decision making process in a variety of applications.

In the article of Keim et al. [54] networks where also mentioned as an potential application of visual analytics. This is caused by the fact that domain scientists need to be involved in the data processing loop to perform a meaningful analysis. In addition, the real world scenarios that can be modeled and analyzed by graphs and

flow networks are constantly increasing, introducing novel challenges. Therefore, the main goals for visual analysis of graphs and flow networks are stated as:

- Integration of massive information flows, covering data transformation, cleaning and filtering [62]
- Provide meaningful visual representations that allow intuitive user interactions. This especially includes the visualization of network meta information. [62, 13]
- Development of techniques that identify changes and invariances in graphs or networks, covering the identification of undesired behavior, clustering of the data and the detection of bottlenecks in network flows.

Therefore, the goal of this work is to provide techniques that are able to tackle the mentioned challenges using the concept of visual analysis to help domain scientists to analyze and understand graph and flow network based datasets more efficient and reliable.

1.3 Contributions and Structure of this Work

In this work novel concepts and techniques for the visual analysis of graphs, networks and flows are presented and their significance to real world applications is shown, covering the domains of engineering, medicine and computer science, and is structured as follows.

Software-definened networks (SDN) is a novel configuration technique that has the potential to become the future backbone of computer networking. SDNs have to be simulated and analyzed to identify applicable configuration settings for real world applications. To determine the quality of a SDN configuration, its packet flow is an important indicator for the analysis. This work presents an interactive system for the analysis of SDN data. An intuitive overview of the SDN hierarchy and the underlying packet flow is provided. The ability to track packets through the SDN by interlinked multiple views forms an novel interactive analysis tool for SDN data (see Chapter 2).

Another real world scenario that can be modeled by flow networks are cyber-physical production systems. This work presents a user-guided visual analysis approach that can answer relevant questions concerning the behavior of cyber-physical systems. The approach generates visualizations of aggregated views that capture an entire production system as well as specific characteristics of individual data features (see Chapter 3).

A flow network often does not solely contain the information of its nodes, connections and flows. In many cases, the single entities of a graph or flow network can consist of multi-dimensional attributes that can be relevant for domain experts. To include this data in the visual analysis process, this work offers a scale-invariant measure based on Pareto optimality that is able to indicate the quality of data points with respect to the Pareto front. In cases where datasets contain noise or parameters cannot easily be expressed or evaluated mathematically, the presented measure provides a visual encoding of the environment of a Pareto front to enable an enhanced visual inspection (see Chapter 4).

In many application graphs or networks cannot be determined directly from the available data. An example are image-based methods where some data needs to be converted to graphs. To tackle these cases, a concept called thinning can be used to provide a one pixel wide representation of visible objects in an image. Therefore, this work presents a novel thinning approach based on a pre-evaluated moving local neighborhood, resulting in an efficient and robust technique (see Chapter 5).

An important aspect of flow networks is the identification of bottlenecks. To tackle this challenge, this work enhances the comparability of different network configurations by utilizing ensemble visualization techniques. In addition, it introduces a novel approach to identify cascaded bottlenecks and evaluate their properties. The presented interactive techniques enable users to explore and analyze planar flow networks (see Chapter 6).

In its entirety this work covers multiple aspects of visual analysis for graphs, networks and flows in different domains, and shows their applicability and efficiency for a variety of applications.

Chapter 2

Flow Tracking in Software-Defined Networking

2.1 Summary

The constantly increasing digitalization of the modern society raises problems in conventional networking such as high complexity, inconsistent policies and scalability issues [80, 31]. To tackle these problems, new networking methodologies such as software-defined network/ing (SDN) are required. In contrast to conventional networks where each networking node is configured separately, SDNs provide controller elements that are able to administrate groups of nodes. This chapter considers each networking node to be administered by exactly one controller element, resulting in a hierarchy for the SDN.

The indirect administration of networking nodes enables SDNs to separate the control plane and the data plane of a network. Although this concept holds the potential to be the future backbone of networking, it also raises new challenges in the field of network segmentation and security, traffic engineering, as well as network provisioning and configuration [59]. Due to these challenges, SDNs are not yet widely applied in real world scenarios. Instead, network analysts run SDN simulations with different settings in order to understand the effects of SDN design choices to the resulting network behavior. An important factor that indicates the quality of the SDN settings is the resulting flow of packets through the network [51].

To analyze the flow of a network, visualization is a common tool. Although various successful network and flow visualization techniques are available, they cannot be applied directly to review a flow in a SDN (see Section 2.2). This is mainly caused by a combination of two effects: First, current approaches do not cover the hierarchical structure and the resulting packet flows of SDNs. Second, current approaches that utilize videos for time-dependent data often result in a phenomenon called change blindness [107].

This chapter is based on the work of Post et al. [94] and presents visualizations forming an interactive analysis tool for SDN data in Section 2.3. A linked view system is presented that relates the hierarchical overview of a SDN to the resulting packet flow through selected elements of the network. By utilizing an interactive brushing and linking approach, network analysts can select nodes or packets of the network and visually track them in a static flow view. This view is designed to identify coherences of packet paths throughout the SDN while avoiding change blindness. Additional interlinked tabular views help in displaying node and packet properties.

Therefore, this chapter contributes:

- Intuitive visualizations of SDN data
- Visually guided flow tracking in SDNs

To show the applicability of the presented visualizations, the analysis of a simulated SDN dataset is tackled in Section 2.4 and concluded in Section 2.5.

2.2 Related Work

The following section will discuss the recent work targeting the visualization of SDNs as well as network flow visualization techniques.

An overview of network visualization techniques can be found in the work of Guimares et al. [32]. Also, several tools [6, 47, 69] are suitable to review SDN controller nodes and their connections. Unfortunately, these tools focus only on the topology of the controller elements within a SDN. In contrast to that, this chapter presents visualizations enabling the analysis of all network nodes within an SDN.

To visualize the flow in a network, statistical methods such as graphs, showing the amount of network traffic per node [86] or node connectivity matrices [117, 45], can be used. Although these techniques provide a suitable overview, they do not make use of a network layout technique. A spatial flow visualization can be achieved by utilizing particles that flow between nodes in a network [112] or utilizing a space-time cube [4]. Although this gives a visual representation of the flow in a network, it introduces visual clutter and can cause change blindness due to animations. In contrast, this chapter presents a static flow visualization that resolves change blindness while relating to the underlying network layout.

To reduce visual clutter, edge bundling [40, 83] summarizes similar connections between nodes. This concept can be extended to distinguish between flow directions [105] or bundle time-varying flow data [81]. As SDNs induce a hierarchical structure (see Section 2.3), classical edge bundling methods cannot be used in this case. Instead, hierarchical edge bundling is required [39]. This technique was already successfully applied to time-varying data [46]. In [41] connections between hierarchically organized structures are bundled. These techniques are used as a starting point for the visualization of SDN in this chapter. Also, the technique of visualizing a storyline with respect to a hierarchy as shown in [68, 98] is extended and adapted to SDN.

2.3 Methods

The following section presents an interactive analysis tool to examine the topology of a given SDN and the flow of packets within. The user is able to specify a temporal interval of interest, referenced as the window. Additionally, networking nodes or packets of interest can be monitored or tracked within the SDN.

2.3.1 Overview

In SDN, controller elements administrate groups of networking nodes. This chapter considers each networking node being administered by exactly one controller element. This leads to a hierarchical tree representation for the SDN. The root node represents the whole SDN. The children of the root node represent the individual

controller elements, referred to as control nodes in the following. The children of a specific controller element (control node) are the nodes in the network that are administered by this controller, referred to as physical nodes, referenced by their IP addresses. The children of one administered node (physical node) are the port nodes of this specific IP node.

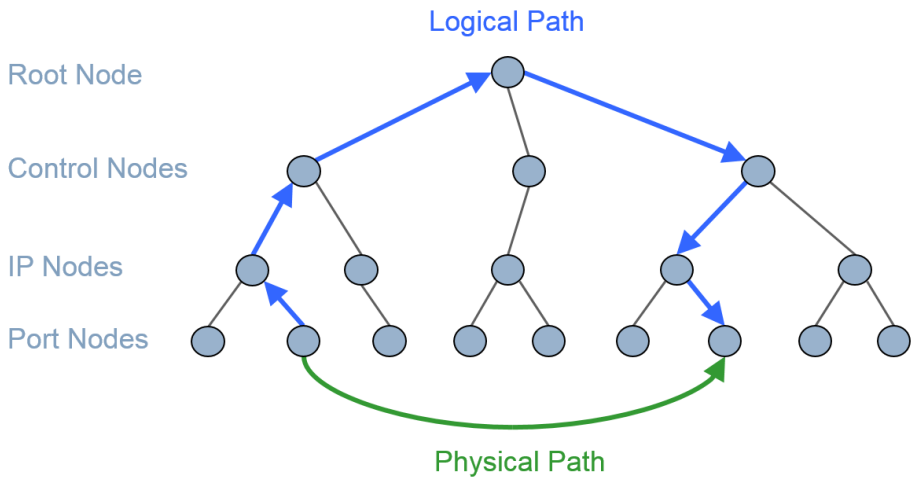
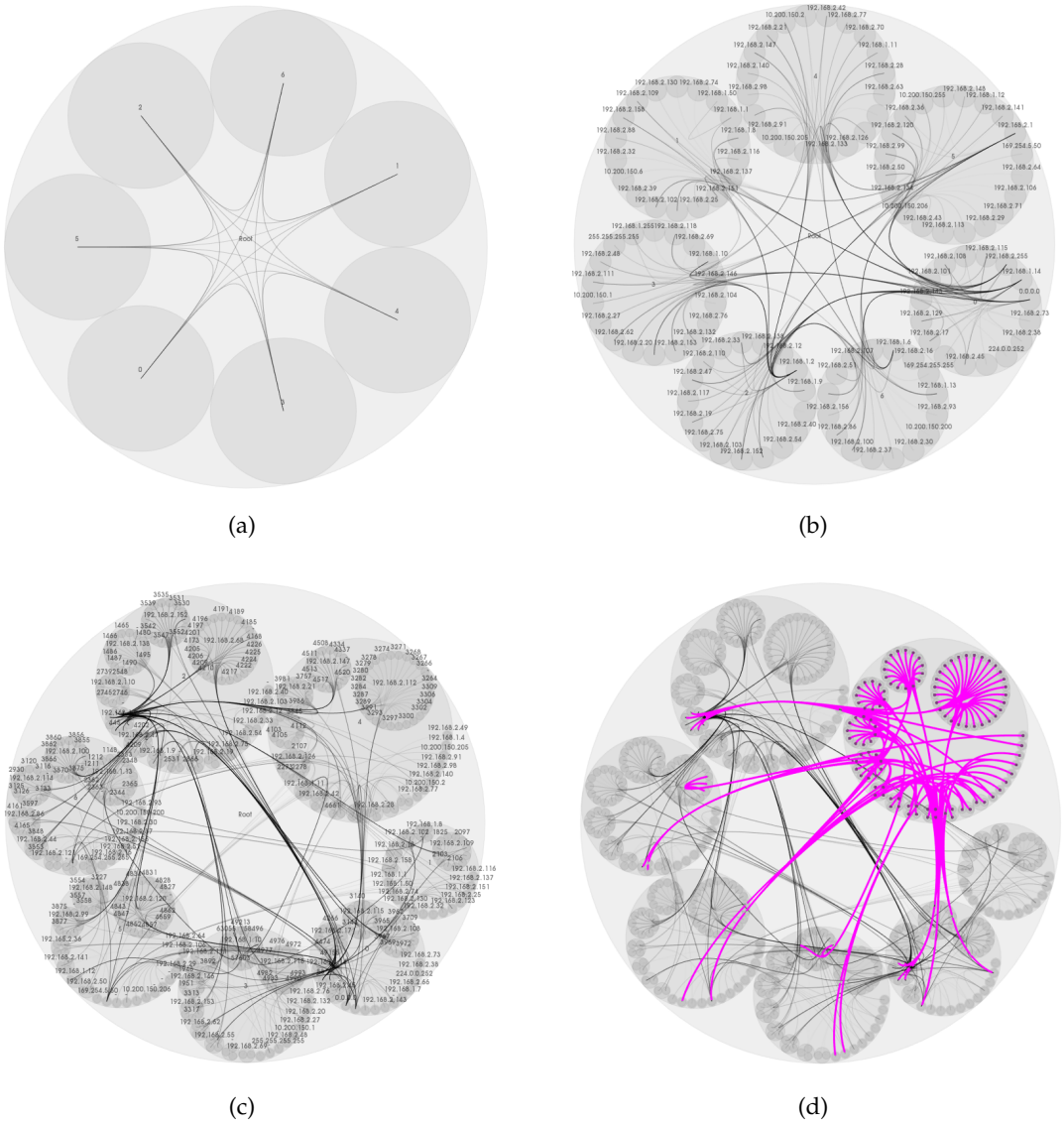


FIGURE 2.1: Schematic comparison of the physical and logical paths for the same transmission within a SDN. The physical path goes directly from the source port to the destination port, while the logical path ascends the hierarchy from the source port up to the root note and then descends down to the destination port.

So the logical path of a transmitted packet follows this hierarchical tree by ascending from the sending port, IP and control node and by descending to the receiving control node, IP and port (see Figure 2.1). In contrast to that, the real physical path of the transmission is only between physical nodes/ports. Still, it is beneficial to understand the logical processes and paths, and to examine the hierarchical nature of the underlying logic. To achieve this, a suitable visual representation like shown in the following is required.

This chapter provides a visualization for an overview of a SDN. In this overview all active nodes in the selected time window are visualized. An active node is a leaf node of the SDN tree that is receiving or sending a network packet within the given time window. To visualize all active nodes in the SDN and the induced hierarchy, the active nodes in the SDN hierarchy are drawn recursively. Here, each node is surrounded by its child nodes in a circular manner as shown in Figure 2.2.

As only active nodes are shown in the network overview, there exist at least one



(a) (b)

(c) (d)

FIGURE 2.2: Overview of a SDN simulation for a selected time window. Images (a)-(c): All active nodes in the selected time window are visualized according to different levels of granularity for the SDN hierarchy. A connecting spline is drawn between each sending and receiving node. Image (d): Visual enhancement of user selected nodes and links by highlighting.

sending or receiving event for these nodes in the user selected time window. Therefore, sent packets are visualized by a spline connecting the involved active nodes. The displayed logical path of a packet passes through the different hierarchy levels of the SDN. All nodes that are passed by a packet on its logical path are used as sampling points for the drawn spline. As a result, the logical path of a packet through the hierarchy of the SDN is visually encoded. In order to avoid visual clutter, the splines are summarized through a hierarchical edge bundling approach [39]. This

results in a fast and intuitive visualization of packets and their way through the SDN hierarchy.

It is possible to select groups of nodes or connections in the overview of the SDN as shown by a magenta color in Figure 2.2 (right image). Here, the selected elements are visually highlighted. Selecting a node that is not a leaf node in the SDN hierarchy results in selecting the whole sub-tree of this inner node. Also, different detail levels of a SDN can be shown. For example, it is possible to only visualize the root node and the control nodes of a SDN. Therefore, it is possible to focus on specific aspects of the network.

Although this gives a suitable first overview of nodes and their connections in a SDN, it is important to understand the amount of incoming and outgoing traffic for the considered nodes. Also, a tracking of packets to find common source or destination nodes as well as the identification of equal or diverging physical or logical paths is desired. Therefore, the presented overview is extended by a flow tracking view in the following.

2.3.2 Flow Tracking

The packet flow in a network is an important feature that helps analysts to determine the quality of the used SDN settings. Although the presented overview of the SDN is a suitable starting point to understand a SDN's hierarchy, the highlighting in Figure 2.2 cannot distinguish between incoming and outgoing packets of a node. Also, a selected edge between nodes only shows a small part of the path that packets passing through this selected connection travel.

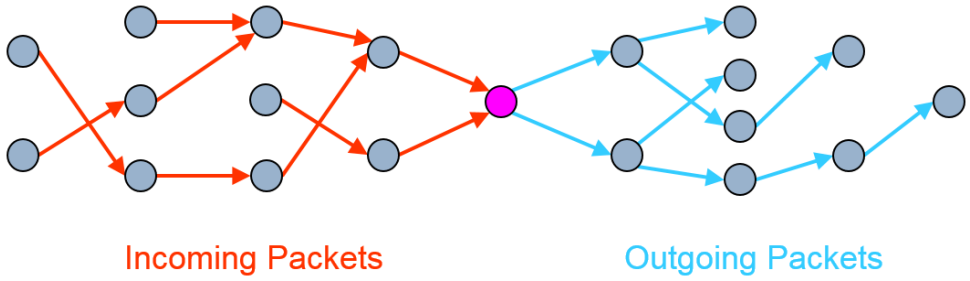


FIGURE 2.3: Schematic tracking of packets traveling through a selected node (magenta), color-coded as incoming (red) or outgoing (blue).

To address these challenges, the overview can be extended by using different highlighting colors for incoming (red) and outgoing (blue) packets, as seen in Figure 2.5. Packets related to the selected SDN elements (nodes or edges) can be tracked through their content ID. For a specific packet, a selected node or edge is passed at a certain point in time. Other nodes or connections in the SDN are passed at an earlier (red) or later (blue) point in time by this packet (see Figure 2.3). This results in a visually guided flow tracking of packets through the SDN. Arrows, color gradients or diverging colors can further improve an intuitive visualization while the line width could be used to display the amount of accumulated network traffic.

Although this improves the highlighting in the overview, the disability to directly compare the path of packets remain an unsolved problem. As shown in Figure 2.2, distinguishing single edges becomes challenging due to visual clutter when an increasing number of edges is selected. This is caused by the effect of an increasing number of highlighted edges intersecting each other in the view.

To tackle the mentioned problems, the presented system presents an additional view shown in Figure 2.5 that displays the selected SDN elements (magenta color) and the network packet traffic passing through them. This flow tracking view is inspired by subway maps. In subway maps, different stations of a train are sorted by their destination order, and their visual representation is connected through a spline. As subway maps form a successful visualization to understand the differences and similarities of multiple subway lines, the goal is to apply this concept to packet flows in SDN data, as shown in Figure 2.4. In the case of a SDN a “train" is a packet and its “stations" are the nodes the packet visits in the SDN. To integrate the different time steps selected in the current time window, time is used as the x-axis of the visualization.

In this visualization each active leaf node obtains a horizontal line that is aligned in parallel to the x-axis. To incorporate the hierarchy of the SDN into the visualization, the line representation of these active leaf nodes are sorted by their logic within the SDN. For each inner active node a box is drawn that includes its active child nodes, meaning that the whole active sub-tree of the node is included. This is done for all active nodes and all hierarchy levels recursively. For each selected node, all related sent and received packets are traced over the user defined time window. This

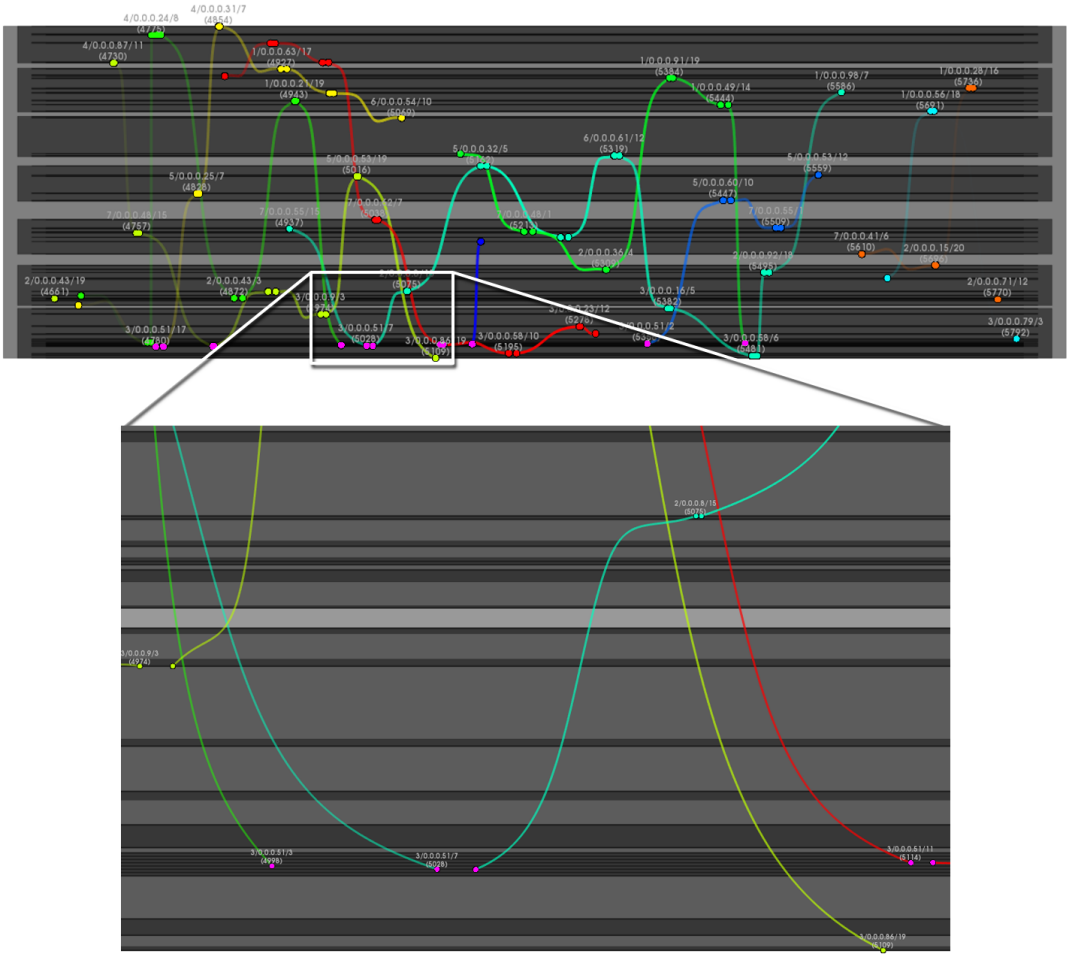


FIGURE 2.4: High contrast version of a flow tracking view for selected SDN nodes (magenta) and their links. Time is used as the x-axis and each active node is represented by a horizontal line. Packets that flow through the SDN are represented by edge-bundled and colored splines connecting the visited nodes. The hierarchy of the SDN is visually preserved by the boxes in the background containing each other hierarchically. The closeup clearly shows that even the ports as the lowest level of the hierarchy are represented.

is achieved by tracking the individual packet's content ID, thereby identifying different packets representing the same content. So in general, for each packet's content passing through a selected element in the SDN, the flow graph of visited nodes is displayed.

To visualize the tracked packets, the path of each packet is visualized by a spline connecting all nodes a packet is visiting. Circles are used to mark the points in time when a packet visits a networking node. A hierarchical edge bundling approach [41] is used to bundle edges with regard to the SDN's hierarchy to avoid visual clutter and identify coherent paths for multiple packets. As before in the overview

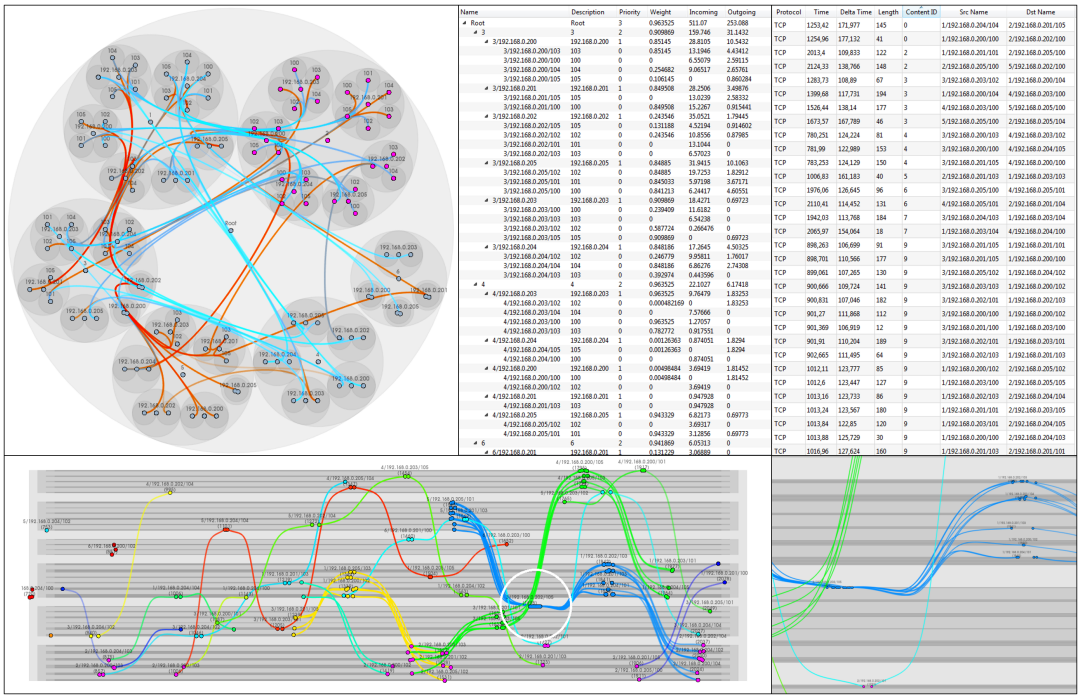


FIGURE 2.5: Application of the presented system to a simulated SDN dataset. Upper left: Overview of the SDN hierarchy. Nodes selected by the user are highlighted. Red and blue lines indicate incoming and outgoing packets, respectively. Upper middle: Tree view of active nodes and their properties. Upper right: List of all tracked packets and their properties indicated by the user's selection. Lower left: Flow view of the tracked packets. Lower right: Closeup (white circle) of the bottleneck identified by the flow tracking view.

visualization, transparency is used to blend out tracked packets at the borders of the given time window. Each spline is colored by a unique color depending on the content ID of the respective packet to distinguish individual packets and flows in the visualization.

This visualization provided the ability to monitor user selected elements within the SDN and to track individual networking packets over different hierarchy levels in the SDN. Low level physical flows in addition to high level logical flows can be tracked and analyzed and coherent flows and bottlenecks in the SDN design can be identified.

2.4 Results

The presented system was tested with a simulated SDN dataset. As mentioned in Section 2.1, the concept of SDN is quite novel and not yet applied in many scenarios.

Resulting from that, suitable real world datasets are not widely available. Instead, this chapter uses a simulated SDN dataset generated by the method presented by Nandi [80] to gain insight and show the effectiveness of the presented visualization methods for SDN.

The results in Figure 2.5 show how the presented system can be used to visualize and analyze SDNs. The overview shows the active nodes in the selected time window of the dataset and a highlighting of nodes that are selected by the user (upper left image). In addition to that, the flow through these nodes is tracked and shown by the a red/blue highlighting. Based on the user's selection, the all views are updated. The tree view (upper middle image) shows the SDN hierarchy and all active networking nodes with their accumulated properties. All packets involved in the flow tracking are shown in the tabular view (upper right image). These views help in analyzing and configuring the SDN.

The flow tracking view (lower left image) presents a static visualization of packet paths. Although the source and destination of some packets (yellow splines) vary in this example, the view indicates that these packets all follow the same logical path. Through the intuitive design of the flow tracking view, the common control nodes can easily be detected. Furthermore, the visualization is able to visually highlight bottlenecks in the SDN design. As Figure 2.5 (lower left image) demonstrates, a larger number of packets (blue splines) is sent through only one networking node (white circle). The lower right image shows a closeup of this feature where individual port nodes/boxes become visible. Therefore, this visualization helps identifying security issues or unequal packet distributions in the SDN design through this visually guided flow tracking.

The overview of the system forms an easy to understand visualization that represents the hierarchy of the SDN. Pairs of sending and receiving nodes are connected through a spline that indicates a packet's flow through the SDN hierarchy. This representation allows to easily understand the connection between nodes in the SDN hierarchy. The used hierarchical edge bundling avoids clutter while giving the user a suitable overview of the SDN hierarchy and its packet flow. Focusing, highlighting or color shifting can help in identifying individual paths and countervail obfuscation.

For a further investigation of user selected nodes and packet paths, the flow tracking view is presented. The use of the subway map methodology allows a visually guided tracking of packet paths through the SDN. The box representation with connecting splines provides the possibility to track packets throughout the network and therefore visually highlight common paths in different hierarchy levels. Using small circles to represent networking nodes being visited at certain points in time prevent the recognition of false intersections. As all time steps in the selected time window are visualized simultaneously, the user is not confronted with change blindness problems that often occur in dynamic visualizations. Also, the used edge bundling enables the identification of similar packet paths an helps network analysts in adjusting their configuration design.

The presented approach is highly interactive as the user can select nodes or edges in each presented view while the remaining views are updated accordingly. So all views are interlinked dynamically. In its entirety, the presented system offers a suitable solution to analyze SDN data, allow a visually guided tracking of packet flows and therefore assist network analysts in identifying problematic settings in the SDN.

2.5 Conclusion

The SDN technology is an immature but upcoming networking technique that requires further investigation through simulations by network analysts to become applicable in real world scenarios. Therefore, this chapter presents a linked view system to allow network analysts to review SDNs, their hierarchies and the flow of packets within. The latter is known to be an important factor for the quality of SDN settings. The presented system is capable of of visualizing a SDN's hierarchy and the packet flow through this hierarchy. For further investigation of the flow, the system contains a flow view to analyze and compare the paths of different packets, find coherences and identify weak spots in the SDN design.

Chapter 3

Visual Analysis of Cyber-Physical Production Systems

3.1 Introduction

The "industrial internet" refers to developments triggered by new information and communication technologies (ICT) in industry. Terms like Industry 4.0, Cyber Physical Production Systems (CPPS) or Smart Factory are associated with the same phenomenon of industrial systems using ICT in production by applying cyber-physical methodologies. The most prominent feature of a CPPS is the interconnection of its different manufacturing elements such as machine tools or workpieces and the resulting amount of production data [79].

It is expected, that these technologies will make it possible to handle the increasing complexity of production systems and to cope with current trends and challenges [53]. Shorter product life cycles and mass customization are leading to higher numbers of product variants and ever smaller lot sizes, making it necessary to adapt a production system in a fast, efficient way.

With the goal of increasing the flexibility of a production system, decentralized concepts within production planning and control have been discussed in theory [65]. Yet, their application in industry has not been widely spread [78]. The availability of modern ICT, as discussed under the term of "Industrial Internet", is expected to have a highly positive effect on the applicability of such concepts [53].

The enormous amount of data generated by cyber-physical systems makes it necessary to devise approaches for refinement, for the data to become truly helpful

to human decision-makers. Also, to identify ways to make machines "intelligent", data analysis and visualization tools are becoming essential. Hence, the effort of this chapter was driven to a large degree by developing new and more effective ways to analyze and visualize production data, allowing humans to have quick access to just the needed, most relevant information.

This chapter is based on the work of Post et al. [92] and introduces a visual analysis approach that captures the performance of a production system in an intuitive manner. For example, bottlenecks and excess capacities are identified and visually highlighted, thereby guiding the user in the analysis. The impact of changes applied to a manufacturing system can be analyzed by utilizing the presented visualizations. Additionally, the approach can generate an aggregated view of an entire system or focus on merely the most interesting features captured in a data set. The developed tool supports a performance driven and yet detailed analysis by enabling a highly efficient evaluation of production data and by guiding a user to ask important questions. An exemplary production system is simulated to depict the characteristics of the visualizations and to show the applicability and effectiveness of the presented analysis tool.

This chapter is structured as follows: The next section provides an overview of related work and defines requirements for data analysis and visualization tools for complex production systems. Section 3.3 describes the underlying simulated production system and focuses on the methodologies used in the presented analysis and visualization tool. Finally, this chapter is concluded by summarizing the main contributions in Section 3.4.

3.2 Related Work

This section will summarize related work in data analysis and visualization tools for cyber-physical production systems and derive the resulting requirements for a visual analysis tool.

3.2.1 Data Analysis in Cyber-Physical Production Systems

The data available in a cyber-physical production system can be used to make production systems more flexible. In this context, flexibility can be understood, on the one hand, as the transformability of the system to engineering changes on medium or long term perspective. On the other hand, flexibility can be understood as achieved by decentralized production control on a short term view.

It is obvious, that the vast amount of data is not useable without refinement. Therefore, user friendly tools for data analysis and visualization are needed [106].

Examples for engineering changes are the reconfiguration, addition, substitution or removal of production equipment, e.g. machine tools, in a manufacturing system [99]. They usually have extensive impacts on the manufacturing system due to the manifold interrelationships among production objects [73] and hence need careful analysis and planning before implementation. The change in one element might result in the disruption of the process chains, material flow or information flow. Therefore, tools are required that can analyze the effects of envisaged engineering changes in a fast and comprehensive manner [72].

Tools that support the planning and analysis of changes in manufacturing systems can be found within the concept of the digital factory [20]. Simulation and evaluation software for products and material flows can be applied in order to analyze processes and their changes. However, such software tools require specific know-how and qualified personnel to use them and to keep them up to date, and are not specialized on engineering changes [102].

A framework specialized for analyzing impacts of engineering changes to existing manufacturing systems is proposed by Malak et al. [72]. Here, the alternative solutions for engineering changes are visualized in a 3D virtual environment where effects on factory layout and material flow can be seen in a spatial context. Although a three-dimensional virtual environment displays information intuitively, and thus gives a realistic feeling of the modeled factory, it shows only partial views of the factory and does not guide the user to the information needed.

For the fast and effective analysis of impacts to engineering changes, both, aggregated and detailed views, are necessary. To enable the overall evaluation of the

given situation, e.g. to examine process chain and information flow consistency, the available data needs to be visualized in an aggregated manner. On the other hand, scalability of the data is required to allow the user to focus on single products or machines, and to well defined time steps of special interest. An essential requirement is to guide the user to the most interesting features of the regarded system, and to show critical issues. Therefore, comparative and interactive data highlighting integrated in the spatial context of the factory is needed. Different perspectives focusing on machines, products and material flows within the visualization tool need to be distinguished and interlinked in an interactive manner.

In contrast to the planning of engineering changes, decisions in production control need to be taken in real-time with limited information. The concept of self-control in a decentralized production system is based on the ability of several elements of the system (e.g. machine tools or work pieces) to act and decide autonomously. In contrast to that, in the centralized approach, planning is accomplished by a superordinated planning entity. Therefore, especially for decentralized production control with a multitude of decision makers, a fast recognition of data patterns is necessary to adapt the behavior and decision rules of the acting elements.

The applicability of different self-organization concepts is tested in several research projects by using prototype factories (e.g. [15, 120, 66]). As the implementation of such prototypes with real machinery involves considerable effort and expenses, they are therefore not meant for real scale experiments. Thus, the amount and complexity of data can still be managed manually, so tools for visual evaluation and optimization of the concepts are missing. Ilsen et al. propose a test field based on a multi-agent system to test several self-organization concepts against each other in a real sized but virtual environment [48]. Here, several different decision routines, e.g. for machine tool selection or production order, are possible. This case shows, that in the analysis of decentrally controlled production data, the impacts of different decision routines need to be visualized. Further, there is a need to identify patterns on an aggregated data level to derive the system's sensitivity to changes of decision routines. As a consequence, aggregated views displaying the overall performance in a spatial context, and detailed views representing the perspective of single elements, are necessary to understand an entire system.

To summarize, one major issue for data visualization is to be intuitively understandable. Therefore, an interactive guidance for the user is required, which makes it easy to find interesting features in a data sets. To get a quick but comprehensive overview of the status of the production system, different perspectives on an aggregated level are needed. These have to be interlinked to navigate through the perspectives. Beside the aggregated views, scalability is a further required functionality that enables to select single hotspots and establish detailed comparisons between machines, products or time steps. Embedding the data into the spatial context of the factory is needed to give the user a realistic and intuitive understanding of the factory and its performance.

3.2.2 Visualization Tools in Production Planning

Various approaches [87, 95, 70, 110] present tools to visualize the performance of a production system. They mainly consist of strategies on stacking or combining single visualization entities, as workload or production time graphs for different machines or points in time. The main disadvantage of these approaches is, that they do not provide spatial context or user guidance, which was identified to be essential for data analysis in cyber-physical production systems.

The visualization of product flows is an important task in the field of production planning. Embedding such flows into an underlying geometric model was already used in other domains like economical trade visualizations [111] and urban traffic visualizations [100]. There, money transfers are embedded into geographical maps and vehicle flows are embedded into traffic maps, respectively. The presented work makes use of this approach and transfers it to the domain of cyber-physical production systems by embedding product flows into virtual factories.

Wu and Acharya [116] present an approach to visualize the workload of a machine with a stacked box representation of the products waiting for this specific machine. Although this representation shows the order and number of waiting products, the approach does not provide a spatial context and only takes a certain point in time into account. Therefore, this chapter extends the approach of Wu and Acharya by applying it to a whole time window and embedding it into a spatial context for all machines at once.

Doil et al. [19] present an exploration system based on a virtual reality environment that allows users to choose different manufacturing settings. Because this is an intuitive way to handle manufacturing settings, this chapter also provides a three-dimensional representation of the factory that is compatible with a virtual reality environment. In addition to the approach of Doil et al., important aspects for factory planning like product flows or machine workloads are visualized directly in this spatial context.

The visualization of production times is handled in the work of Zhang [119]. Their approach presents a comparative visualization of production times where the user can examine the production times under certain conditions. Although that is a good representation of the production times, their scatterplot alignment makes it hard to compare different products. In contrast to that, this chapter visually captures the development of manufacturing time against degree of completion. Here, all products are aligned consistently, enhancing the user to directly compare production times.

Ertek et al. [24] visualize statistical features of production times for different manufacturing settings. Their approach can compare production times resulting from these settings, but no user guidance to find good settings is provided. In addition to that, the visualizations presented in the current work highlight interesting features and thereby guide the user in the analysis of manufacturing settings.

Based on the derived requirements and the found issues in production planning, this chapter shows the development of a visual analysis tool for virtual manufacturing systems.

3.3 Methods

The following sections will describe the production system that is simulated to obtain the data used for analysis, and the methodologies to visually analyze this data and guide the user to interesting features.

3.3.1 Characterization of the Simulated Production System

In a first step, a simulation is used to acquire production data of a virtual factory. All of the factory's components like the machine tools and the product workpieces are virtual as well. Each machine has a machine type and each product a product type, so there are different types of virtual products and machines.

The work plan, i.e. the order of operations required to produce a final product, is given externally and can not be changed (see Table 3.1). Thus, the product type defines which operations need to be processed sequentially to finish the product, while the machine type defines the operation the machine is capable of.

Product Type	Operation Number	Production Technology	Process Time [min/pc]
A	1	Milling	52
	2	Milling	65.3
	3	Drilling	200
	4	Tapping	211.1
B	1	Milling	83.3
	2	Turning	163.9
	3	Drilling	100
	4	Tapping	88.9
	5	Turning	16.4
C	1	Turning	185.8
	2	Drilling	300
	3	Turning	142.1

TABLE 3.1: Work plan describing the individual operations performed for each product type and their according processing times.

The ability of machines of a certain type to perform an operation with specific requirements is encoded into the machine type. For example, if the machines of a certain type are able to drill holes, but their accuracy cannot be guaranteed to be high enough for a certain operation, their machine type marks this operation as not performable. Though, other machines might be able to perform this operation with the required accuracy. Thus, the technological capabilities are encoded into the machine types.

The material removal rate (MRR) may vary during operations depending on material type, cutting speed and depth, cutting aids, tool type or other factors. This leads to different process times, even for operations with the same production technology. The required set-up times for each operation are included in the resulting processing times that are given in Table 3.1.

To finish the production of a product, all of its operations need to be processed in order, while each operation takes a certain time. Since in the presented example no machine is capable of performing all operations, the products have to be processed on different machines sequentially. So the current operation of a product is first finished on one machine, and then the product is transported to another machine. Since this new machine might be busy, the product is enqueued. To do so, each machine has a queue of waiting products that are processed in order of arrival (first in - first out).

If there are different machines of the same type, the question arises, which of these machines should process a certain product. This question can not be answered in a perfectly optimal way for real-life sized problems due to its high computational complexity. So an optimal solution can not be calculated in a feasible time, but heuristics can be evaluated efficiently to come close to an optimal solution of product distributions. This chapter uses the heuristic of always choosing the machine that will have processed the product's individual operation first. Other heuristics could be considered as well, but since the choice of heuristic is not important for the demonstration of the presented analysis tool, the described heuristic is chosen out of simplicity reasons.

Another issue is the optimal arrangement of machines in the factory. This problem is also computationally very expensive and cannot be solved optimally in a feasible time for a larger number of machines. Therefore, the arrangement of machines in the presented example was chosen as demonstrated in the next section by simply distributing groups of identical machines within the factory.

The transportation times of products between the machines depend highly on the arrangement of those machines. Since the production batches in the used example are very large, the resulting transportation times are very small in comparison. Therefore, the transportation times are visually disappearing in this example. Still, the methodologies that are presented in the next section are easily extendable in a straightforward way to also visually include transportation times, as will be seen shortly. In the presented example, a free transportation model is used. Naturally, other simulations could use restricted transportation routes to implement conveyor belts or other transportation methods.

The production data used for the analysis describes which product and which operation is performed on which machine at which point in time. To acquire this data, each product type is virtually produced 30 times in a simulation, while starting with a product of type A, then B and C, and then repeating this loop 30 times with a temporal gap of 10 minutes in between the products. This means, a new product of a specific type starts its virtual manufacturing every 30 minutes. To analyze the gathered production data, the visualizations and methodologies described in the following sections are used.

3.3.2 Visualization

Based on the requirements developed in Section 3.2, a tool for the user-guided visual analysis of simulated production data was designed as described below. The tool is a linked view system, visualizing the manufacturing process under different aspects. This means, that there are different views, each showing the same data but having a focus on different aspects. The presented tool contains a flow view, a workload view and a production view.

Additionally, the views are interlinked by transferring user interactions like selection and highlighting of products, product types or machines from one view to all views. Another user interaction is to choose a time window by manipulating a point in time and an interval size in all views. Then, this time window will be considered for visualization. This enables the user to zoom in and out onto certain interesting points in time.

All of the views of the system only show the data that occurs during this chosen time window, thereby treating this window consistently for all views. By doing so, the user is enabled to focus on certain features while the overall picture is preserved. This helps the user in building a mental map of the production data. Since all views of the presented system always show data for the same time window or selection, a cognitive transition from one view to the others is straightforward.

After the virtual manufacturing system is simulated once, the whole tool and its views work in real-time to provide flexibility of interactions to the user. The presented tool can be used to analyze virtual factories, provide user guidance for later optimization or comparison, and help in decision making.

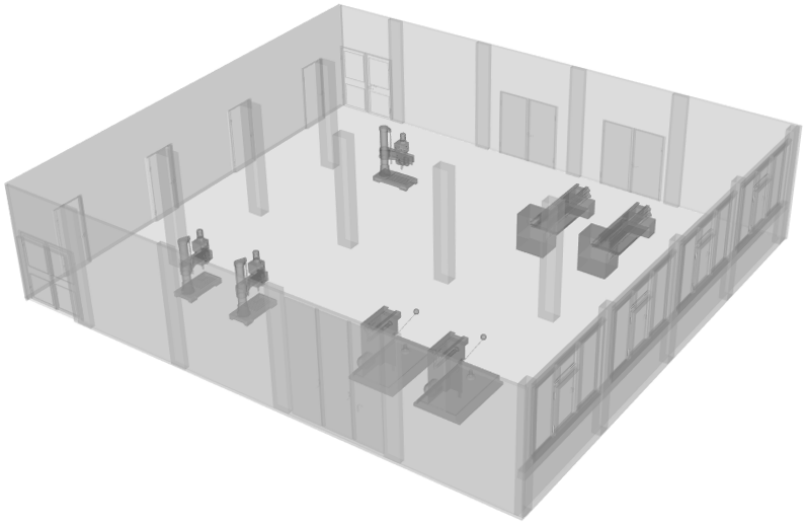


FIGURE 3.1: Visual representation of the factory model and its machines. Here, the setup contains two milling machines, two turning machines, one drilling machine and two tapping machines.

The overall goal is the optimization of the production process in the virtual factory (see Figure 3.1) with respect to a diversity of parameters. Still, this optimization cannot be done fully automatically because of its high computational complexity. This stresses the importance of the presented tool to support users in their analysis tasks. Although the optimal solution is unknown to users, the presented tool can be used to iteratively improve factory settings. By that, users are enabled to approximate an optimal solution, thereby finding a sufficient solution and gain a certain confidence in their production process.

The realization for each view of the tool is presented in the following sections.

Flow View

The flow view provides the most general and intuitive visualization and helps creating an overview of the virtual manufacturing system by containing different visual elements (see Figure 3.2).

The geometric model of the factory and its machines is displayed to form a basis for an embedding into a three-dimensional virtual context. To reduce the visual occlusion of other graphical elements, semi-transparency is used for the geometric model. Using a three-dimensional model of a factory as a spatial context is a familiar working environment for analysts and domain experts who are used to these kinds

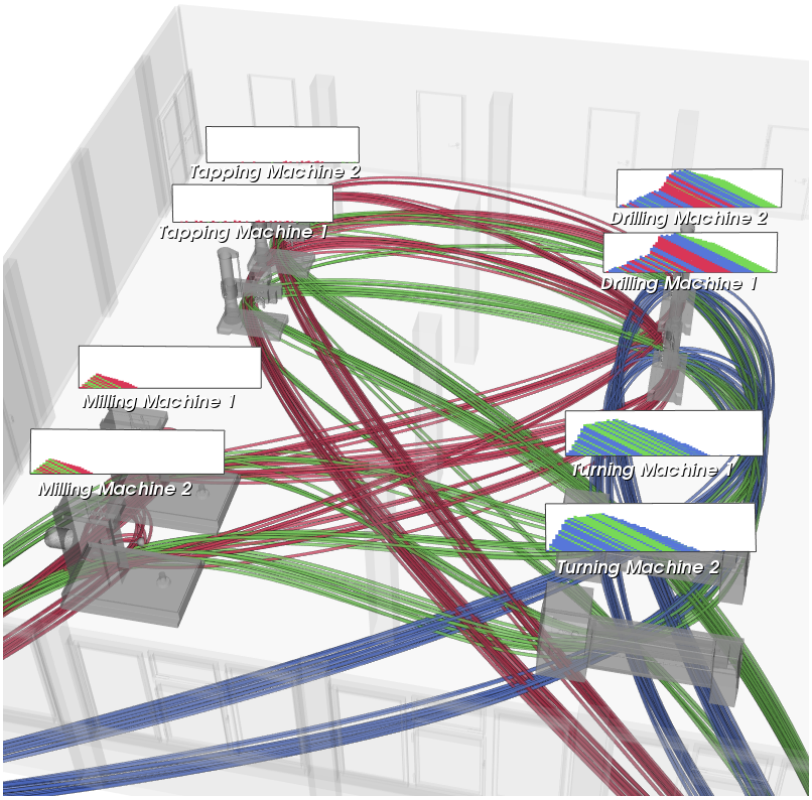


FIGURE 3.2: Flow view of a virtual production system showing the geometric model of the factory and its machines, the product flow for all products color-coded by product type and the machine workloads for all machines.

of visualizations and interactions. Also, due to its simplicity and intuitiveness, this embedding is suitable for presentations.

The product flow extends this natural environment to allow the tracking of products throughout the factory as described later. This results in a direct visual feedback on the emergent behavior of the products in combination with the used heuristic method for distributing products onto machines. A user can see what machines individual products were processed on, and how the products are distributed.

In addition to the visualization of the product flow, the presented view contains a visualization for machine workloads. Here, the workload for each machine is displayed by showing the machine's queue for several points in time. These graphical elements are attached to the individual machines, leading to a direct visual feedback on the performance of machines and their queues. Also, the visual embedding into a spatial context supports a higher degree of intuitive understanding.

Combining multiple elements in one view results in a tool that is capable of analyzing factory arrangements and mechanisms. The realization of the product flow and the machine workload is shown in the following.

Product Flow: The product flow is the combination of the trajectories formed from all the products moving throughout the factory over time. This is demonstrated in Figure 3.2, while the images in Figure 3.3 show a more detailed look on the properties of the product flow.

A spline [26] is a piecewise polynomial function of a fixed degree. It can smoothly interpolate a sequence of points without fluctuating too much, so it is well suited to follow a trajectory. These splines are used to represent the product flow by visualizing a spline segment for each moving product.

Instead of visualizing the "real" transportation routes between machines, the spline representations show the topological routes of the products, meaning the order in which machines are visited. If real transportation routes were provided as a model or restrictions in the transportation were known, these "real" paths could be used instead.

The supporting points of a spline are set to the machine positions that perform the respective production steps of a product, with an additional random offset to avoid visual clutter among multiple splines. The smoothness of the splines helps the user to follow the progress of individual products. The splines are visualized using Gouraud shaded [37] tubes. This means, that small tubes are used for their geometric models, and that these geometries are lit by a light source to enhance the natural understanding of their shape and progress. Additionally, alpha-blending the spline representations help making them halfway transparent to further distinguish individual spline tubes while preserving an overall picture of all spines. Figure 3.4 shows a closeup of both milling machines from Figure 3.2 that demonstrates these visual effects.

Like all other data mining and visual analysis approaches, this method has its limits. Large numbers of products or product types can lead to visual clutter and increased confusion. Fortunately, for the chosen spline representation there exist methods like attribute-driven edge bundling [84] or hierarchical clustering of flow

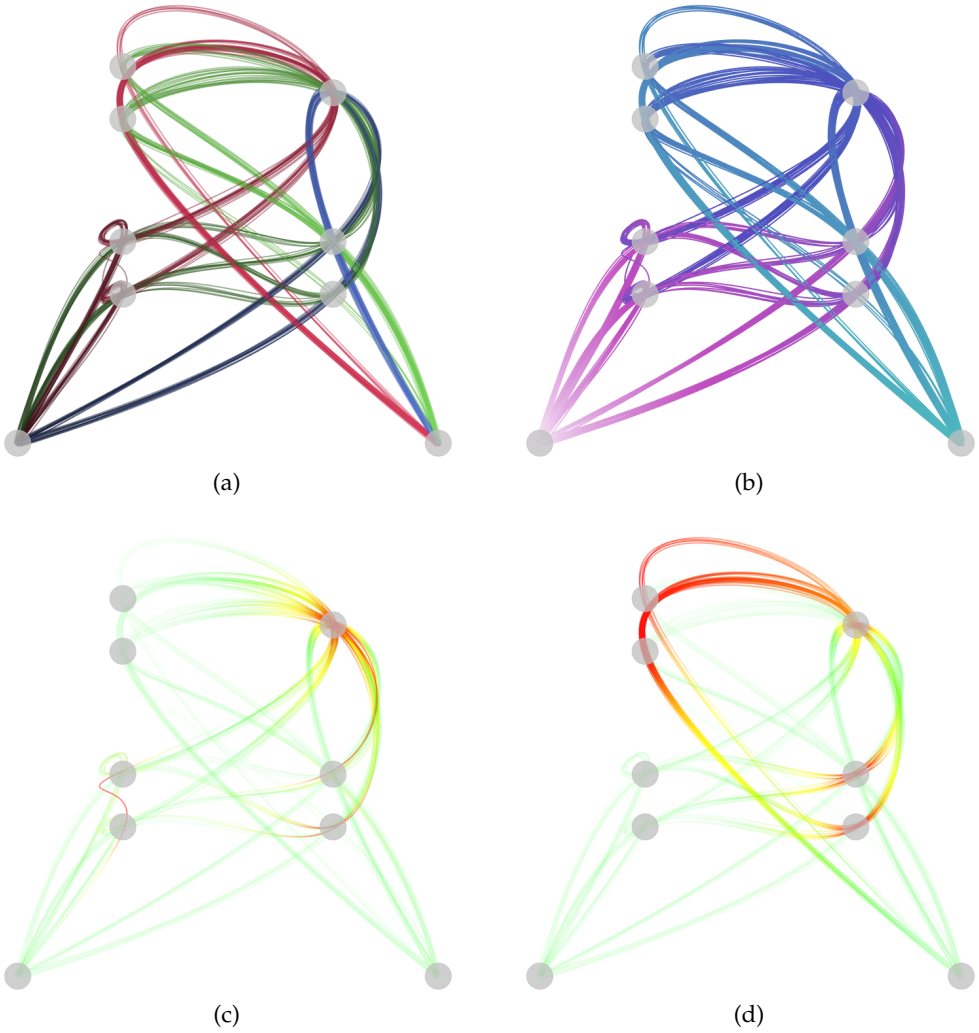


FIGURE 3.3: Detailed views of the product flow demonstrating the visualization of different properties: (a) product types color-coded, (b) product completion ratios ranging from white over magenta and blue to cyan (compare Figure 3.8), (c) relative waiting times and (d) relative processing times per product ranging from green over yellow to red.

maps [85] to compensate or mitigate those effects. In addition to that, general filtering and data aggregation techniques can be used to analyze and compare facets of interest in the production data.

By utilizing these visualizations, an intuitive tool for the examination of properties in the product flow is introduced. By color-coding different properties, the user is guided to interesting events and locations in the simulated production system. Figure 3.3 shows four examples for different properties. In these visualizations, the lower left dot represents the factory's entrance position for all products, while the lower right dot is the exit position when the products are fully manufactured. Also,

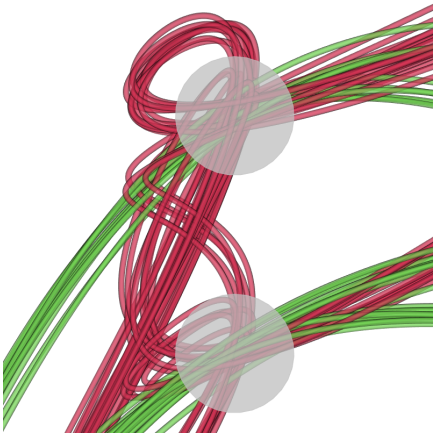


FIGURE 3.4: Closeup of the product flows demonstrating different methods to reduce visual clutter.

for demonstration purposes, the images only feature one drilling machine (top right dot in each image).

Figure 3.3(a) shows the overall product flow for three different types of products. Here, each spline has a color given by the respective product's type, coded as red, green and blue. This enables the user to follow and distinguish products by their specific type. Also, the user can see how products of the same type are split up and distributed onto different machines for certain production operations. In the example in Figure 3.3(a) this directly generates inside into the unequal distribution of blue products.

Another alternative to showing product types in the product flow is to visually encode the waiting or processing times of products. In Figure 3.3(c) the relative waiting time of an operation (in respect to the mean waiting time of all operations of the respective product type) is coded as color. Here, the colors range from green over yellow to red. For a product of a certain type, green means, that an operation has a low waiting time compared to the other operations of the same product type. Yellow represents an average waiting time, while red encodes a high waiting time. This highly intuitive visualization method guides the user to machines where products have to wait longer than average before being processed, like the single drilling machine in this example.

Besides the presented properties, further attributes like production time can be visualized. By mapping the values of such an attribute directly to colors, bottlenecks of production steps can be seen for the products individually. Mapping relative

attribute values, i.e. values set in comparison to the values of all other products of the same type, extends this mechanism and enables the user to visually analyze bottlenecks for an entire product type.

Although this is a powerful tool to locate weaknesses in the factory design, this flow visualization can not show single product locations and their waiting positions. Therefore, an extension of the overall flow visualization is required.

This issue is tackled by restricting the overall product flow visualization to the user defined time window. Here, only the positions of a product within this temporal window are displayed. A product is moving within the time window defined by the user, resulting in a path segment of a certain length. If a product moves slowly caused by longer transportation, waiting or production times, the respective path segment will become shorter. In contrast, if a product moves faster, the respective path segment will become longer, since the product travels a longer distance within the given time interval. Figure 3.5 shows an example for equally large time windows moving forward in time (left to right image), thereby resulting in products moving through the virtual factory. The user is enabled to see that the overall flow slows down with increasing time, since the single drilling machine in the top-right corner of each frame is a bottleneck.

To determine the position of a product moving from a machine A to a machine B for a specific time, the route between machine A and B is divided into three parts. The first third of this route represents the operation of the product being processed at machine A. The middle third represents the transportation of the product from machine A to machine B. The last third of the route represents the time the product waits in queue to be processed by machine B. So if the product just finished its waiting period and starts being produced by machine A, the position of this product is the position of machine A. If the product just finished its production period at machine A and starts traveling to machine B, the position of this product is one third of the path from machine A to B. If the product reaches machine B and starts waiting to be processed by machine B, the position of this product is two thirds of the path from machine A to B. And if the product finished its waiting period and starts being produced by machine B, the product's position is the position of machine B.

The advantage of linearly interpolating the position for transportation, waiting

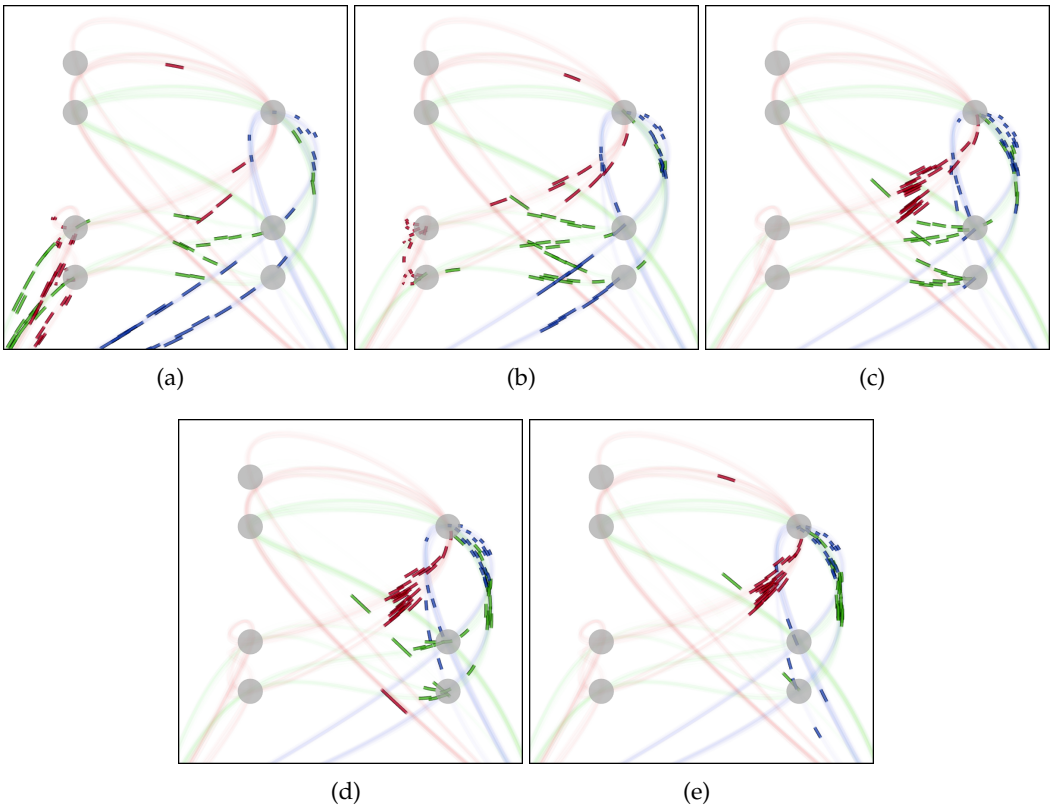


FIGURE 3.5: Sequence of product flows for a short temporal window moving forward in time (left to right image), resulting in products moving through the virtual factory.

and production phases is, that a product's position advances when the product's status is increased. This is the case when either the product's transportation advances, the remaining time the product has to wait in a queue decreases, or when the product's degree of completion increases during production. By that – in addition to spatial processes like transportation – temporal processes like waiting or being produced are transferred into motion, thereby achieving a high level of intuitiveness.

This method forms a visual encoding of the product flow and its efficiency, and allows the user to follow single products over the whole production time. Additionally, it is possible to identify machines with bottlenecks resulting in a high production time. Furthermore, the user can see the location of the products of different types and the amount of products that are produced within the concerned time window. Also, choosing the time window to be the whole simulated time interval results in Figure 3.3(a).

Although this is a powerful visualization, it is hard to identify the order in which

multiple products line up in queue to wait for a certain machine. This is overcome by the visualization of the machine workload.

Machine Workload: To tackle the problem of requiring additional insight into the waiting queue of a machine, the presented approach embeds a suitable visualization for each machine's workload into the three-dimensional factory model (see Figure 3.2). By doing that, the spatial context of the virtual factory model is preserved and hence the intuitiveness is increased.

For each point in time of the time window defined by the user, all products in the waiting queue of a machine are shown in their unique product type's color. For consistency reasons, this is the same color as used in the flow view.

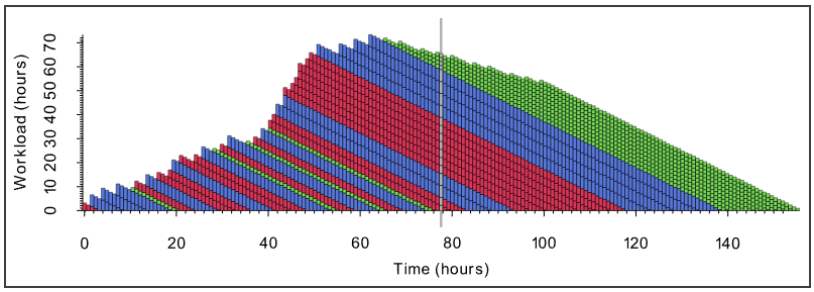
The waiting products are visualized as stacked boxes. Each box has a black frame to distinguish successive products of the same color, meaning the same product type. For each particular product, the height of the respective box corresponds to the remaining production time this product will need at the machine it is waiting for. The product a machine is currently working on is located at the bottom of the stack, while recently enqueued products are added at the top.

A single stack represents the waiting queue of a machine at a certain point in time. The height of the stack equals the accumulated height of all boxes, thereby visualizing the workload of the machine. Instead of limiting this visualization to a single point in time, a stack for each point in time of the user defined time window is visualized. This is consistent with the flow visualization. Additionally, the current point in time is marked for better orientation. This results in a visualization that is able to provide visual feedback on the development of a machine's workload in contrast to showing only a single point in time.

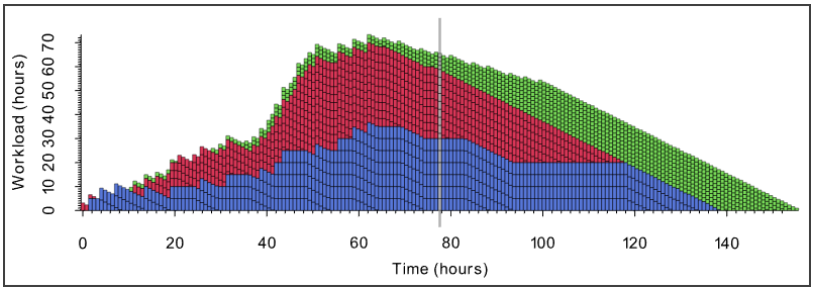
The result is a visualization for each machine's workload, representing the exact amount of waiting products and their production time and order for each specific point in time. Figure 3.6 visualizes the workload of the first drilling machine from the example shown in Figure 3.2. Figure 3.6 shows, that the products waiting in the machine's queue are quite unordered. This indicates, that rearranging the products in the machine's queue into blocks of the same product type might have a high potential in minimizing time losses due to tool changes within the machine.

When considering set-up times, this could speed up the average production time by decreasing the overall set-up time. As this example shows, the user can intuitively analyze machine queues with respect to the number or types of products, their workloads and even trends over time.

In addition to showing the real order of products in a machine's queue, the user is provided with the possibility to sort the individual stacks of the visualization by product type. By doing that, the insight into the order of the products in the queue is lost, but instead, a more direct visual feedback on the number and the accumulated production time is gained for all the products of a certain type, as the bottom image of Figure 3.6 shows.



(a)



(b)

FIGURE 3.6: Unsorted (a) and sorted (b) workload of the first drilling machine showing the development of the machine's queue with its individual products waiting to be processed.

Although the flow view is a suitable tool to review several aspects of the manufacturing process, it is not able to compare the workloads of different machines or the influence of design changes in the factory layout on production times, e.g. for identifying bottlenecks in the production. This is tackled by further views that are introduced in the following.

Workload View

The flow view lacks the ability to directly compare the workload of the machines in the virtual factory. Therefore, a workload view is provided as shown in Figure 3.7 for the example used in Section 3.3.2. This view consists of a workload graph for each machine, visualizing the different graphs positioned on top of each other. To make the workloads comparable, the same coordinate system is used for all graphs, meaning that the workloads of all the machines have the same x-coordinate for an arbitrary point in time.

Consistent to previous visualizations, not only a single workload for a single point in time is shown per machine, but the workloads for all points in time of the user-defined time window are displayed. This provides insight into the development of a machine's workload over time, and offers the possibility to temporally zoom in onto interesting features. Since the workload graphs for all machines are linked, they always remain comparable. Additionally, the current point in time is highlighted for better orientation.

Here, the focus lies on the magnitude of a machine's workload, meaning the accumulated production times of all products in the machine's queue (compare to height of graph in Figure 3.6). To not overload the visualization, only important information is included. Thus, all information about the individual products forming this workload is neglected.

The workload of a machine is shown in the foreground as the height of the respective graph. At each point in time, the highest workload of all machines is calculated and displayed in the background as a second graph with a light gray color. So the background is the same for all graphs. This helps in comparing the workload of different machines.

The color of the graph in the foreground representing a machine's actual workload ranges from white to blue, thereby encoding the ratio of the machine's workload to the globally highest workload by color. This means, that a high relative workload results in a blue color, whereas in contrast, a low relative workload results in a white color. This enables the user to relate the workload of individual machines to the globally worst workload, thereby intuitively identifying critical machines

dependent on time.

Multiple machines of the same type with a high workload may indicate a need for more machines of this type, while multiple machines of the same type with a low workload may indicate redundant machines. If both occurs for different points in time, the user may want to redistribute the workloads. Because of that, the introduced visualization is a good basis for analyzing machine workloads.

It is not necessary to optimize a machine in general, but at specific points in time, when bottlenecks occur. The presented visualization helps identifying these interesting points in time and the corresponding critical machines, which then can be investigated and optimized further. As the visualization is intentionally not embedded into the three-dimensional factory model, it also provides a good overview of machine workloads and their critical features.

Production View

A missing feature of the previous views is to examine the production times for all products of a specific type. This is done in the production view. The specific product type can be chosen by the user, or an instance of the production view can be displayed for each product type in parallel. Figure 3.8 shows the production view for product type C with its three production operations (compare Table 3.1).

The top image of Figure 3.8 shows, that each individual product is visualized as one slice. Although the manufacturing of different products starts at different points in time, they are shown aligned in the production view to ensure comparability. This allows the user to visually analyze production times.

A time against degree of completion graph is shown per product on its individual slice. This means, that the x-axis measures the time since the start of the manufacturing of the individual product. The y-axis measures the degree of completion for each product.

Since each product's slice is parallel to the xy-plane, the remaining z-axis measures the individual products themselves. As all products of the same type start their manufacturing at different points in time, the z-axis can also measure this temporal offset.

With increasing manufacturing time (x-axis), each product's degree of completion (y-axis) increases from zero (purple color) until it reaches a value of one (cyan color), meaning a fully manufactured product. The level of completion is calculated by dividing the elapsed process time through the overall process time of a product. Here, halfway completed operations are taken into account by interpolating their relative level of completion linearly.

The resulting horizontal lines that can be seen in Figure 3.8 represent periods of time in which the individual product is not processed. Instead, the product is waiting in a queue for the respective machine to start producing the next operation of this product. Several products with the same degree of completion lead to visual plateaus. This is caused by equal operations being finished for several products. Since plateaus are caused by a number of products waiting for the next operation to begin, the vanishing of a plateau indicates that there are no more waiting products for the upcoming operation. In general, transitions between neighboring plateaus represent processing phases, while the plateaus themselves represent inactive phases, i.e. transportation or waiting phases.

The two lower images in Figure 3.8 show the same data as the top image but without possible occlusion problems. Here, the lower left image clearly shows the different process phases, while the completion ratio is visualized by color analog to the top image. The begin and end of process phases are highlighted by thin black lines. It becomes clear why optimizing machine workloads and thereby inactive phases is that important for optimizing the overall manufacturing times by comparing the time intervals of process versus inactive phases.

The lower right image shows an overview of the amount of products that have exceeded a certain completion ratio after a given manufacturing time, ranging from none (white color) to all products (blue color). This visualization focuses more on the temporal trend of the products' completion, thereby enabling the user to identify interesting features based on the distribution the product's completion ratios during the manufacturing process.

In contrast to displaying only two-dimensional graphs for the minimum, the

maximum or the average production times for all products of the same type, the presented visualizations are capable of providing trend analyzes over the whole manufacturing time, while still preserving a high level of intuitiveness. The views are also able to show interesting features like drastic changes in manufacturing time. This is not only possible for the overall production time of a product, but also for each individual operation. Yet, the user is enabled to get an overview over all products of the same type. Furthermore, the user is visually guided to operations that take longest or increase the production time most. At last, a visual exploration of changes in production times is possible by comparing the resulting production views based on different factory setups.

The presented flow view, workload view and production view focus on different aspects and form a combined system of interlinked views allowing the user to examine production properties and factory performance.

3.4 Conclusions

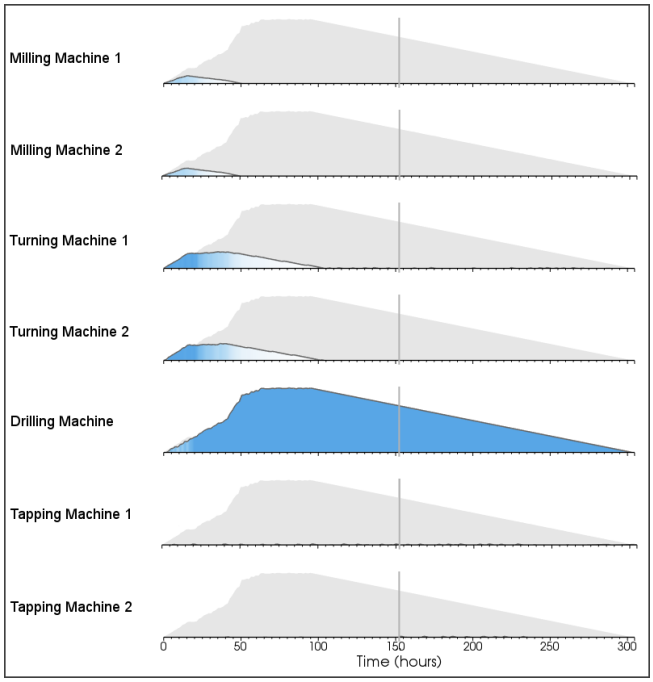
This chapter introduced a tool for analysis and visualization of manufacturing data generated by cyber-physical production systems. The tool displays manufacturing data in an intuitive format that communicates relevant information to a human expert. To enable an overall understanding of a manufacturing system's state and process, aggregated views were generated. In addition, the visualizations can focus on individual machines or products and zoom in onto interesting time steps. The user is guided between the interlinked views showing machine workloads, state of workpieces being manufactured, or material flows. Bottlenecks or excess machining capacities can visually be highlighted, thus guiding the user to interesting locations and events. The influence of changes of the factory setup, e.g. addition, removal or reconfiguration of machining tools, can be simulated, analyzed and evaluated.

To support more realistic analysis and visualization of process chains, properties like machine accuracies and criteria like costs or lead times are a natural and straightforward extension of the presented manufacturing system.

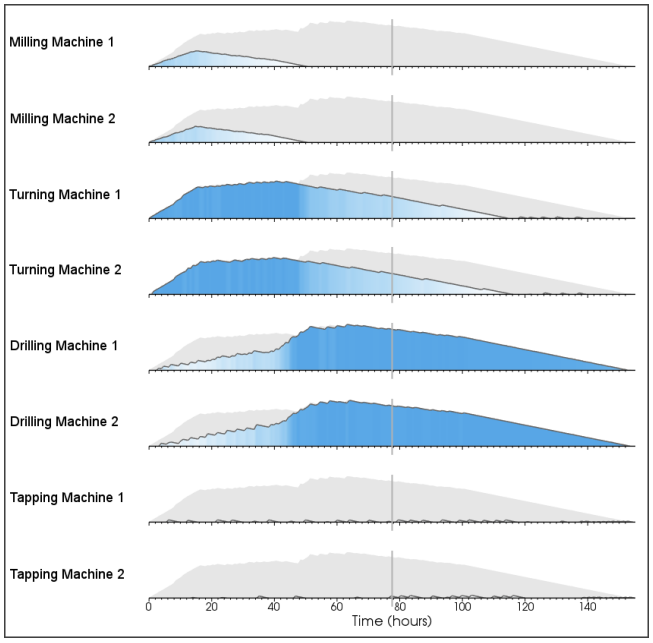
The presented visual analytics tool can be used to show the impact when using

different decision rules for production planning and control. Thus, this tool is suitable for the analysis of the behavior of self-controlled production environments. Here, the presented tool provides access to and visualizations of the detailed, underlying data and its patterns.

Considering the ever increasing size and complexity of data created by today's production systems, opportunities exist to greatly increase the flexibility of production systems with a focus on time or cost reduction, rapid adaptation to new manufacturing demands and product quality control. The presented approach holds the potential to evaluate these opportunities by mining production data and analyzing different engineering changes, thereby adding value to the decision-making process.



(a)



(b)

FIGURE 3.7: Workload view showing a workload for each machine in the virtual factory, thereby guiding the user to machines potentially being overloaded or redundant at certain interesting points in time. Image (a) shows a configuration with a single drilling machine, while image (b) reduces the overall production time (see scale of time axis) dramatically by using two drilling machines.

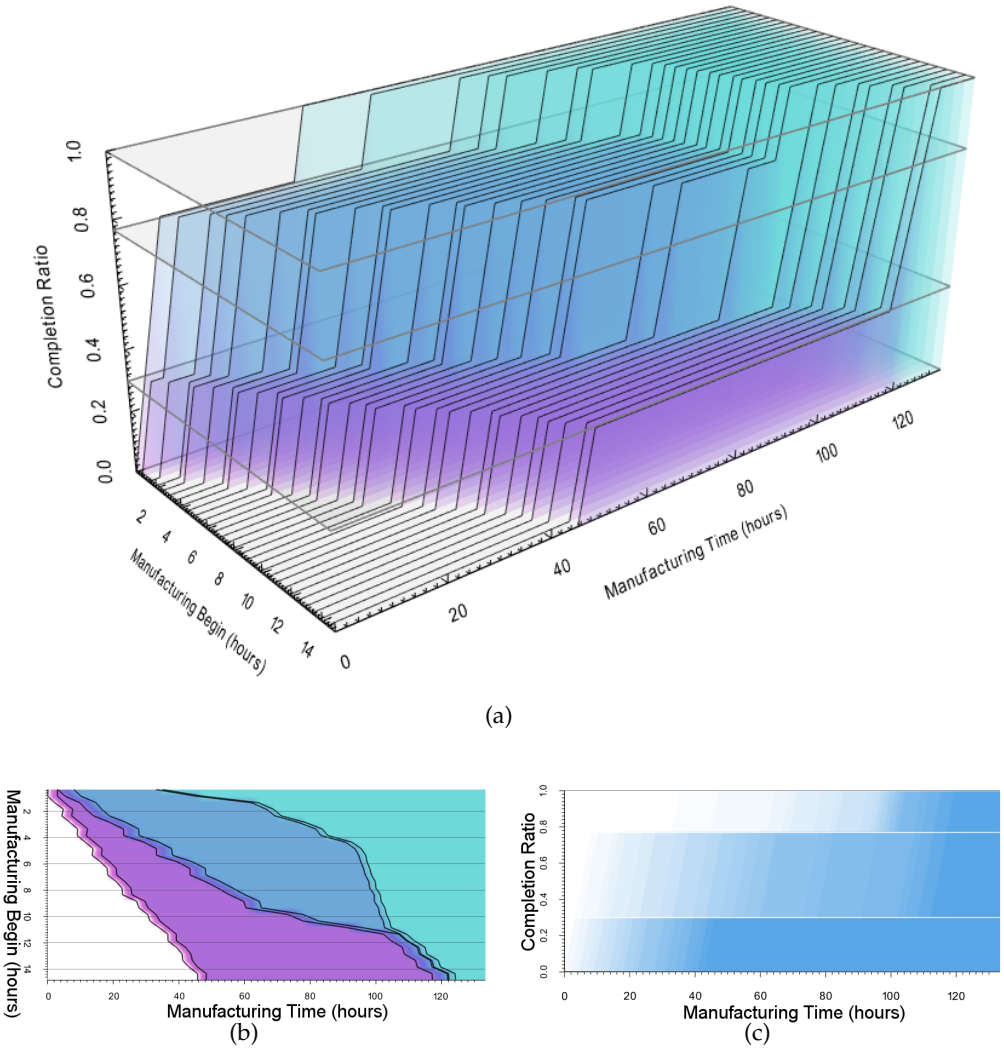


FIGURE 3.8: Production view for all products of type C showing inactive and processing phases of their three operations under different aspects of manufacturing time, manufacturing begin and degree of completion.

Chapter 4

Analysis of High-dimensional Data

4.1 Introduction

The representation of data quality within a high-dimensional dataset was mentioned as one of the top challenges in information visualization [61]. This is especially the case when a dataset contains noise or parameters such as aesthetics that cannot easily be evaluated mathematically. Although Pareto optimality is a widely used concept to identify optimal points in a high-dimensional space, hidden dimensions justify to not only consider optimal but also nearly optimal points. So the concept of Pareto optimality has to be extended to obtain measures on how efficient data points are.

Therefore, this chapter is based on the work of Post el al. [88] and introduces the Pareto factors describing three different ways to evaluate the relative amount of data points that are more efficient than the evaluated point in the sense of Pareto optimality. By utilizing these new measures, it is possible to evaluate the quality of all given data points and thereby guide users not only to points on the Pareto front, but also to interesting solutions near the Pareto front. The new measures can be embedded into established visualization techniques such as scatterplots or parallel coordinates in a straightforward manner as will be shown in this chapter.
Therefore, this chapter contributes:

- Scale-invariant and flexible measures based on Pareto optimality, called Pareto factors
- Visual encoding of Pareto factors in established information visualization techniques

4.2 Related Work

Pareto optimality is a widely used concept to identify optimal high-dimensional points in an arbitrary space [25, 27]. This section summarizes visualization techniques that are based on the concept of Pareto optimality.

Different applications such as fishery or architecture use Pareto optimality to identify interesting data points[82, 7]. Although this directly results in a set of optimal data points, these domains are usually confronted with various dimensions not well expressible or previously unknown. Therefore, this paper extends the definition of Pareto optimality.

Ruotsalainen et al. [101] use the gradient of the Pareto front to help the user navigate through this front to find interesting solutions. Although this is a suitable technique to navigate through Pareto optimal points only, hardly to express qualities like aesthetics require to consider solutions that are not quite Pareto optimal as well. Instead of being completely Pareto optimal, these solutions might have other properties such as aesthetics that are improved. This can be accomplished with the measures introduced in this chapter.

Witowski et al. [115] perform a study on how several known visualization techniques can be applied to visualize the Pareto front. They point out that a combination of several tools is most promising to visualize the Pareto front in a suitable way. These techniques can be extended by the measures introduced in this chapter, thereby enabling users to evaluate the quality of found solutions with respect to Pareto optimality.

4.3 Methods and Results

In compensation criteria like the Kaldor-Hicks efficiency [108] known from economics, a data point is defined to be more efficient if the sum of all gains is greater than the sum of all losses in comparison to another data point. Here, an optimal point is a data point for that no other data point is more efficient. Figure 4.1(a) shows the optimal point B in blue and all other points in dark gray. Point A is no Kaldor-Hicks optimum because there exists a more efficient point above the diagonal going

through point A, and point B is optimal because there exists no such point for B. The problem with this kind of measure is that it is not invariant to anisotropic scaling. Figure 4.1(b) shows the same set of data points, this time scaled anisotropically by a factor of three in the horizontal direction. Here, point B is not optimal any more because there exist more efficient points and point C becomes the new optimum.

To avoid this problem, this chapter is based on Pareto optimality which is scale-invariant [28]. A data point x is more efficient than another point y regarding Pareto optimality if x is greater than y in at least one dimension and not smaller in all other dimensions. Then, a data point is Pareto optimal if there exists no other data point that is more efficient. Figures 4.1(c) and 4.1(d) show the same data points with the same scalings as before, this time evaluated with Pareto optimality. As can be seen, the blue Pareto optimal points remain optimal after anisotropic scaling.

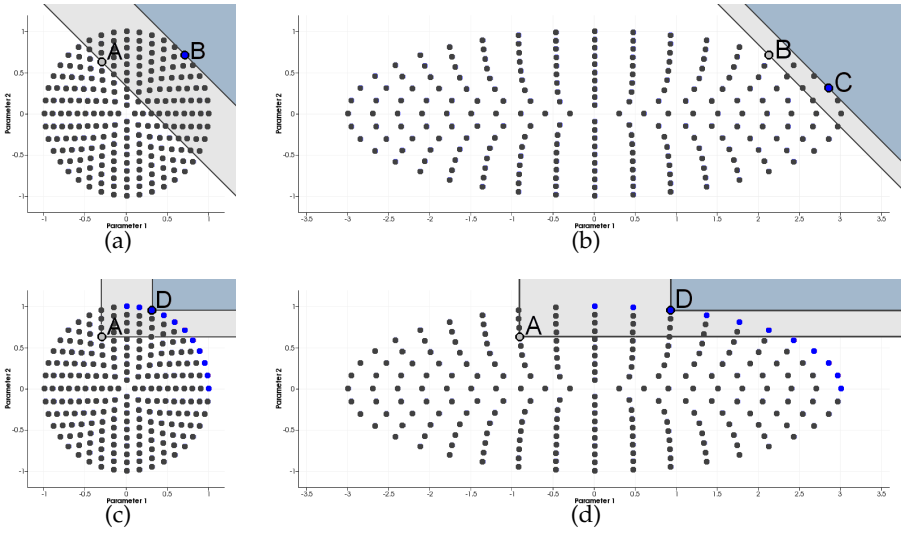


FIGURE 4.1: A set of data points with different scalings. (b) and (d) are anisotropically scaled by a factor of three. (a) and (b) show the optimality towards a compensation criterion and (c) and (d) show Pareto optimality (blue points are optimal). From (a) to (b) the optimum changes from point B to point C, meaning that this measure is dependent on the (anisotropic) scale. In contrast to that, the Pareto optimality in (c) and (d) is scale-invariant with multiple points being optimal.

To provide information on how far away from optimality non Pareto optimal points are, this chapter introduces novel measures as described in Equations 4.1, 4.2, and 4.3. The relations $<_p$ and $>_p$ respectively mean less and more efficient in the sense of Pareto optimality as described above. Here, $\vec{x}_1, \ldots, \vec{x}_n$ are the given data

points and $p_<(.)$, $p_>(.)$ and $p_{<>}(.)$ are the new measures called *Pareto factors*. These measures preserve scale-invariance since $<_p$ and $>_p$ are also invariant to anisotropic scaling.

The measure $p_<(.)$ provides the relative number of less Pareto efficient data points to all points. This behaves like a distance to the least Pareto efficient points (compare to second row of Figure 4.2). The measure $p_>(.)$ provides the complement of the relative number of more Pareto efficient data points to all points. This behaves like a distance to the most Pareto efficient points (compare to third row of Figure 4.2). And the measure $p_{<>}(.)$ is a combination of both previous measure and provides the relative number of less Pareto efficient data points to points that are either less or more Pareto efficient. This behaves likes a distance between the least and most Pareto efficient data points (compare to forth row of Figure 4.2).

$$p_<(\vec{x}_i) = \frac{N_<(\vec{x}_i)}{n-1} \tag{4.1}$$

$$p_>(\vec{x}_i) = 1 - \frac{N_>(\vec{x}_i)}{n-1} \tag{4.2}$$

$$p_{<>}(\vec{x}_i) = \frac{N_<(\vec{x}_i)}{N_<(\vec{x}_i) + N_>(\vec{x}_i)} \tag{4.3}$$

with

$$N_<(\vec{x}_i) = \left| \{ \vec{x}_j \in \{\vec{x}_1, \ldots, \vec{x}_n\} : \vec{x}_j <_p \vec{x}_i \} \right| \tag{4.4}$$

$$N_>(\vec{x}_i) = \left| \{ \vec{x}_j \in \{\vec{x}_1, \ldots, \vec{x}_n\} : \vec{x}_j >_p \vec{x}_i \} \right| \tag{4.5}$$

Normalizing these measures to range from zero to one for all data points yields what will be called the *normalized Pareto factors* and is useful for evaluation, weighting or visual analysis. Until now, the user cannot choose how far all data points are considered or how far only Pareto optimal points are of interest. To compensate this, the normalized Pareto factors are raised to some user defined power. Figure 4.2

shows the effect of different exponents, where higher exponents focus on Pareto optimal points only, while lower exponents preserve an overview over all data points.

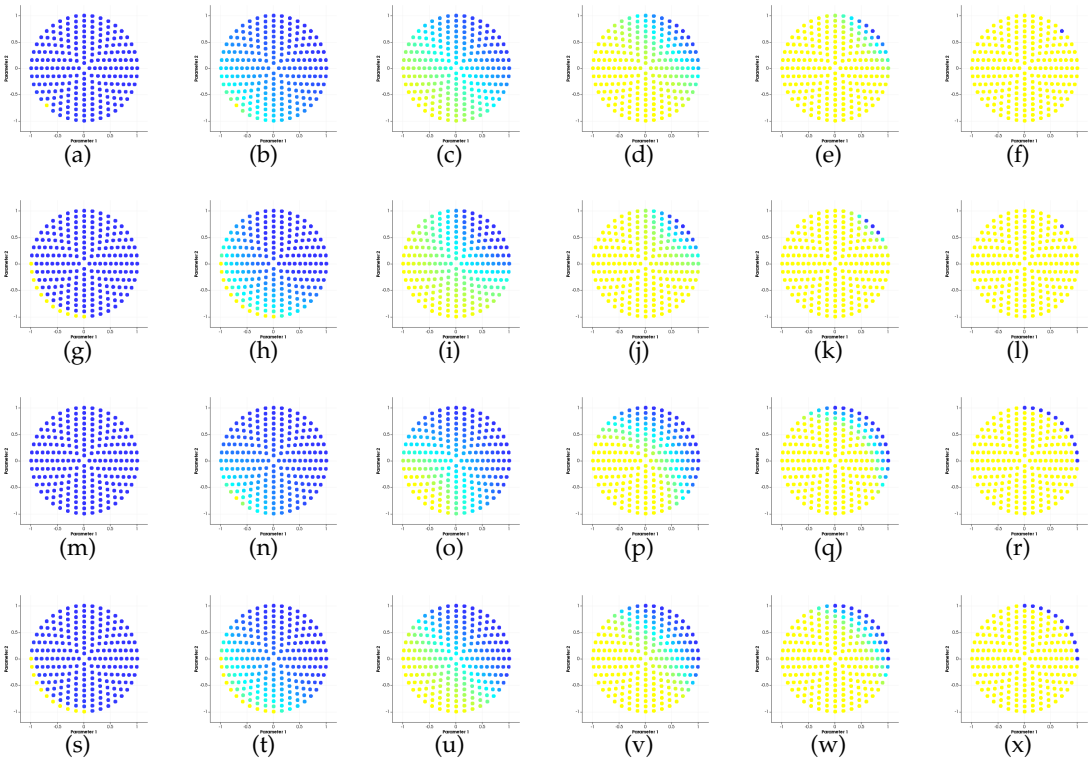


FIGURE 4.2: The introduced Pareto factors are used to evaluate the quality of data points with respect to Pareto optimality. The first row shows a scale-dependent compensation measure as a comparison. The second, third and forth rows show the new Pareto factors $p_<$, $p_>$, and $p_{<>}$, respectively. A scatterplot visualization is extended showing the value of the measures using a color scale ranging from yellow (0.0) over cyan (0.5) to blue (1.0). The effect of different exponents is shown for the exponents 0.0, 0.25, 1.0, 4.0, 16.0 and ∞ (from left to right). As shown, the different measures favor points near the least or most Pareto efficient data points differently.

Figure 4.2 shows how the Pareto factors can be applied to established information visualization techniques like scatterplots. A color range from yellow (Pareto factor of 0.0) over cyan (0.5) to blue (1.0) is used. Users can interactively manipulate the used exponent to focus more or less on the Pareto optimal points only. The applicability of the presented measure is not limited to scatterplots only but can also be applied to scatterplot matrices, parallel coordinate plots, or star plots, for example. Within these techniques, the presented Pareto factors allows to evaluate the quality of data points.

4.4 Conclusion

This chapter introduced novel measures for the quality of data points in a high dimensional space based on Pareto optimality. It was shown that the introduced measures are scale-invariant and enable the evaluation of the efficiency of data points with respect to Pareto optimality. Based on these measures, it was possible to visually extend established information visualization techniques. This helps to not only consider Pareto optimal data points that might not be the desired solution for problems with noise or hard to evaluate parameters like aesthetics. Instead, also nearly Pareto optimal solutions can be analyzed to find a desired tradeoff.

Chapter 5

Graph Extraction using Fast 3D Thinning

5.1 Summary

In many applications graphs and networks need to be analyzed, although they are not defined directly through the input datasets. A often occurring example are image data. Often, these images capture objects, that can be represented as a graph or flow network. Example application, that obtain image data as an input and need to transform those images into graphs and networks are face recognition [44], vascular trees [97] and text recognition [35]. Therefore, the input images need to be transformed into a graph or a flow network before the visual analysis of the extracted graphs can be performed, which is a task of the image processing field.

To understand the shape of different objects in an image and therefore offering the first step to transform an image into a graph, centerlines are a common tool. They offer a geometric and topological representation, which allows further examinations [56]. To obtain a centerline, thinning is often the first step, where voxels are successively deleted until a skeleton remains [96]. The resulting skeleton can be further used to obtain a graph that can be examined using visual analytics methodologies.

Unfortunately, the properties of a skeleton are not unique and alter depending on the use case in different application [17]. Independent from the application, skeletons need to fulfill specific requirements to be suitable for further examinations. First, skeletons require a one pixel thickness, to be analyzed properly. Additionally,

interesting structures of an image are usually segmented, resulting in a discrete binary mask, which needs to be handled by the thinning algorithm. Furthermore, the thinning result requires the ability to capture the geometry of the thinned objects as well as possible. This means, that the skeleton needs to preserve the length of an object and it needs to be located in the object's center. Moreover, the topological properties of the object have to be retained. This means that, connected components or branches need to be presented by a skeleton with the same properties. Finally, as thinning often needs to be computed for various objects, a fast computation is required. Depending on the application, these list of properties needs to be extended.

Therefore, this chapter is based on the work of Post et al. [90, 91] and presents a fast and robust thinning approach (Section 5.3). The algorithms remove successively voxels from the input object by evaluating a moving local neighborhood until the desired skeleton remains. In order to achieve a fast computation, all possible neighborhood settings are stored in a lookup table. To provide skeleton outputs with different properties, the presented method allows the use of different lookup table. To show the effectiveness of the presented solution, this chapter offers a runtime comparison with the widely used ITK library and demonstrates skeletons obtained with the presented approach (Section 5.4).

5.2 Related Work

The following section will give an overview on thinning algorithm classes and presents relevant examples and their properties.

Reviews of thinning approaches are provided by Lam et. al [60] (2D) and Saed et. al [103] (3D). They stated, that thinning methods can be divided into iterative and non-iterative approaches. Iterative approaches start with an object that is thinned by iteratively deleting border voxels using a specific scheme until the skeleton remains. This category itself can be divided into parallel and sequential approaches where sequential approaches check for each voxel separately whether it can be deleted or not. Parallel thinning algorithms decide in each iteration for which can be deleted. These algorithms often differentiate between the direction from where voxels are deleted

and alter their iterations based on them. Depending on the amount of different iterations an algorithm performs the number of cycles for a parallel thinning algorithm can be determined.

Thinning can be performed for grey scale images as shown in [77, 18]. Although these algorithms can handle arbitrary image data, their computational effort increases. Additionally, in medical image processing thinning is applied to a preprocessed image where the object to be thinned is already determined via a segmentation step. This results in a binary image that does not benefit from the flexibility of a grey-scale image thinning. Therefore, this paper addresses binary images as the medical domain does not require a grey scale thinning approach.

Thinning methods for binary images are available as sequential [38] or parallel approaches [3, 1]. Their common drawback is the thinning result computed without concerning the local neighborhood of a voxel. They result in skeletons that do not have a thickness of one voxel. As mentioned before, this is an important requirement in medical image processing. To solve this problem, this paper presents a thinning algorithm based on local neighborhood evaluations, that outputs a skeleton with a thickness of one voxel.

Various methods [8, 21, 71, 74] use a moving local neighborhood window to evaluate if a voxel can be deleted or not. Unfortunately, they do not preserve the geometric and topological properties of the examined medical object. Therefore, this paper presents a thinning method based on local neighborhood evaluation that results in a skeleton suitable for medical image processing.

Lee et. al [64] presented a parallel, topology and geometry preserving algorithm that results in a skeleton with a thickness of one voxel. Therefore, this algorithm is able to fulfill the mentioned requirements for skeletonization of medical image data. The approach is implemented in different libraries [75, 42]. The underlying algorithm determines deletable voxels by checking different properties. Although the approach is based on a lazy evaluation (if one property does not hold, the remaining do not need to be checked) this approach can be computationally expensive. Therefore, this paper presents a fast implementation that is able to identify deletable voxels correctly, fast and with an equal time consumption in all possible cases by using a lookup table.

5.3 Methods

The following section describes how thinning can be performed by using lookup tables. In addition, the section shows a general procedure for parallel thinning that can utilize arbitrary lookup tables to increase the flexibility.

5.3.1 Local Neighborhood Lookup Tables

In order to determine deletable voxels, a moving local neighborhood, as shown in Lee et al.'s approach [64], is used. Therefore, the work states a family of thinning algorithms that can be utilized to obtain skeletons fulfilling different properties from an binary image data. In each thinning algorithm, a list of criteria (see Table 5.1) is used to evaluate the moving local neighborhood. Their three sets of used criteria and the different thinning results will be explained in the following.

Simple thinning (T_1), requiring criteria $C_1 - C_4$, results in a skeleton that preserves the object's geometry and topology, while neglecting line-like branches.

Instead, the implementation of a **medial axis thinning** (T_2), which additionally preserved the length of an object, requires criteria $C_1 - C_5$. In addition to the properties of the simple thinning algorithm, condition C_5 avoids a shortening of line-like structures, which outputs a medial axis.

To achieve a **medial surface thinning** (T_3) as a thinning result, criteria $C_1 - C_4$ and C_6 are required. Criteria C_6 verifies, if the observed voxel is embedded in the medial surface or its border. If this is the case, the voxel is not allowed to be deleted, thereby remaining the medial surface of the structure.

In the algorithms presented by Lee et al. the moving local neighborhood determines, if a voxel is deletable, by checking the applied criteria according to their order in Table 5.1. This evaluation is performed until either a criteria is not fulfilled (voxel will not be deleted) or all criteria are fulfilled (voxel will be deleted). Depending on the criteria, that causes the computation to stop, the runtime for a local neighborhood evaluation varies. As a result, the runtime of the approach depends on the presence of slow cases in the input object, whereas a dependency of the image size is preferred.

#	Formula	Usage	Intuition
C_1	$f(v) = 1$	T_1, T_2, T_3	The current voxel has to be filled.
C_2	$f(v - d) = 0$	T_1, T_2, T_3	Considering the current direction of thinning, the predecessor of the voxel is not allowed to be filled.
C_3	$E(v) = 0$	T_1, T_2, T_3	The Euler characterists of a point needs to be unchanged. This means, that the geometric properties of the thinned object are preserved
C_4	S(v) = 1	T_1, T_2, T_3	The considered point needs to be a simple point. This means, that removing this point, remains the objects topology.
C_5	$\lvert\{n : n \in N_{26}(v) \wedge f(n) = 1\}\rvert > 1$	T_2	The number of neighbors in a local neighborhood of a point needs to be higher than one. This prevents shortening line-like structures and therefore keep their length.
C_6	$\neg\forall i \in \{1...8\} : Index(N_i^2(v)) \in \{153, 165, 170, 195, 204, 240\} \vee \lvert N_i^2(v)\rvert \leq 3$	T_3	The voxel is not allowed to be a medial surface point or the edge of a medial surface while considering the octants in the voxel neighborhood, as shown in Lee et al's work. The method presents a lookup table deciding weather a voxel belongs to the medial surface or not. The list of matching cases (identified by their number) was extended to obtain correct results.

TABLE 5.1: Thinning criteria of the presented lookup tables and their intuition. The different use of these criteria alters the output of the thinning result.

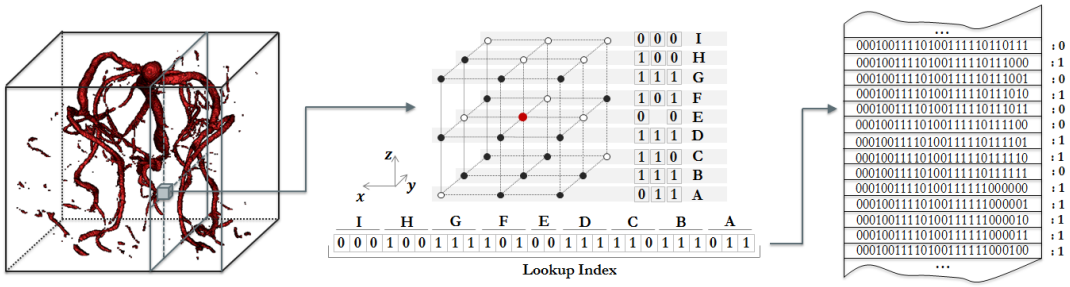


FIGURE 5.1: Scheme for neighborhood encoding with example output. The moving local neighborhood evaluates the single voxels of the volume by encoding its neighborhood setting to a lookup index. This index can be used to determine weather a voxel needs to be deleted or not.

As the evaluation output of the moving local neighborhood remains the same in the entire computation of the thinning algorithm, their result can be computed in advance and stored in a lookup table. Based on that, the algorithm does not need to recheck all criteria again during its runtime. Instead it can request the output of the current neighborhood setting in the lookup table and therefore requiring an constant time consumption for each voxel evaluation.

To achieve a standardized indexing scheme for arbitrary lookup tables, the presented method of Post et al. [90] is used as shown in Figure 5.1. The table index is calculated by stringing together the filled (value 1) and not filled voxels (value 0) from the local neighborhood, except for the current voxel value itself. As this value needs to be checked in the thinning procedure as well, this would not lead to a reduction of time consumption. In total, the resulting lookup table holds 2^{26} entries, storing all possible settings of a voxel's local neighborhood. This results in a 8MiB file, that can be loaded in less then one second. For each of the presented thinning strategies (simple thinning, medial axis thinning and medial surface thinning) our approach provides a separate lookup table. Each of the provided tables can be used as a basis for the thinning algorithm. Besides the presented lookup tables, novel lookup tables can be created if they match the presented index scheme shown in Figure 5.1.

```
function THINNING( Image, LookupTable )
    repeat
        modified ← false

        for d ∈ {up, down, right, left, forward, backward} do

            Candidates ← ∅

            for v ∈ Voxels_Image do
                if ( Image( v )                                  =  1  ∧
                     Image( v − d )                              =  0  ∧
                     LookupTable( Index( Neighborhood_Image( v ) ) ) =  1 ) then
                    Candidates ← Candidates ∪ { v }

            for c ∈ Candidates do
                if LookupTable( Index( Neighborhood_Image( c ) ) ) = 1 then
                    Image( c ) ← 0
                    modified ← true

    until ¬modified
```

FIGURE 5.2: Pseudocode for parallel thinning, using arbitrary lookup tables.

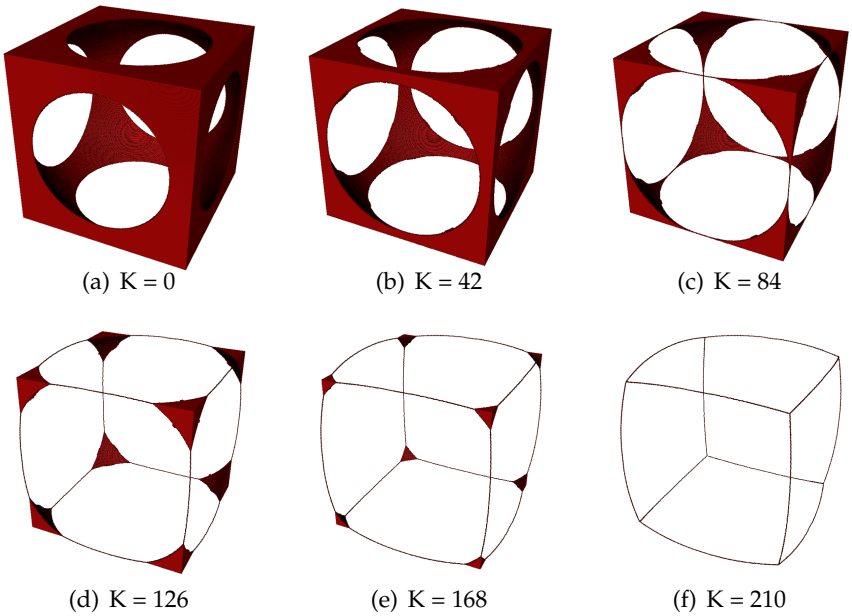
5.3.2 Thinning Algorithm

The following pseudocode sketches the optimized thinning procedure and shows how the lookup table is used to successively remove voxels until the skeleton remains:

As an initial step, the input image is enlarged in each direction by adding a voxel with the values of 0. Still, the thinning algorithm works on the original image voxels. Resulting from the border duplication, there are no special border cases to consider for the thinning procedure. In the original image, the algorithm needs to check each voxel if it belongs to the border what can be computational expensive. This problem can be avoid by the enlarged border.

After that, the actual thinning can be started. Each iteration consists of 6 cycles that alter the direction d of the thinning (up, down, right, left, forward, backward). This ensures, that the remaining skeleton lies as close to the center of the thinned object as possible.

In each iteration all voxels of the input image are evaluated by the moving local neighborhood. If their corresponding entry in the lookup table ($LookupTable(v)$) identifies the current voxel as deletable, it is stored in a list of *Candidates*. This list holds all voxels that fulfill the five deletion criteria and thus they are candidates to



(a) K = 0 (b) K = 42 (c) K = 84

(d) K = 126 (e) K = 168 (f) K = 210

FIGURE 5.3: Simple thinning applied to an artificial dataset showing the thinning result after K iterations.

be removed.

After all voxels are evaluated and the list of *Candidates* is filled, the algorithm rechecks the lookup table for all *Candidates* sequentially. If the checked voxel is still deletable it is finally deleted in the image. This can change the local neighborhood of voxels in the list of *Candidates* what makes the recheck necessary. The recheck of a voxel needs to be followed directly by its deletion, as simultaneous deletion could lead to topological or geometrical changes of the skeleton.

After the recheck and final deletions, the next cycle starts from a different direction, working on the manipulated image. This is repeated until the image cannot be modified any longer from any of the thinning directions.

As Lee et al. showed, this results is a skeleton that has a thickness of one, preserves the geometric as well as the topological properties of the thinned object.

Figure 5.3 shows the progress of the thinning procedure while thinning an artificial dataset. The thinning results after 0, 42, 84, 126, 168 and 210 iterations can be reviewed. The images show that the thinning procedure successively removes voxels from the original set of voxels until a skeleton with the thickness of one voxel remains which is located in the center of the original object.

5.4 Results and Discussion

In order to show the effectiveness of the presented approach, this section presents a comparative study to the widely utilized ITK implementation. The standard implementation, that is provided in ITK Version 4.8 [52, 43] can be applied to a 3D dataset. As the implemented method performs a single thinning procedure in each layer of the three-dimensional input image, it can not be guaranteed, that the connection between the layers is preserved. Instead, the implementation of the ITK Journal, which uses Lee's medial axis thinning, is considered for a comparative study [42]. To obtain fair results, the algorithm was uncoupled from the ITK framework to provide an equal datastructure for the tested approaches. In order to complete the study, the simple- and medial axis approaches of Lee et al. where added to the ITK implementation.

The resulting skeletons can be seen in Figure 5.4. Therefore, four datasets (box cross, hollow cube, engine [114] and vessels) are thinned with the three thinning approaches presented in this chapter as well as the ITK implementation. As the results show (see Table 5.5), our solution is able to perform thinning tasks up to twice as fast as the approach of Lee et al. . The factor of speedup is highly depending of the relative amount of hard to evaluate neighborhoods, where the approach of Lee et al. needs to check various criteria and the lookup table approach only requires a constant evaluation time.

The presented approach outputs a correct result, that fulfills the criteria encoded in the lookup tables. The system is designed flexible, as arbitrary lookup tables can be loaded, if they provide the described file format. The presented method contains three lookup tables for simple thinning, medial axis thinning and medial surface thinning. As Lee et al. showed, their criteria are able to delete the maximum number of voxels in each iteration correctly. Their implementation in combination with the presented lookup table approach lead to a minimal computational effort, if a parallel algorithm design is not considered.

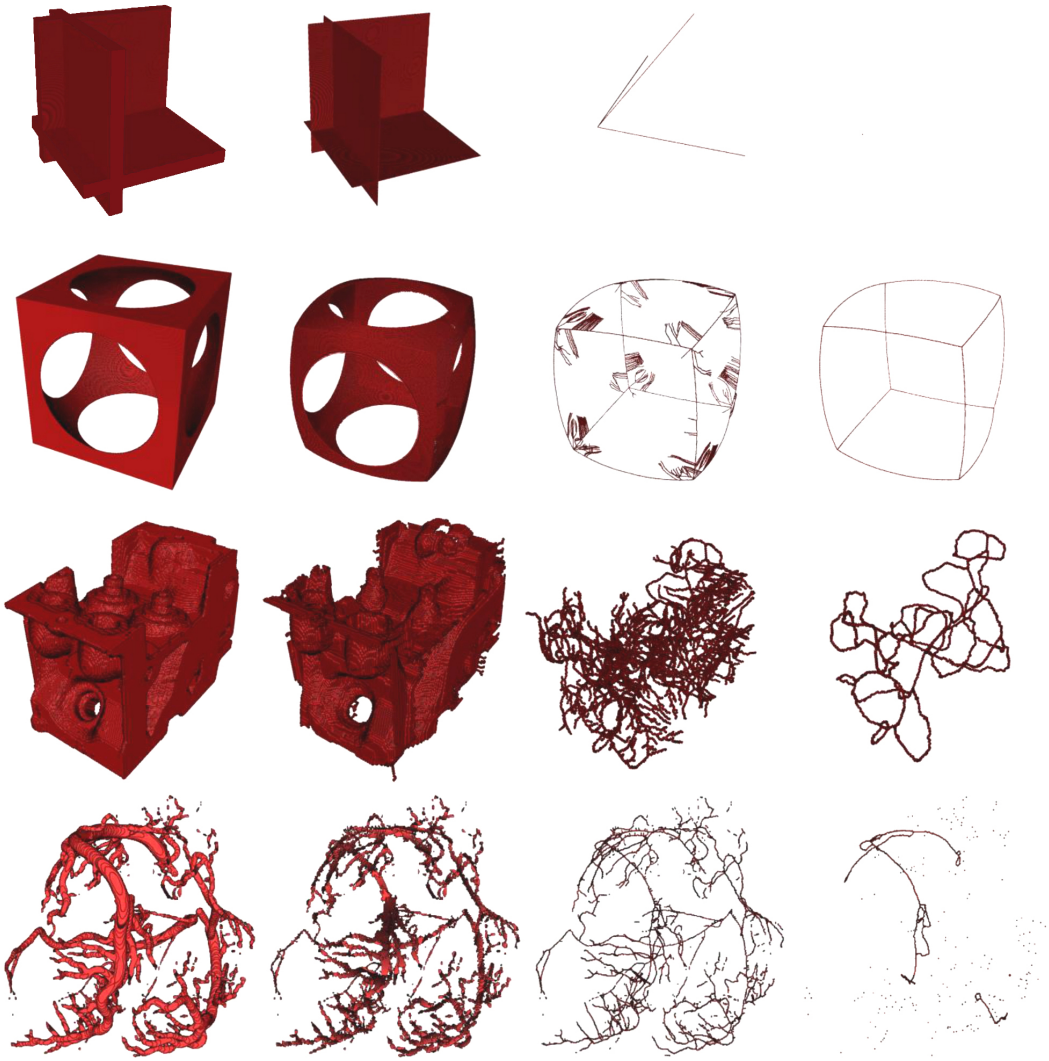


FIGURE 5.4: Example datasets and their thinning results. From top to bottom: Box cross dataset, hollow cube, machine engine and vessels. Left to right: original object, medial surface thinning, medial axis thinning and simple thinning. The results show, how the different encoded criteria output altering thinning results. According to the special needs in different applications, the table can be selected. As these examples show, medial surface thinning is beneficial for plate lite structures (e.g. engine), wheres medial axis thinning shows good results for tube like structures (e.g. vessels).

			Medial Surface Thinning				Medial Axis Thinning				Simple Thinning			
Dataset	Size	Voxel Ratio [%]	# Iterations	Voxel Ratio [%]	ITK Thinning [s]	LUT Thinning [s]	# Iterations	Voxel Ratio [%]	ITK Thinning [s]	LUT Thinning [s]	# Iterations	Voxel Ratio [%]	ITK Thinning [s]	LUT Thinning [s]
Box Cross	512 x 512 x 512	23.14	150	0.493	330.6	195.4	125	0.001	275.5	168.9	138	0.000 0007	335.3	194.3
Hollow Cube	513 x 513 x 513	15.15	41	0.442	95.1	58.3	38	0.017	85.2	52.2	38	0.004	81.8	52.2
Engine	256 x 256 x 128	12.60	16	1.309	3.24	1.78	24	0.125	3.98	2.32	23	0.027	3.53	2.14
Vessels	512 x 512 x 199	0.4997	11	0.070	8.02	5.09	7	0.016	4.98	3.29	104	0.002	72.9	45.6

FIGURE 5.5: Comparison of the presented approach with the ITK journal implementation. The ITK solution for medial axis thinning was decoupled from the general ITK framework to allow a fair comparison to the presented solution. It was possible to alter the implementation of the used criteria to allow simple- and medial surface thinning as presented by Lee et al. . As a result, all algorithms can be compared to the presented solution. The results show, that the lookup table approach implemented in uor approach is able to perform thinning tasks up to twice as fast as the implementation of Lee et al.'s approach.

5.5 Conclusion

This paper presents a local neighborhood-based thinning implementation utilizing a lookup table. Therefore, the algorithm is optimized to allow a fast and robust thinning. The obtained implementation is able to generate a one pixel wide skeleton that preserves the geometry and topology of the examined objects. As the lookup table approach of the moving local neighborhood reduces the computational effort of finding deletable voxels to a constant cost, the underlying algorithm is accelerated. The presented approach offers three different thinning approach: simple thinning, medial axis thinning and medial surface thinning. The required lookup tables are provided to be reused in an arbitrary framework .

Chapter 6

Visual Analysis of Network Bottlenecks

6.1 Summary

The analysis of flows is an important topic in various applications such as cyber security [50], biological pathways [118] and cyber physical manufacturing systems [55]. In particular, one aim of designing a manufacturing system is to identify weaknesses in the manufacturing system's layout in an early planning stage to minimize costs, raise the product quality and shorten production times. An important factor to optimize production system is the identification and elimination of bottlenecks [63].

The analysis of bottlenecks in cyber physical manufacturing systems can be described by flow networks with machines as nodes/vertices and product flows as edges of the network. Depending on the factory setting, each of these edges has a specific capacity, describing the maximum amount of products that can flow between the two connected machines. To identify the bottleneck of a manufacturing system, the correlating flow network is subject of analysis. Contrary to intuition, the bottleneck of a flow network is not a single edge between two nodes. Instead, a bottleneck is a whole set of edges. The minimum cut of a flow network can help in 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 being the remaining nodes. In this mathematical 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 compare various network configurations and their resulting bottlenecks. A sufficient solution targeting all mentioned problems was not provided so far, as shown in section “Related Work".

This chapter is based on the work of Post et al. [89, 93] and extends the definition of a minimum cut in a flow network by separating the nodes of a flow 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 specific edges that are the bottlenecks of the network. To define an intuitive visualization for bottlenecks in a network, this chapter presents an intuitive visualization based on Voronoi diagrams [30] derived from the underlying graph’s node layout. This chapter uses color-coded regions to indicate bottleneck transition of a flow network. Based on these regions, an ensemble visualization technique for multiple configurations of a flow network is presented in this chapter. The resulting ensemble visualization indicates common bottlenecks and differences in the underlying configurations by an intuitive color-coding of Voronoi cells (section “Methods"). Section “Conclusion" will summarize this chapter.

Therefore, this chapter contributes:

- An extended definition of the minimum cut in flow networks
- An intuitive visualization of a minimum cut in a flow network
- An intuitive ensemble visualization for multiple configurations of a flow network

6.2 Related Work

This section will present the state of the art in minimum cut visualization as well as ensemble visualizations for flow networks.

6.2.1 Visualization of Minimum Cuts

Vehlow et al. [113] 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 to that, the presented approach introduces a visual encoding for the minimum cut based on Voronoi cells.

Brandes et al. [12] presented a planar visualization for the minimum cut in flow networks by arranging a network in an rectangular manner and adding a poly-line to indicate the cut. This method is widely used in open source solutions as [58, 57, 34]. Although this method gives a suitable first indication of the minimum cut, it can not indicate edges in a flow network that represent a bottleneck for the network. The presented approach utilizes the method of Brandes et al. as a starting point and refines the definition of a minimum cut to enhance transitions that form the bottleneck of the considered system.

6.2.2 Ensemble Visualization of Flow Networks

The ensemble visualization of graphs is an important feature to allow a comparative investigation of their behavior. A state of the art report was given by Borgo et al. [33]. They presented a large variety of visualization techniques for graph ensemble. The relevant techniques will be discussed below.

Using the third dimension to stack different network configurations on top of each other was presented by Brandes et al. [11] and Itoh et al. [49]. Although this provides a good overview over the capacity values in the considered network configurations, the approach is not able to indicate common bottlenecks. In contrast to that, the presented approach uses a visualization based on regions that indicate the specific location of nodes with respect to the bottleneck of a network.

Temporal variance [5, 9, 10], variety-based edge visualization and selective accumulated visualization [9] form techniques that are visualizing the variety of occurring capacity constraints in a network ensemble. Although they provide an interactive visualization for an overview of the occurring capacity values, they lack the ability to indicate common bottlenecks. Therefore, the presented approach allows to

identify common bottlenecks of a given ensemble of a network's capacity configurations.

Cesario et al. [14] presented a technique to visualize graph ensembles by visually encoding multiple capacities per edge. This enables users to get an overview of different network configurations, but lacks the ability to show the resulting bottlenecks. In contrast to that, this chapter focuses on the visualization and accumulation of bottlenecks and their propagation.

Boyandin et al. [10] presented a visualization of graph capacities that are aligned in a plane and can therefore be reviewed in total. Although, the line representation of the minimum cut could be added to each of these representations easily, it would be hard to identify their similarities. Therefore, this chapter presents a single visualization for the entire ensemble that is able to indicate stable and divergent bottleneck regions.

6.3 Single Bottlenecks

The analysis of bottlenecks in flow networks is an essential task for many real world applications in planning and engineering. This section will introduce a method to visually inspect single bottleneck fronts in planar flow networks. Also, a method to analyze the propagation of bottlenecks for a network with an ensemble of different configurations will be demonstrated.

6.3.1 Methods

Flow Networks

This chapter will rely on the general definition of flow networks with a single source and sink, which will be presented below. Figure 6.1 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.

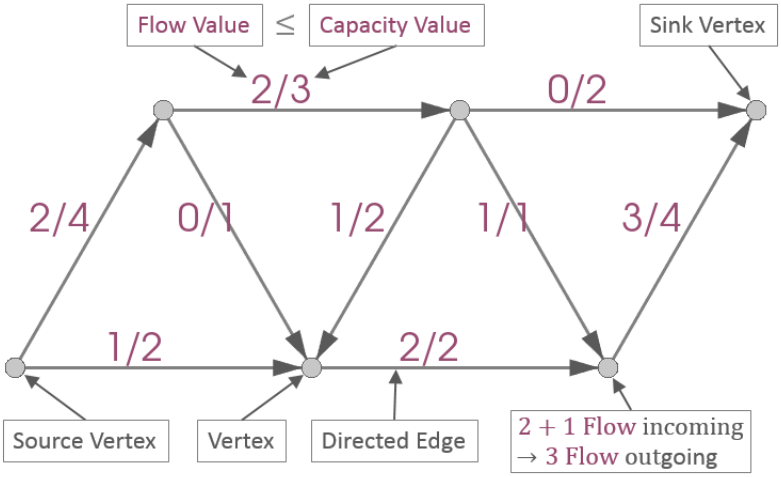


FIGURE 6.1: Flow network consisting of vertices, directed edges, a source and sink vertex, and a flow and capacity value per edge. The capacity limits the flow. Except the source and sink, all vertices preserve the flow.

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 have to apply for such a flow. The flow should be limited by the capacity, so $\forall e \in E : f(e) \leq c(e)$, meaning, that 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, so $\forall v \in V \setminus \{s,t\} : \sum_{(w,v)\in E} f(w,v) = \sum_{(v,w)\in E} f(v,w)$, meaning, that 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 a 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 chapter focuses on planar flow networks, so 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 2D plane without edges crossing each other. Figure 6.1 shows an example for a planar flow network with a proper embedding into the image plane.

Maximum Flows

A *maximum flow* $\hat{f}$ on a network N has the largest value among all possible flows on N, so 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. So 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 [29] is a general approach to find such a maximum flow. Figure 6.2 shows the individual steps of this approach. To understand this approach, the definition of a residual network needs to be understood.

For the flow networks in Figure 6.2(a)-6.2(g), the respective residual networks are shown in Figure 6.2(b)-6.2(h). 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 $G_f = (V, E_f)$. So 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 now operates on these residual networks. For a flow network in Figure 6.2(a)-6.2(g) the residual network in Figure 6.2(b)-6.2(h) is calculated. 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

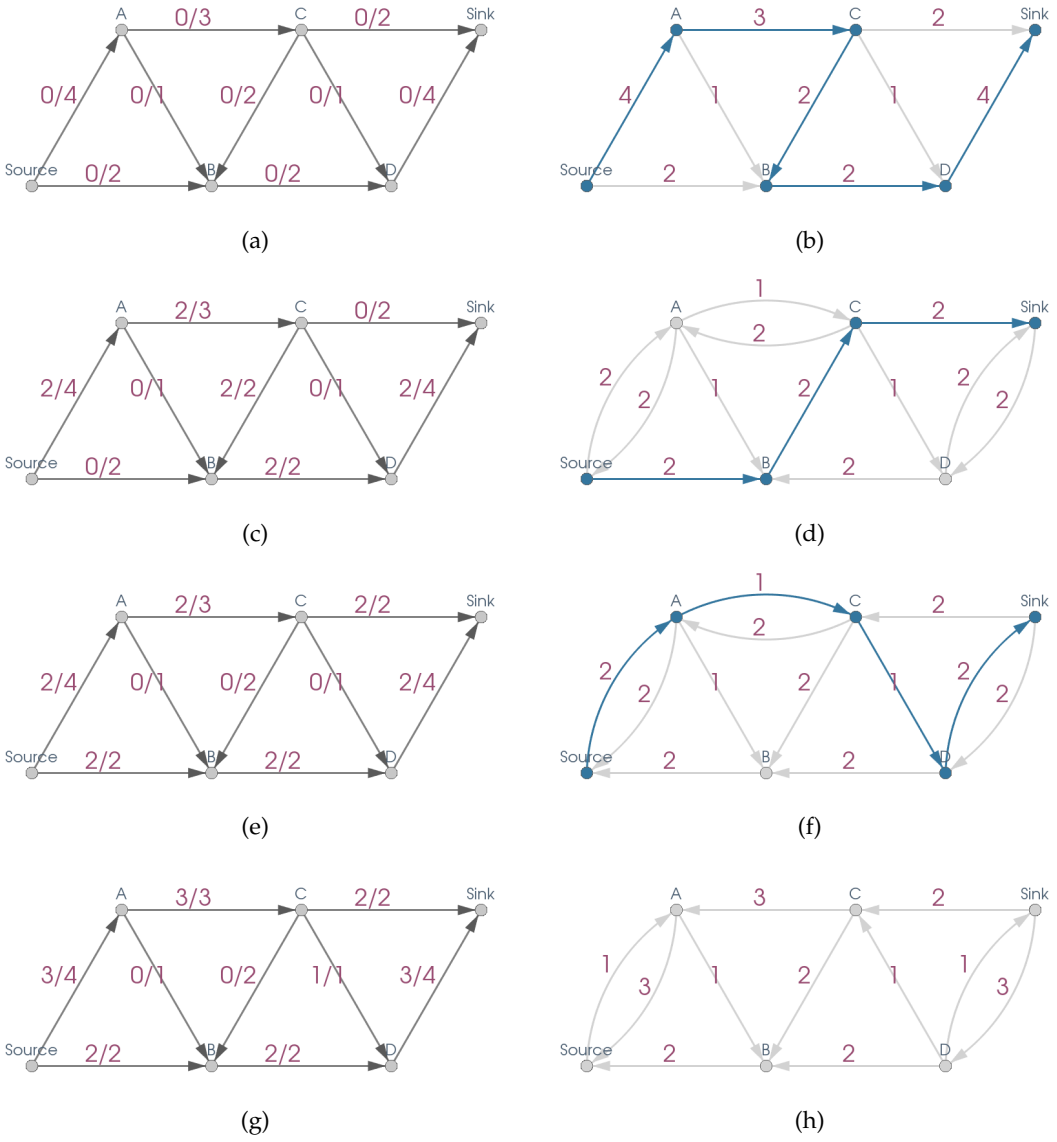


FIGURE 6.2: Iterations of the Ford and Fulkerson method to improve the flow of a network. To a given flow network (left images), the residual network is calculated and a path from source to sink is found (right images). This augmenting path is used to increase the flow along the path's edges in the original network. When no more path is found (image (h)), the flow is a maximum flow (image (g)).

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. Figure 6.2(g) shows the maximum flow with a value of 5.

The algorithm of Edmonds and Karp [22] 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 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 $O(|V| \cdot |E|^2)$, meaning that 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 faster 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.

Minimum Cuts

To find bottlenecks in networks, maximum flows could be considered. As the maximum flow in Figure 6.2(g) shows, there are exhausted edges. For these edges the flow value equals the capacity value, so the flow can not be increased any further. An example for such an edge is the edge $(Source, B)$ with values "2/2". One could easily think, that increasing the capacity of this edge would result in a larger maximum flow, meaning, that this edge would be called a bottleneck edge in the following. It turns out, that this early intuition is wrong, and an increase of the capacity of this edge would not increase the value of the maximum flow. To countervail this effect, this chapter focuses on cuts instead of flows.

A *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 a 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 [23] 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 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 then is $\check{C} = (\check{S}, \check{S}')$.

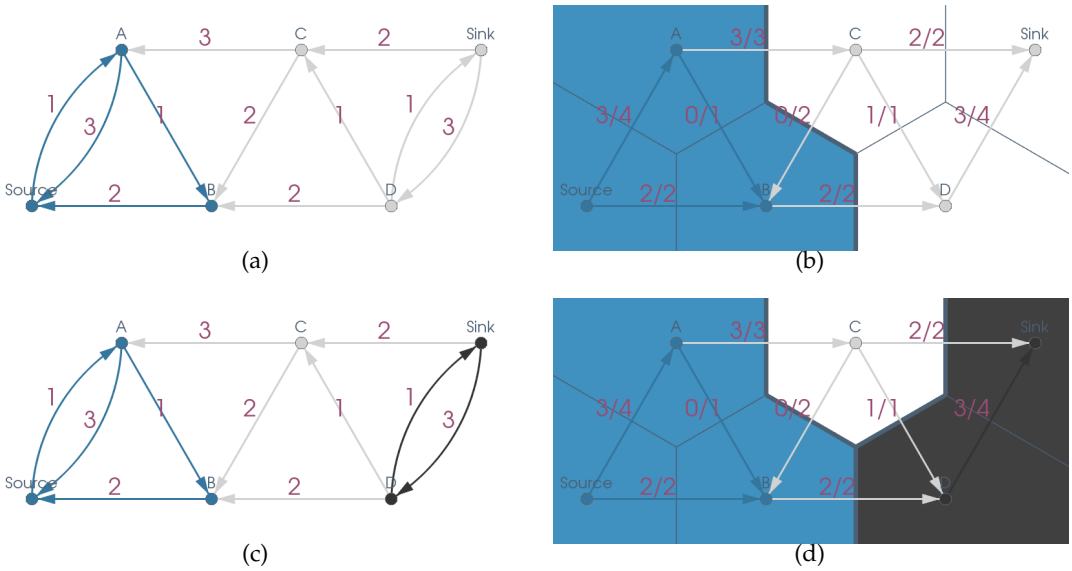


FIGURE 6.3: Residual network of an exemplary network with maximum flow (compare Figure 6.2(h)) with vertices/edges reachable forwards from source or backwards from sink (left images), and the original network with minimum cut and classified Voronoi cells (right images). The classical construction of a minimum cut (upper images) suggest wrong bottleneck edges, while the new extended construction (lower images) shows the true bottleneck transitions (blue to black).

Figure 6.3(a) shows the residual network of the maximum flow (compare Figure 6.2(h)), the collection of vertices starting from the source in blue, and the remaining vertices in white. To enhance the intuitiveness of the visualization and enable users to easily analyze minimum cuts, Figure 6.3(b) colors the Voronoi cells [30] of each vertex by a partition specific color, blue for the vertices in S and white for all other vertices in S'. The Voronoi cell of a vertex is the area that is 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 can not 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. So instead of increasing the capacity of such an edge, the flow along this path could be improved. So an edge ending in S (blue region) can not 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 and

example, the edge (A, C) with values "3/3" is considered. But 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 chapter extends the construction of a minimum cut by adding a third set $T \subset V$ to the partition. Figure 6.3(c) and 6.3(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. So the new partition is $P = (S, R, T)$ (blue / white / black regions) with disjoint sets $S, R, T \subset V$, and $S \cup R \cup T = V$, and $s \in S$ and $t \in T$.

Analog to before, all edges starting in T (black region) can not 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 can not 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. So the overall flow would not increase, hence the edge is no bottleneck. Out of analog reasons, edges starting in R (white region) also can not be bottleneck edges.

The only edges left that can be bottlenecks are edges starting in S (blue region) and leading to T (black region). As can be seen in Figure 6.3(d), the edge (B, D) with values "2/2" is of this kind and is a bottleneck edge, so increasing the capacity of this edge would lead to a larger maximum flow (compare Figure 6.4(c)). In general, not only some, but all edges leading from S (blue region) to T (black region) are 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. So 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 $f(v, w) < c'(v, w)$, so the edge (v, w) will be 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, so all edges from S (blue region) to T (black region) are bottleneck edges.

So the overall approach works by first performing a max-flow calculation followed by two separate breadth-first-searches in the residual network starting forwards 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 $O(|V| + |E|)$, meaning that the complexity and limitations of the overall approach are dependent only on which max-flow algorithm is chosen, as described above.

6.3.2 Results

Ensemble Visualization

The visualization in Figure 6.3(d) (compare Figure 6.4(a)) intuitively shows the different regions of a network with maximum flow. By construction, there is enough capacity left to increase the flow from the source to the vertices in the blue region. On the other hand, there is enough capacity left to increase the flow to the network's sink from one of the vertices in the black region. The white region consists of the remaining vertices that are not able to increase the flow in either direction. So to identify bottlenecks for a single network, users can intuitively consider transitions

from a blue to a black region. Since multiple Voronoi cells form regions, in general there is not just a single bottleneck edge, meaning a single transition, but a closed front of bottleneck transitions. In addition to that, the different regions deliver insight into whether vertices and edges are in front of the overall bottleneck front or behind it.

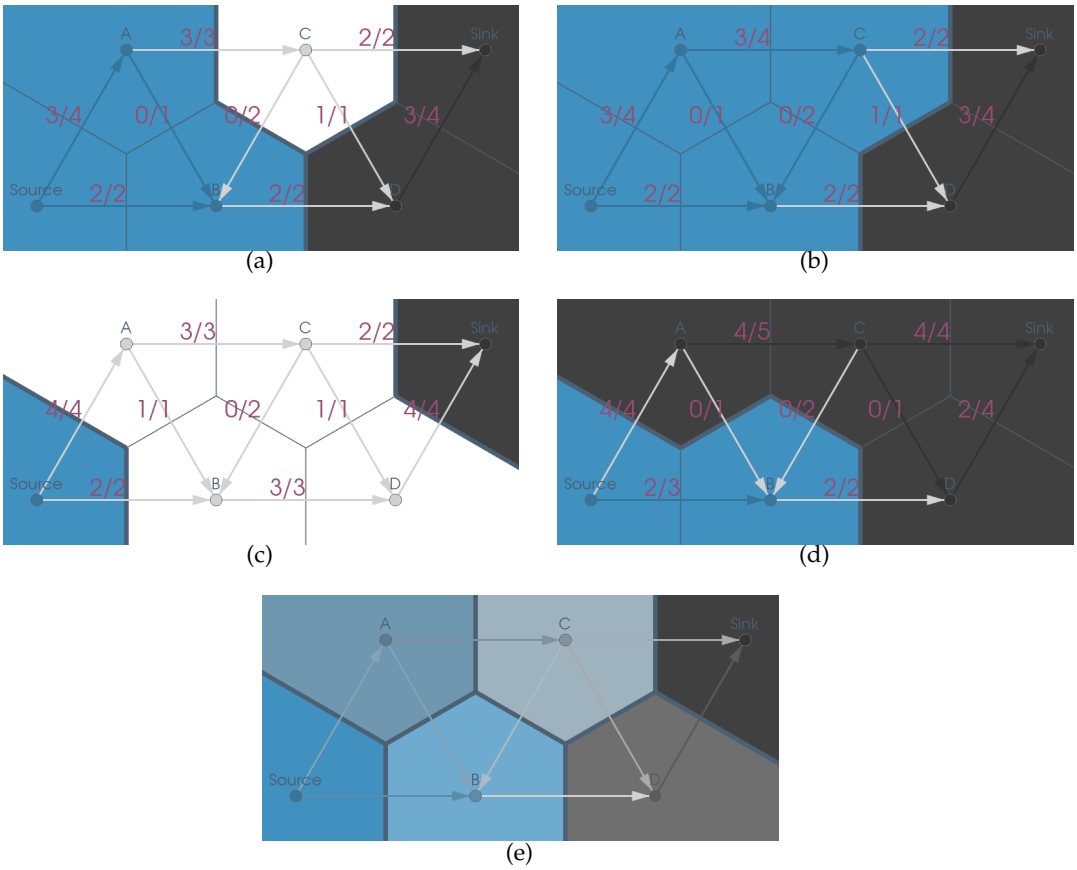


FIGURE 6.4: Ensemble of the same network with four different exemplary capacity configurations leading to four different maximum flows and minimum cuts (images (a)-(d)). Transitions from a blue to a black region are bottlenecks. The samples of the ensemble are accumulated and averaged, thereby giving an overview of stable regions (fully blue or black) and the progress of the bottlenecks (image (e)).

In real world application there often is the need to compare results for different layouts or configurations. Here, the capability to get an overview of different capacity configurations for the same underlying graph is desired. To tackle this problem, Figure 6.4(a)-6.4(d) show an ensemble of networks, their maximum flow and the different regions of the extended minimum cut. The different configurations were chosen arbitrarily to demonstrate a variety of possible scenarios. The individual Voronoi cells are accumulated and averaged as shown in Figure 6.4(e).

Fully blue or black regions in the accumulated image indicate a completely stable location in front of or behind the bottleneck, respectively. Fully white regions never occur in any other (blue or black) region. The advantage of having chosen a blue / white / black color theme is, that these colors accumulate uniquely. A region that is bluer appears more often before the bottleneck front in the individual ensemble members, while blacker regions appear more often behind the bottleneck. The same holds for whiter regions in the accumulated image that appear more often in neither of the two (blue or black) regions.

Fully blue, black or white regions indicate a stable behavior of the respective vertices in all ensemble members regardless of their different capacity configurations. In contrast to that, regions with a not so well defined color indicate unstable regions, meaning a divergent behavior for the different ensemble members. Also, a fully blue to fully black transition indicates a stable bottleneck regardless of the different ensemble configurations. Overall, the presented visualization methodologies enable an intuitive identification of network bottlenecks and the analysis of bottleneck propagation for an ensemble of a network with different capacity configurations.

Scalability

To demonstrate the scalability of the method presented in Section 6.3.1, a larger randomly generated flow network is analyzed in Figure 6.5. The source of the flow network is on the left side of the images, and the blue region in Figure 6.5(b) is where flow can travel freely from the source. On the other side, the sink is on the right side of the images, and the black region is from where flow can travel freely to the sink. The edges leading from the blue to the black region form the bottleneck front. Because the presented flow network is quite large, the vertex and edge labels in Figure 6.5(a) were intentionally omitted to not further overload the image. Even though, the visualization in Figure 6.5(b) is able to clearly shows the bottleneck front of interest.

6.4 Cascaded Bottlenecks

This section focuses on cascaded bottlenecks, multiple bottleneck fronts occurring sequentially after another. A method to analyze those bottlenecks in planar flow networks is presented and its effectiveness is demonstrated.

6.4.1 Methods

Section 6.3.1 has demonstrated 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 6.6(a) shows, there are cases where there are no direct edges leading from S to T since the blue and the black region are separated spatially. These flow networks do not have a single bottleneck front but cascaded bottlenecks sequentially following one another. The question arises on 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 6.6(b)). The SCCs can be calculated efficiently by Tarjan's algorithm [109] 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 $i \in \{1, ..., k\}$. So 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 (see Section 6.3.1), its SCCs indicate candidates for the bottlenecks. Additional flow can move freely within one SCC, while crossing the boundary between two neighboring SCCs might rise the need to increase the capacity value of this particular edge (see Figure 6.6(b)).

The boundaries of the SCCs from Figure 6.6(b) are used to show the cascaded bottlenecks of interest in Figure 6.7(a). Since there is no single edge with a capacity value that could be increased to increase the overall flow, the question arises on

which capacity values to increase. To tackle this question, the forward graph (see Figure 6.7(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 $G_F = (V, E_F, w_F)$. So 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. And 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 6.8). These distances will now be called *forward distance* and *backward distance*, respectively. This can be done efficiently by Dijkstra's algorithm [16] 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 be able to transport additional flow. Since shortest paths are used, it is ensured 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.

6.4.2 Results

The methodologies presented in Section 6.4.1 can be utilized to develop a method to interactively analyze cascaded bottlenecks in planar flow networks (see Figure 6.9). Here, 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 (see Section 6.3.1). Additional flow can travel freely on these paths, while the incoming and outgoing edges themselves have the potential to be bottlenecks. In contrast to that, by construction, a transition from

a SCC of one color to a 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 as color-coded components in Figure 6.9.

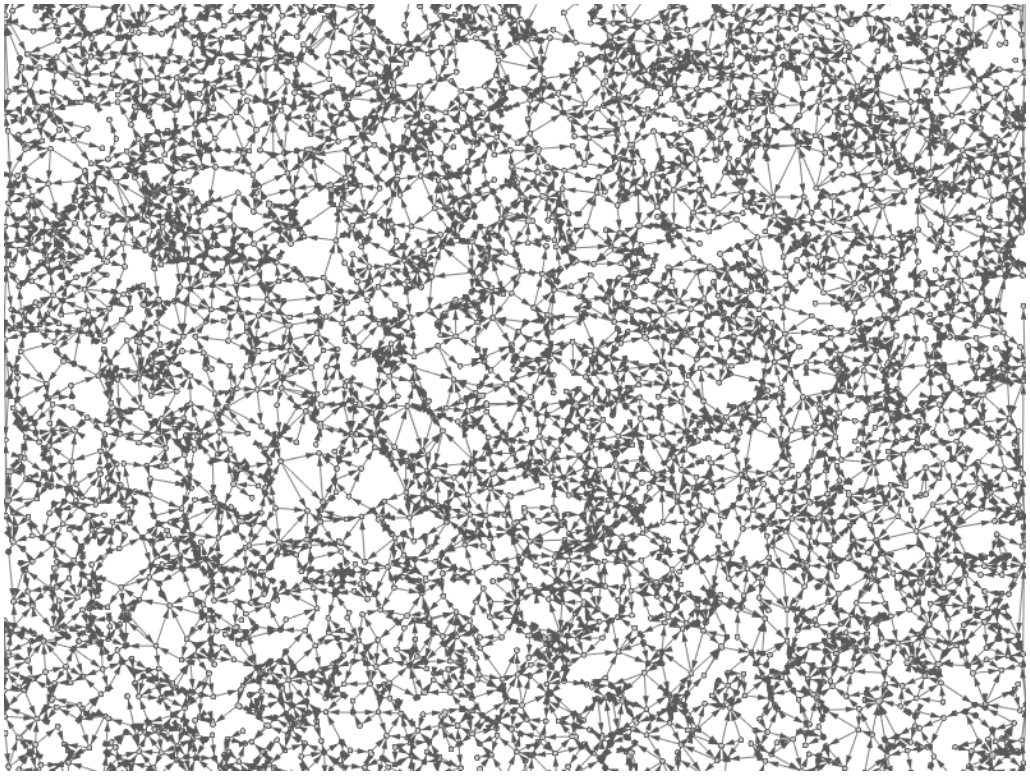
The augmenting paths within each SCC are shown as spline segments in Figure 6.9. Each segment can by selected by the user. Though, not all possible segments are shown. To enhance the usability and restrict the selection to meaningful segments, the segments are filtered and unwanted segments are discarded. 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 therefore 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. This way, 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 Fig 6.9 per row and demonstrates the effectiveness of the presented interactive method to analyze cascaded bottlenecks.

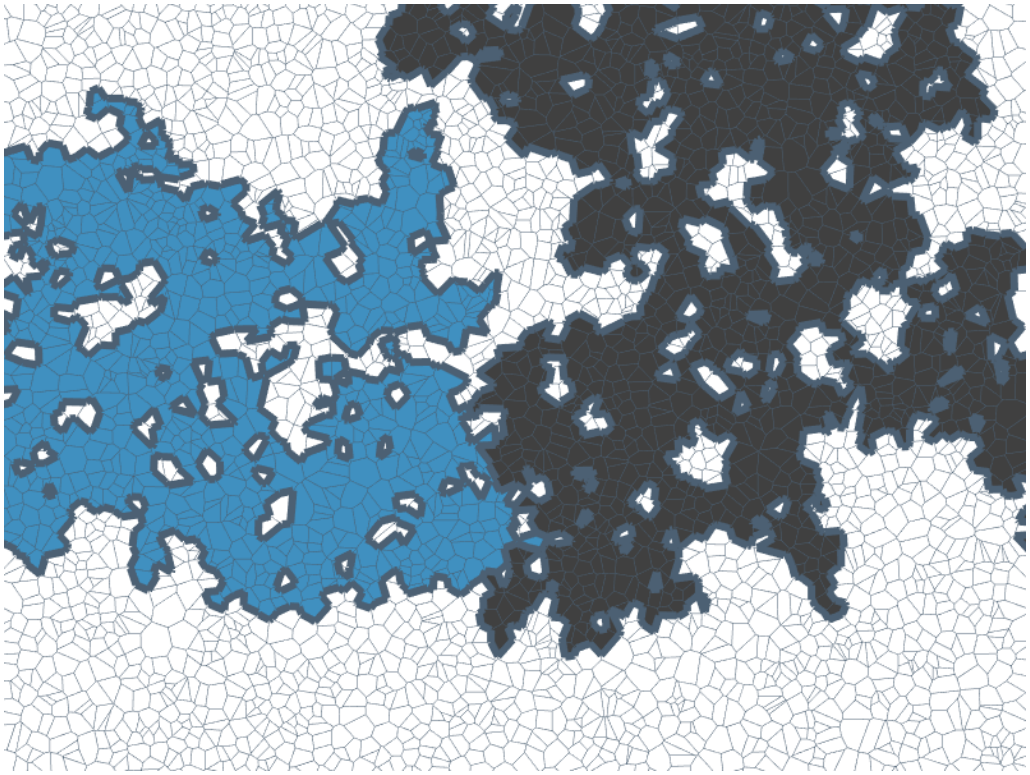
6.5 Conclusion

This chapter has introduced a novel approach to visualize and compare bottlenecks in flow networks for real world applications like cyber physical manufacturing systems. Therefore, product flows and constraints of a manufacturing system were mapped to a network. This chapter 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 allowed a fast and intuitive identification of bottleneck transitions in a flow network.

As the planning of production systems involves the comparison of different factory settings, various configurations of a network need to be compared. This chapter has presented an ensemble visualization technique based on the visualization of extended minimum cuts. The presented examples demonstrated, that the ensemble visualization technique indicates stable and divergent bottleneck transitions and regions, thereby enabling users to visually identify and analyze similarities and differences of bottlenecks for an ensemble of flow network configurations.

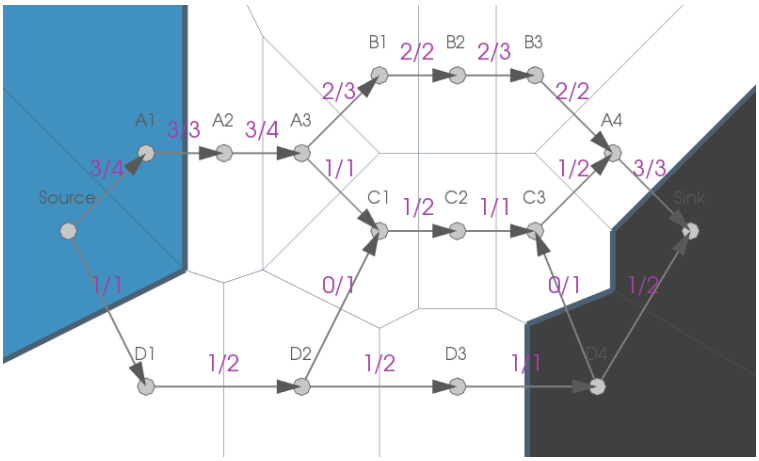


(a)

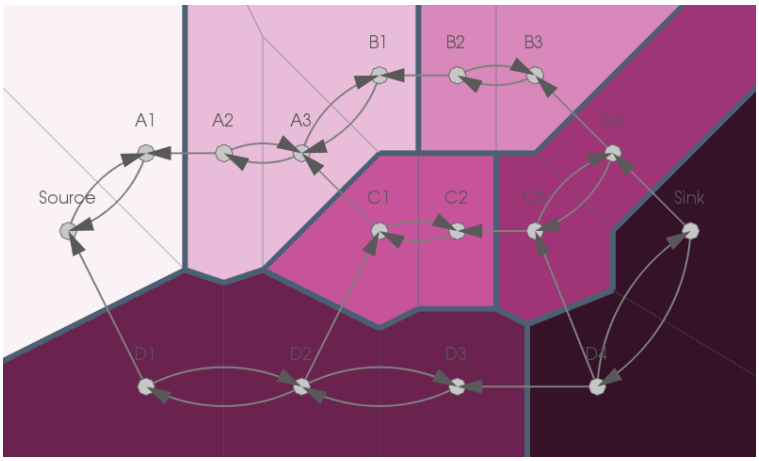


(b)

FIGURE 6.5: Randomly generated planar flow network with 5000 vertices and 10000 directed edges (image (a)), and its extended minimum cut (image (b)). Edges going from the blue to the black region are bottleneck edges, thereby forming a bottleneck front.

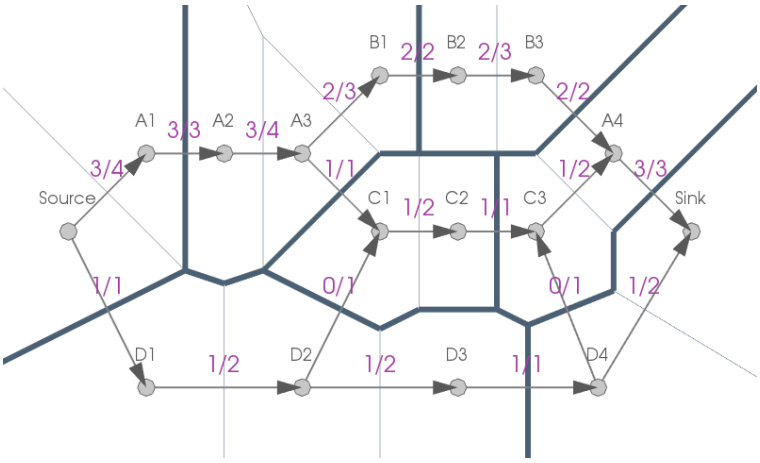


(a)

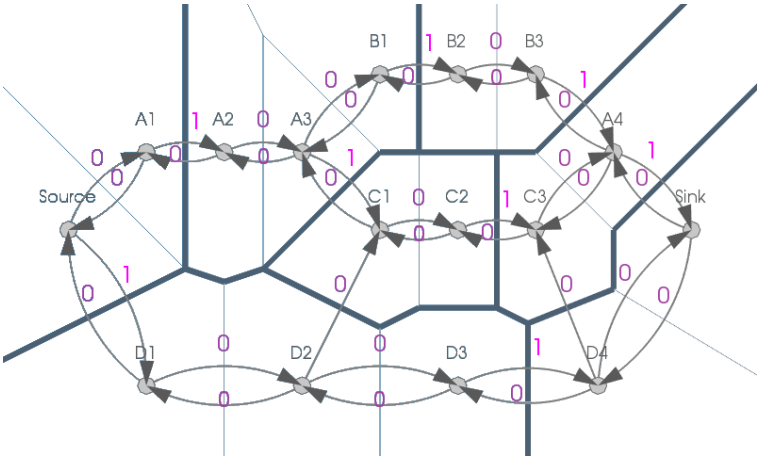


(b)

FIGURE 6.6: 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)).

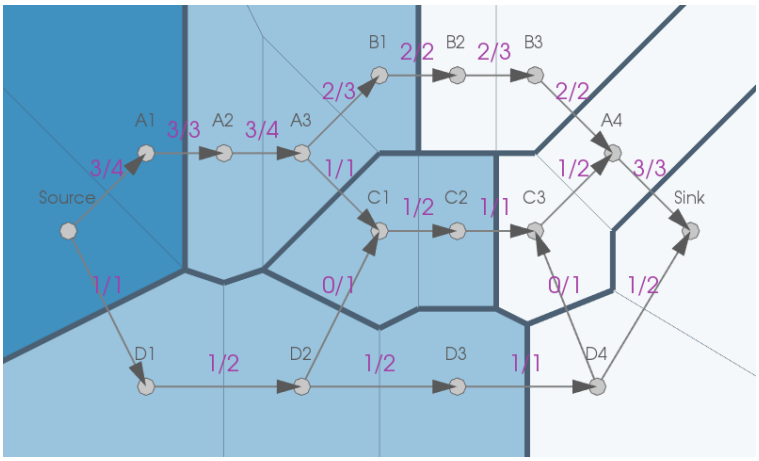


(a)

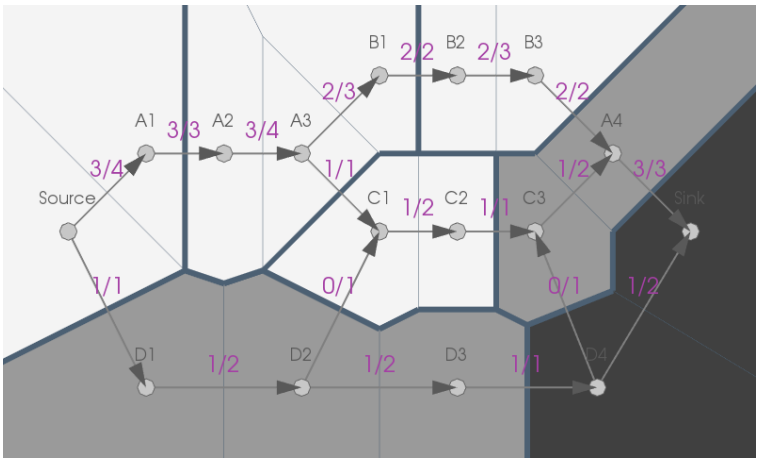


(b)

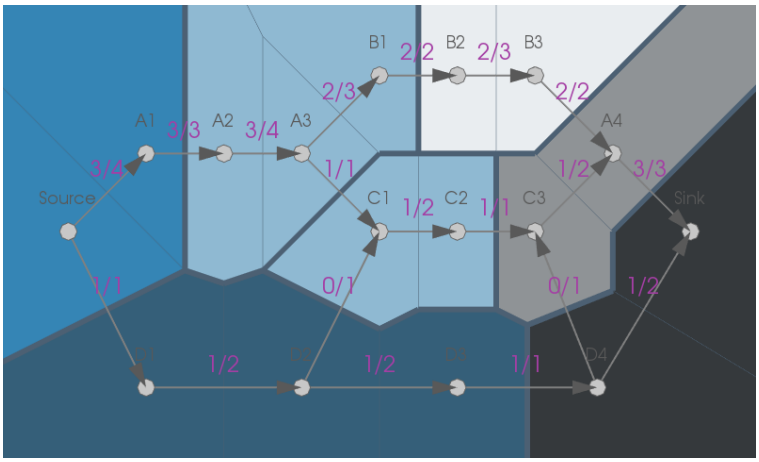
FIGURE 6.7: The flow network from Figure 6.6(a) with its cascaded bottleneck candidates (image (a)). This flow network is used for the construction of the forward graph (image (b)).



(a)



(b)



(c)

FIGURE 6.8: 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 (rich color) to 2 (pale color). By utilizing different color channels both distances can be displayed simultaneously in an unambiguous way (image (c)).

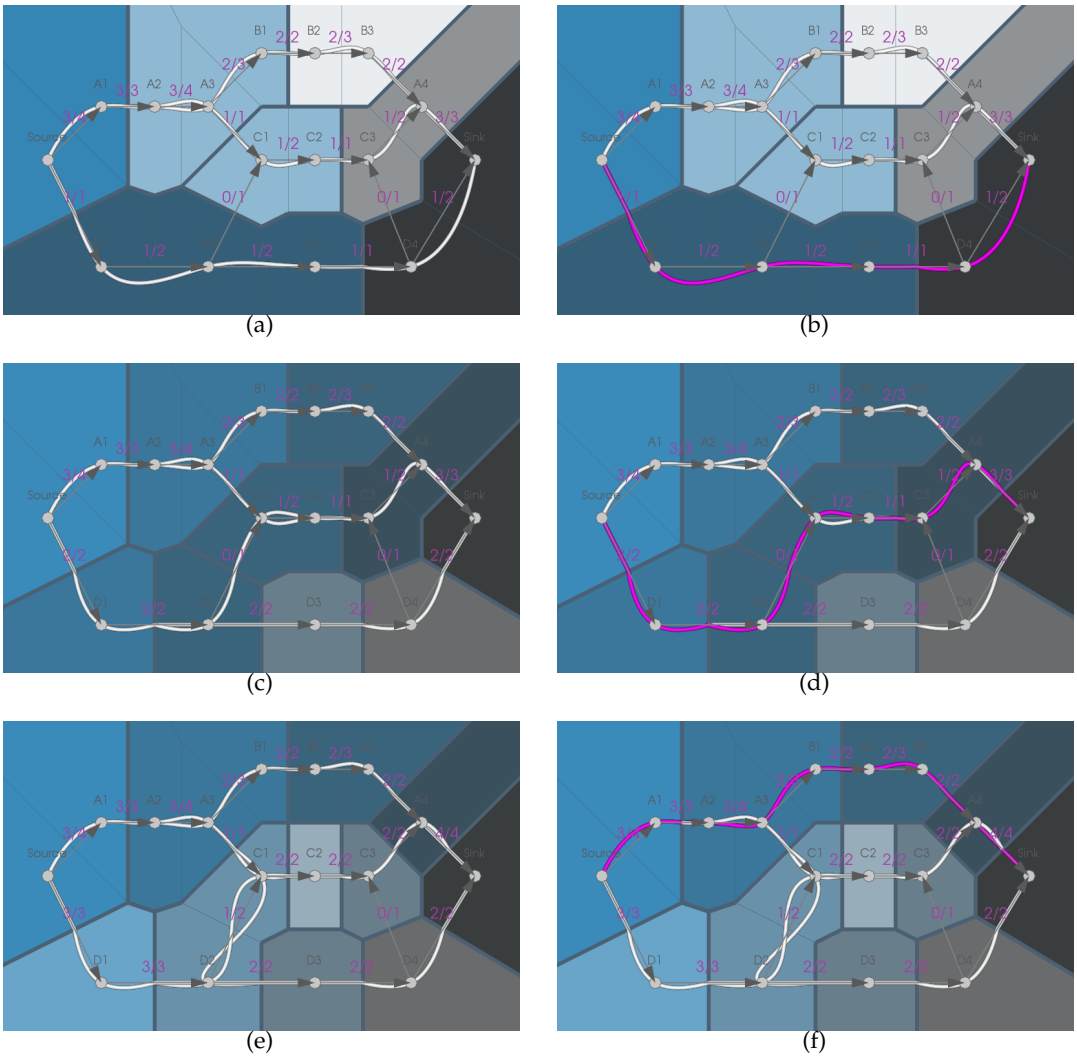


FIGURE 6.9: Iterations of the feedback loop. The current flow network and all its filtered path segments are displayed as splines (white) in the left images. The user can select path segments (magenta) and thereby construct a path from source to sink (right images). This path is applied by increasing edge capacities on the path accordingly and calculating the increased overall flow for the next iteration below. Only the capacities of edges leading from one to another component might need to be adjusted. The component colors indicate the relative effort to send flow from the source to the component (blue color), or from the component to the sink (black color). Richer colors indicate relatively less effort and thereby less bottlenecks that need to be overcome to increase the overall flow.

Chapter 7

Discussion and Conclusion

This chapter will give a summary of the contributions and resulting implications in the area of visual analysis for graphs, networks and and flows.

7.1 Summary

Graphs and flow networks are important concepts to model a huge variety of real world problems. Their increasing sizes and complexities raise the need for efficient and intuitive techniques to allow the analysis and decision making for domain scientists from different domains.

Based on the visual analysis principle *Analyse First - Show the Important - Zoom, Filter and Analyse Further - Details on Demand* this work presents a variety of visual analysis methodologies that are suitable to understand and examine graphs and network flows. In particular, the contributions of this work are listed below.

Flow Tracking in Software-Defined Networking : In order to understand the flow patterns in software-defined networks this work offers an intuitive overview of the SDN hierarchy and the underlying packet flow. The ability to track packets through the SDN and to interlink multiple views of the SDN forms an interactive analysis tool that is successfully applied to a simulated dataset.

Visual Analysis of Cyber-Physical Production Systems : The behavior of cyber-physical production systems can be hard to understand without proper visualization. Therefore, this work generates visualizations of aggregated views that capture

an entire production system as well as specific characteristics of individual data features. To show the applicability and effectiveness of the presented methodologies, an exemplary production system is simulated and analyzed.

Analysis of High-dimensional Data : Besides the topology of graphs and flow networks, domain scientists are also interested in a variety of properties of graphs and networks. Those properties can often be high-dimensional. This work offers a scale-invariant measure based on Pareto optimality that provides a visual encoding of the environment of a Pareto front to enable an enhanced visual inspection.

Graph Extraction using Fast 3D Thinning : A variety of applications require data transformation steps to extract the graph or flow network of interest. Therefore, this work presents a novel thinning approach for 3D image data that can be utilized to generate graphs and flow networks. This new lookup-table approach is fast and robust which was demonstrated by multiple examples.

Ensemble Visualization of Network Bottlenecks : Bottlenecks are an important aspect when considering flow networks. In order to compare the bottlenecks resulting from different flow network settings, this work introduces intuitive visual mechanisms enabling domain experts to visually analyze stable regions of a network and identify critical transitions. Those transitions form a varying bottleneck front for different configurations of network restraints. To tackle this challenge, this work enhances the comparability of different network configurations by utilizing ensemble visualization techniques.

Visual Analysis of Cascaded Network Bottlenecks : As bottlenecks in flow networks can form different fronts that follow one another, users require a mechanism to understand these cascaded bottlenecks. They can lead to a variety of combinations to improve a network which the need to be analyzed. This work introduces a novel approach to identify cascaded bottlenecks and evaluate their properties. The presented interactive technique enables users to explore and analyze the different combinations to improve planar networks.

7.2 Implications

This work showed the importance and the potential of visual analysis for graphs, networks and flows. The need of visual analysis is driven by the permanently increasing size and complexity of datasets as well as the number of applications that benefit from these techniques. The presented work showed that different domains and applications have varying requirements and demand a suitable selection of analysis techniques. Therefore, this work offered a variety of visual analysis techniques that are suitable for graphs and flow networks.

As visual analysis consists of multiple steps, each of these steps need to be implemented, thus users from different domains can make reliable decisions based on the gained insight. This work provided suitable solutions for different steps of the visual analysis pipeline to gain this insight, covering data transformation, visual mapping, parameter refinement and analysis, model building and visualization as well as user interaction.

In its entirety this work and the presented concepts and techniques form a framework that enriches domain scientists with new visual analysis tools and methodologies that help users in analyzing their data and gain insight from the underlying structures. The applicability and effectiveness of the presented approaches were demonstrated by tackling different domains and applications in this work while preserving the generality to be applicable to further domains.

Curriculum Vitae

Schulausbildung

1992–2001 **Abitur am Friedrich-Spee-Gymnasium, Geldern**

Wehrdienst

2001–2002 **Moritz-von-Nassau-Kaserne, Emmerich**

Studium

2002–2011 **Diplom an der RWTH Aachen University**

Berufstätigkeit

2011–2012 **RWTH Aachen University**

2013–2014 **Technische Universität Kaiserslautern**

Promotion

2014–2018 **Doktorgrad an der Technischen Universität Kaiserslautern**

Bibliography

[1] M. Ahmed and R. Ward. "A rotation invariant rule-based thinning algorithm for character recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.12 (2002), pp. 1672–1678. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2002.1114862.

[2] Alfred V. Aho, John E. Hopcroft, and Jeffrey Ullman. *Data Structures and Algorithms*. 1st. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1983. ISBN: 0201000237.

[3] Carlo Arcelli and Gabriella Sanniti di Baja. "A One-Pass Two-Operation Process to Detect the Skeletal Pixels on the 4-Distance Transform". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 11.4 (1989), pp. 411–414.

[4] B. Bach et al. "A Review of Temporal Data Visualizations Based on Space-Time Cube Operations". In: *In USENIX LISA*. 2000, pp. 305–317.

[5] Benjamin Bach, Emmanuel Pietriga, and Jean-Daniel Fekete. "GraphDiaries: Animated Transitions and Temporal Navigation for Dynamic Networks". In: *IEEE Transactions on Visualization Computer Graphics* 20.5 (2014), pp. 740–754.

[6] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. *Gephi: An Open Source Software for Exploring and Manipulating Networks*. 2009.

[7] Maryam Booshehrian et al. "Vismon: Facilitating Analysis of Trade-Offs, Uncertainty, and Sensitivity In Fisheries Management Decision Making". In: *Computer Graphics Forum* (2012). ISSN: 1467-8659. DOI: 10.1111/j.1467-8659.2012.03116.x.

[8] Gunilla Borgefors, Ingela Nystrm, and Gabriella Sanniti Di Baja. "Computing skeletons in three dimensions". In: *Pattern Recognition* 32.7 (1999), pp. 1225 – 1236.

[9] Ljudmilla Borisjuk et al. “Integrating data from biological experiments into metabolic networks with the DBE information system”. In: *In Silico Biology* 5.2 (2004), pp. 93–102.

[10] Ilya Boyandin, Enrico Bertini, and Denis Lalanne. “A Qualitative Study on the Exploration of Temporal Changes in Flow Maps with Animation and Small-Multiples”. In: *Computer Graphics Forum* 31.3pt2 (2012), pp. 1005–1014.

[11] Ulrik Brandes and Steven R. Corman. “Visual Unrolling of Network Evolution and the Analysis of Dynamic Discourse”. In: *Information Visualization* 2.1 (2003), pp. 40–50.

[12] Ulrik Brandes, Sabine Cornelsen, and Dorothea Wagner. “How to Draw the Minimum Cuts of a Planar Graph”. In: *Graph Drawing: 8th International Symposium Proceedings*. Springer Berlin Heidelberg, 2001, pp. 89–119.

[13] Lothar Braun et al. “Flow-inspector: a framework for visualizing network flow data using current web technologies”. In: *Computing* 96.1 (2014), pp. 15–26.

[14] Nathaniel Cesario, Alex Pang, and Lisa Singh. “Visualizing node attribute uncertainty in graphs”. In: *Proc. SPIE* 7868 (2011), 78680H–78680H–13.

[15] Moritz Chemnitz et al. “SOPRO - Advancements in the self-organising production”. In: *IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2010*. Ed. by ETFA 2010. Piscataway, NJ: IEEE, 2010, pp. 1–4. ISBN: 978-1-4244-6850-8. DOI: 10.1109/ETFA.2010.5641198.

[16] Thomas H. Cormen et al. *Introduction to Algorithms, Third Edition*. 3rd. The MIT Press, 2009. ISBN: 0262033844, 9780262033848.

[17] Nicu D. Cornea, Deborah Silver, and Patrick Min. “Curve-Skeleton Properties, Applications, and Algorithms”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.3 (May 2007), pp. 530–548.

[18] Michel Couprie, Nivando Bezerra, and Gilles Bertrand. “A Parallel Thinning Algorithm for Grayscale Images”. In: *Discrete Geometry for Computer Imagery*. Ed. by Rocio Gonzalez-Diaz, Maria-Jose Jimenez, and Belen Medrano. Vol. 7749. Springer Berlin Heidelberg, 2013, pp. 71–82.

[19] F. Doil et al. "Augmented Reality for Manufacturing Planning". In: *Proceedings of the Workshop on Virtual Environments 2003*. ACM, 2003, pp. 71–76.

[20] Denis V. Dorozhkin et al. "Coupling of interactive manufacturing operations simulation and immersive virtual reality". In: *Virtual Reality* 16.1 (2012), pp. 15–23. ISSN: 1359-4338. DOI: 10.1007/s10055-010-0165-7.

[21] Ulrich Eckhardt and Gerd Maderlechner. "Invariant Thinning." In: *IJPRAI* 7.5 (1993), pp. 1115–1144.

[22] Jack Edmonds and Richard M. Karp. "Theoretical Improvements in Algorithmic Efficiency for Network Flow Problems". In: *J. ACM* 19.2 (1972), pp. 248–264.

[23] P. Elias, A. Feinstein, and C. Shannon. "A note on the maximum flow through a network". In: *Information Theory, IEEE Transactions on* 2.4 (1956), pp. 117–119.

[24] Gurdal Ertek et al. "Visual and analytical mining of transactions data for production planning and marketing". In: *Intelligent Manufacturing Systems*. 2004, pp. 848–859.

[25] Petri Eskelinen and Kaisa Miettinen. "Trade-off analysis approach for interactive nonlinear multiobjective optimization". In: *OR Spectrum* (2011).

[26] Gerald Farin, ed. *Curves and Surfaces for CAGD. A Practical Guide.: A Practical Guide*. 5th ed. Morgan Kaufmann Series in Computer Graphics and Geometric Modelling, 2001.

[27] Allan M. Feldman and Roberto Serrano. *Welfare Economics and Social Choice Theory*. 2. Springer, 2006.

[28] Mark Fleischer. "Scale Invariant Pareto Optimality: A Meta–formalism for Characterizing and Modeling Cooperativity in Evolutionary Systems". In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*. GECCO '05. ACM, 2005, pp. 233–240.

[29] L. R. Ford and D. R. Fulkerson. "Maximal Flow through a Network." In: *Canadian Journal of Mathematics* 8 (1956), pp. 399–404.

[30] Steven Fortune. “Voronoi Diagrams and Delaunay Triangulations”. In: *Handbook of Discrete and Computational Geometry*. Ed. by Jacob E. Goodman and Joseph O’Rourke. CRC Press, Inc., 1997, pp. 377–388.

[31] O. N. Foundation. *Software-Defined Networking (SDN) Definition*. “https://www.opennetworking.org/sdn-resources/sdn-definition". 2016.

[32] V. T. Guimares et al. “A Survey on Information Visualization for Network and Service Management”. In: *IEEE Communications Surveys Tutorials* 18.1 (2016), pp. 285–323.

[33] Steffen Hadlak, Heidrun Schumann, and Hans-Jrg Schulz. “A Survey of Multi-faceted Graph Visualization”. In: *Eurographics Conference on Visualization (EuroVis) - STARs*. Ed. by R. Borgo, F. Ganovelli, and I. Viola. The Eurographics Association, 2015. DOI: 10.2312/eurovisstar.20151109.

[34] Steven Halim. *VisuAlgo*. “https://visualgo.net/maxflow". Accessed: 2017-01-26.

[35] Y. M. Y. Hasan and L. J. Karam. “Morphological text extraction from images”. In: *IEEE Transactions on Image Processing* 9.11 (2000), pp. 1978–1983. ISSN: 1057-7149.

[36] Ivan Herman, Guy Melançon, and M. Scott Marshall. “Graph Visualization and Navigation in Information Visualization: A Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 6.1 (Jan. 2000), pp. 24–43. ISSN: 1077-2626. DOI: 10.1109/2945.841119.

[37] H.Gouraud. *Continuous shading of curved surfaces*. 5th ed. Rosalee Wolfe, 1998.

[38] C. J. Hilitch. “Linear Skeletons From Square Cupboards”. In: *Machine Intelligence 4*. Ed. by B. Meltzer and Donald Michie. Edinburgh University Press, 1969, p. 403.

[39] Danny Holten. “Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data”. In: *IEEE Transactions on Visualization and Computer Graphics* 12.5 (Sept. 2006), pp. 741–748.

[40] Danny Holten and Jarke J. van Wijk. "Force-directed Edge Bundling for Graph Visualization". In: *Proceedings of the 11th Eurographics / IEEE - VGTC Conference on Visualization*. EuroVis'09. Berlin, Germany: The Eurographs Association, 2009, pp. 983–998.

[41] Danny Holten and Jarke J. Van Wijk. "Visual comparison of hierarchically organized data". In: *Comput. Graph. Forum* (2008).

[42] H. Homann. *Implementation of a 3D thinning algorithm*. "http://hdl.handle.net/1926/1292". Oct. 2007.

[43] H. Homann. "Implementation of a 3D thinning algorithm". In: (Oct. 2007).

[44] Rein-Lien Hsu, M. Abdel-Mottaleb, and A. K. Jain. "Face detection in color images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.5 (2002), pp. 696–706. ISSN: 0162-8828. DOI: 10.1109/34.1000242.

[45] Christopher Humphries et al. "ELVIS: Extensible Log VISualization". In: *Proceedings of the Tenth Workshop on Visualization for Cyber Security*. VizSec '13. ACM, 2013, pp. 9–16.

[46] C. Hurter et al. "Bundled Visualization of DynamicGraph and Trail Data". In: *IEEE Transactions on Visualization and Computer Graphics* 20.8 (2014), pp. 1141–1157.

[47] Hyperglance Inc. *hyperglance*. "https://www.hyperglance.com/". 2016.

[48] R. Ilsen, H. Meissner, and J. C. Aurich. "Virtual Test Field for Sustainability Assessment of Cybertronic Production Systems". In: *43rd Proceedings of the North American Manufacturing Research Institution of SME*. Ed. by NAMRI/SME. 2015.

[49] M. Itoh et al. "Visual Exploration of Changes in Passenger Flows and Tweets on Mega-City Metro Network". In: *IEEE Transactions on Big Data* 2.1 (2016), pp. 85–99.

[50] J. Jaffe. "Bottleneck Flow Control". In: *IEEE Transactions on Communications* 29.7 (1981), pp. 954–962.

[51] Manar Jammal et al. “Software defined networking: State of the art and research challenges”. In: *Computer Networks* 72 (2014), pp. 74 –98. DOI: http://dx.doi.org/10.1016/j.comnet.2014.07.004. URL: http://www.sciencedirect.com/science/article/pii/S1389128614002588.

[52] Hans J. Johnson et al. *The ITK Software Guide*. Third. *In press*. Kitware, Inc. 2013. URL: http://www.itk.org/ItkSoftwareGuide.pdf.

[53] H. Kagermann, W. Wahlster, and J. Helbig. *Recommendations for implementing the strategic initiative INDUSTRIE 4.0: Final report of the Industrie 4.0 Working Group*. acatech – National Academy of Science and Engineering, 2013.

[54] Daniel A. Keim et al. “Visual Data Mining”. In: ed. by Simeon J. Simoff, Michael H. Böhlen, and Arturas Mazeika. Berlin, Heidelberg: Springer-Verlag, 2008. Chap. Visual Analytics: Scope and Challenges, pp. 76–90. ISBN: 978-3-540-71079-0. DOI: 10.1007/978-3-540-71080-6_6.

[55] M. Kikolski. “Identification of production bottlenecks with the use of Plant Simulation software”. In: *Ekonomia i Zarzadzanie* 8.4 (2017), pp. 103–112.

[56] C. Kirbas and F.K.H. Quek. “Vessel extraction techniques and algorithms: a survey”. In: *Bioinformatics and Bioengineering, 2003. Proceedings. Third IEEE Symposium on*. 2003, pp. 238–245.

[57] S Klamt and A von Kamp. “An application programming interface for CellNetAnalyzer”. In: *BioSystems* 105 (2011), pp. 162–168.

[58] S Klamt, J Saez-Rodriguez, and E D Gilles. “Structural and functional analysis of cellular networks with CellNetAnalyzer”. In: *BMC Systems Biology* 1 (2 2007), open access.

[59] D. Kreutz et al. “Software-Defined Networking: A Comprehensive Survey”. In: *Proceedings of the IEEE* 103.1 (2015), pp. 14–76.

[60] Louisa Lam, Seong-Whan Lee, and Ching Y. Suen. “Document Image Analysis”. In: ed. by Lawrence O’Gorman and Rangachar Kasturi. IEEE Computer Society Press, 1995. Chap. Thinning Methodologies&Mdash;a Comprehensive Survey, pp. 61–77.

[61] Robert S. Laramee and Robert Kosara. "Challenges and Unsolved Problems". In: *Human-Centered Visualization Environments: GI-Dagstuhl Research Seminar, Dagstuhl Castle, Germany, March 5-8, 2006, Revised Lectures*. Ed. by Andreas Kerren, Achim Ebert, and Jörg Meyer. Springer Berlin Heidelberg, 2007, pp. 231–254.

[62] Bongshin Lee et al. "Task Taxonomy for Graph Visualization". In: *Proceedings of the 2006 AVI Workshop on BEyond Time and Errors: Novel Evaluation Methods for Information Visualization*. BELIV '06. Venice, Italy: ACM, 2006, pp. 1–5. ISBN: 1-59593-562-2. DOI: 10.1145/1168149.1168168.

[63] Chi G. Lee and Sang C. Park. "Survey on the virtual commissioning of manufacturing systems". In: *Journal of Computational Design and Engineering* 1.3 (2014), pp. 213 –222.

[64] Ta-Chih Lee, Rangasami L. Kashyap, and Chong-Nam Chu. "Building Skeleton Models via 3-D Medial Surface/Axis Thinning Algorithms". In: *CVGIP: Graph. Models Image Process.* 56.6 (1994), pp. 462–478. ISSN: 1049-9652.

[65] Paulo Leitão. "Agent-based distributed manufacturing control: A state-of-the-art survey". In: *Engineering Applications of Artificial Intelligence* 22.7 (2009), pp. 979–991. DOI: 10.1016/j.engappai.2008.09.005.

[66] Paulo Leitão and Francisco Restivo. "ADACOR: A holonic architecture for agile and adaptive manufacturing control". In: *Computers in Industry* 57.2 (2006), pp. 121–130. ISSN: 01663615. DOI: 10.1016/j.compind.2005.05.005.

[67] Bingdong Li et al. "Review: A Survey of Network Flow Applications". In: *J. Netw. Comput. Appl.* 36.2 (Mar. 2013), pp. 567–581. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2012.12.020.

[68] Shixia Liu et al. "StoryFlow: Tracking the Evolution of Stories". In: *IEEE Transactions on Visualization and Computer Graphics* 19.12 (Dec. 2013), pp. 2436–2445.

[69] LiveAction. *NetFlow Visualizations*. "http://www.liveaction.com/netflow-visualization/". 2016.

[70] C.-T. Lu, A. P. Boedihardjo, and J. Zheng. "AITVS: Advanced Interactive Traffic Visualization System". In: *IEEE Computer Society*, 2006, p. 167.

[71] C.Min Ma and Milan Sonka. "A Fully Parallel 3D Thinning Algorithm and Its Applications". In: *Computer Vision and Image Understanding* 64.3 (1996), pp. 420 –433.

[72] R. C. Malak and J. C. Aurich. "Software Tool for Planning and Analyzing Engineering Changes in Manufacturing Systems". In: *Procedia CIRP* 12 (2013), pp. 348–353. ISSN: 22128271. DOI: 10.1016/j.procir.2013.09.060.

[73] R. C. Malak, X. Yang, and J. C. Aurich. "Analysing and Planning of Engineering Changes in Manufacturing Systems". In: *The 44th CIRP Conference on Manufacturing Systems*. Ed. by N. A. Duffie and M. F. DeVries. 2011.

[74] Antoine Manzanera et al. "A unified mathematical framework for a compact and fully parallel n-D skeletonization procedure". In: *Proc. SPIE, Vol. 3811, Vision Geometry VIII*. July 1999, pp. 57–68.

[75] *Matlab Central - File Exchange*. "http://www.mathworks.com/matlabcentral/fileexchange/43400-skeleton3d". Accessed: 2016-02-12.

[76] Nasrullah Memon et al. "Social Network Data Mining: Research Questions, Techniques, and Applications." In: *Data Mining for Social Network Data*. Ed. by Nasrullah Memon et al. Vol. 12. Annals of Information Systems. Springer, 2010, pp. 1–7. ISBN: 978-1-4419-6286-7.

[77] Samira S. Mersa and Ahmed M. Darwish. "A new parallel thinning algorithm for gray scale images". In: *IEEE Nonlinear Signal and Image Proc. Conf.* 1999, pp. 409–413.

[78] L. Monostori, J. Váncza, and S.R.T. Kumara. "Agent-Based Systems for Manufacturing". In: *Annals of the CIRP* 55.2 (2006), pp. 697–720.

[79] László Monostori. "Cyber-physical Production Systems: Roots, Expectations and R&D Challenges". In: *Procedia CIRP* 17 (2014), pp. 9–13. ISSN: 22128271. DOI: 10.1016/j.procir.2014.03.115.

[80] Sunit Kumar Nandi. "Topology generators for Software Defined Network testing". In: *International Conference on Electrical, Electronics, and Optimization Techniques* (2016).

[81] Quan Nguyen, Peter Eades, and Seok-Hee Hong. "StreamEB: Stream Edge Bundling". In: *Proceedings of the 20th International Conference on Graph Drawing*. GD'12. Redmond, WA, 2013, pp. 400–413.

[82] SMYJ Oh et al. "Process-driven BIM-based optimal design using integration of EnergyPlus, genetic algorithm, and pareto optimality". In: *Proceedings of the IBPSA building simulation 2011 conference, Sydney, Australia* (2011), pp. 894–901.

[83] V. Peysakhovich, C. Hurter, and A. Telea. "Attribute-driven edge bundling for general graphs with applications in trail analysis". In: *2015 IEEE Pacific Visualization Symposium (PacificVis)*. 2015, pp. 39–46.

[84] V. Peysakhovich, C. Hurter, and A. Telea. "Attribute-Driven Edge Bundling for General Graphs with Applications in Trail Analysis". In: *2015 IEEE Pacific Visualization Symposium (PacificVis)*. 2015, pp. 39–46.

[85] Doantam Phan et al. "Flow Map Layout". In: *IEEE Information Visualization (InfoVis)*. 2005, pp. 219–224.

[86] Dave Plonka. "Flowscan: A network traffic flow reporting and visualization tool". In: *In USENIX LISA*. 2000, pp. 305–317.

[87] Dave Plonka. "FlowScan: A Network Traffic Flow Reporting and Visualization Tool". In: *Proceedings of the 14th USENIX Conference on System Administration*. USENIX Association, 2000, pp. 305–318.

[88] Tobias Post et al. "A High-Dimensional Data Quality Metric using Pareto Optimality". In: *EG/VGTC Conference on Visualization (EuroVis) - Posters*. doi: 10.2312/eurp.20171187. 2017.

[89] Tobias Post et al. "Ensemble Visualization of Bottlenecks in Planar Flow Networks". In: *Physical Modeling for Virtual Manufacturing Systems and Processes*. Vol. 869. Applied Mechanics and Materials. Trans Tech Publications, 2017, pp. 234–243.

[90] Tobias Post et al. “Fast 3D Thinning of Medical Image Data based on Local Neighborhood Lookups”. In: *EG/VGTC Conference on Visualization (EuroVis) - Short Papers*. doi: 10.2312/eurovisshort.20161159. 2016.

[91] Tobias Post et al. “OpenThinning: Fast 3D Thinning based on Local Neighborhood Lookups”. In: *IEEE Visualization Conference (Vis) - Workshop on Visualization in Practice*. 2016.

[92] Tobias Post et al. “User-Guided Visual Analysis of Cyber-Physical Production Systems”. In: *ASME Journal of Computing and Information Science in Engineering (JCISE)* 17.2 (2017). doi: 10.1115/1.4034872, p. 021005.

[93] Tobias Post et al. “Visual Analytics of Cascaded Bottlenecks in Planar Flow Networks”. In: *Leipzig Symposium on Visualization in Applications (LEVIA)*. urn: nbn:de:bsz:15-qucosa2-328032. 2018.

[94] Tobias Post et al. “Visually Guided Flow Tracking in Software-Defined Networking”. In: *IEEE Symposium on Visualization for Cyber Security (VizSec)*. doi: 10.1109/VIZSEC.2016.7739586. 2016.

[95] Kristin Potter et al. “Ensemble-Vis: A Framework for the Statistical Visualization of Ensemble Data”. In: *IEEE Workshop on Knowledge Discovery from Climate Data: Prediction, Extremes.* 2009, pp. 233–240.

[96] Bernhard Preim and Charl P. Botha. *Visual Computing for Medicine: Theory, Algorithms, and Applications*. 2nd ed. Morgan Kaufmann Publishers Inc., 2013.

[97] F. K. H. Quek and C. Kirbas. “Vessel extraction in medical images by wave-propagation and traceback”. In: *IEEE Transactions on Medical Imaging* 20.2 (2001), pp. 117–131. ISSN: 0278-0062. DOI: 10.1109/42.913178.

[98] Khairi Reda et al. “Visualizing the Evolution of Community Structures in Dynamic Social Networks”. In: *EuroVis'11*. 2011, pp. 1061–1070.

[99] G. Reinhart et al. “Cycle-Oriented Manufacturing Technology Chain Planning”. In: *Proceedings of the 3rd International Conference on Changeable, Agile, Reconfigurable and Virtual Production (CARV)*. 2009.

[100] Richard K Brail Richard E Klosterman, ed. *Planning Support Systems: Integrating Geographic Information Systems, Models, and Visualization Tools*. 1st ed. Esri Press, 2001.

[101] Henri Ruotsalainen, Elina Madetoja, and Jari Hämäläinen. "Navigation on a Pareto-optimal front utilizing gradient in formation in interactive multiobjective optimization". In: *International Conference on Engineering Optimization* (2008).

[102] M. Sacco et al. "Virtual Factory Manager". In: *Virtual and mixed reality*. Ed. by Randall Shumaker. Vol. 6773-6774. LNCS sublibrary. SL 3, Information systems and applications, incl. internet/web, and HCI. Heidelberg and New York: Springer, 2011, pp. 397–406. ISBN: 978-3-642-22023-4.

[103] Khalid Saeed et al. "K3M: A universal algorithm for image skeletonization and a review of thinning techniques." In: *Applied Mathematics and Computer Science* 20.2 (2010), pp. 317–335.

[104] R. Scheepens et al. "Visualization, Selection, and Analysis of Traffic Flows". In: *IEEE Transactions on Visualization and Computer Graphics* 22.1 (2016), pp. 379–388. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2467112.

[105] David Selassie, Brandon Heller, and Jeffrey Heer. "Divided Edge Bundling for Directional Network Data". In: *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)* (2011).

[106] Ulrich Sendler, ed. *Industrie 4.0: Beherrschung der industriellen Komplexität mit SysLM*. Xpert.press. Berlin Heidelberg: Springer, 2013. ISBN: 978-3-642-36916-2. DOI: 10.1007/978-3-642-36917-9.

[107] Daniel J. Simons and Daniel T. Levin. "Change blindness". In: *Trends in Cognitive Sciences* 1.7 (1997), pp. 261 –267.

[108] Edward Stringham. "Kaldor-Hicks Efficiency and the Problem of Central Planning". In: *Quarterly Journal of Austrian Economics* 4.2 (2001), pp. 41–50.

[109] Robert Tarjan. "Depth first search and linear graph algorithms". In: *SIAM JOURNAL ON COMPUTING* 1.2 (1972).

[110] Teryl Taylor et al. “FloVis: Flow Visualization System”. In: *CATCH*. 2009, pp. 186–198.

[111] The World Bank. *TradeMap Visualizer*. 2011. URL: http://devdata.worldbank.org/TradeMapVisualizer/DataVisualizer.html.

[112] Uncharted. *TorFlow*. “https://torflow.uncharted.software/". 2016.

[113] Corinna Vehlow, Fabian Beck, and Daniel Weiskopf. “Visualizing Group Structures in Graphs: A Survey”. In: *Computer Graphics Forum* (2016), n/a–n/a. ISSN: 1467-8659. DOI: 10.1111/cgf.12872.

[114] *VolVis*. “http://volvis.org". Accessed: 2016-03-09.

[115] Katharina Witowski, Martin Liebscher, and Tushar Goel. “Desicion Making in Multi-Objective Optimization for Industrial Applications - Data Mining and Visualization of Pareto Data”. In: *European LS-DYNA Conference* (2009).

[116] Peter Y. Wu and Sushil Acharya. “Visualizing Capacity and Load: A Production Planning Information System for Metal Ingot Casting”. In: *CONISAR Proceedings* 4.1823 (2011).

[117] Ling Xiao, John Gerth, and Pat Hanrahan. “Enhancing Visual Analysis of Network Traffic Using a Knowledge Representation.” In: *IEEE VAST*. IEEE Computer Society, 2006, pp. 107–114.

[118] H Yu et al. “The Importance of Bottlenecks in Protein Networks: Correlation with Gene Essentiality and Expression Dynamics”. In: *PLoS Computational Biology* 3.4 (2007).

[119] Ping Zhang. “Visualizing production planning data”. In: *Computer Graphics and Applications, IEEE*. Vol. 16. 1996, pp. 7–10.

[120] D. Zühlke. “SmartFactory - A Vision becomes Reality”. In: *13th IFAC Symposium on Information Control Problems in Manufacturing (INCOM 09)*. Ed. by Natalia Bakhtadze and Alexandre Dolgui. Vol. 13. V.A. Trapeznikov Institute of Control Sciences, Russia: International Federation of Automatic Control, 2009. ISBN: 978-3-902661-43-2.