Model Order and Cycle Breaking in SCCharts

Max Philipp Wilhelm Riepe

Bachelor's Thesis
March 2022

Prof. Dr. Reinhard von Hanxleden
Real-Time and Embedded Systems
Department of Computer Science
Kiel University

Advised by
Sören Domrös

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Kiel,

Abstract

Graphical abstraction comes with a lot of benefits, such as being easy to read and understand with minimalistic knowledge required. Creating an aesthetically pleasing layout for a given textual model is often achieved with an automatic layout algorithm.
The Sugiyama algorithm, also known as the layered or the hierarchical algorithm, is known to produce crossing minimal drawings. It is separated into five phases. This thesis will focus on comparing different strategies for the first stage, known as cycle removal or cycle breaking. While there are certain aesthetic criteria such as edge crossings or drawings size, which are well-defined scales to measure the quality of the layout, the results presented show that these sometimes are outweighed by other criteria.

Acknowledgements

I would like to express my gratitude to Sören Domrös for advising this thesis and offering constant feedback, ideas, and support. I am especially grateful for the swiftness of communication. Furthermore, I would like to thank Prof. Dr. Reinhard von Hanxleden for the valuable feedback and opportunity to write my thesis. The environment in the Real-Time and Embedded Systems Groups was extremely supportive. The daily virtual tea meetings helped me to find a daily routine in the times of exclusively working from home.

Finally, I would like to extend my gratitude to my family and friends, who not only provided encouragement, but also took the time to complete the survey used in this thesis.

Contents

List of Figures

List of Tables

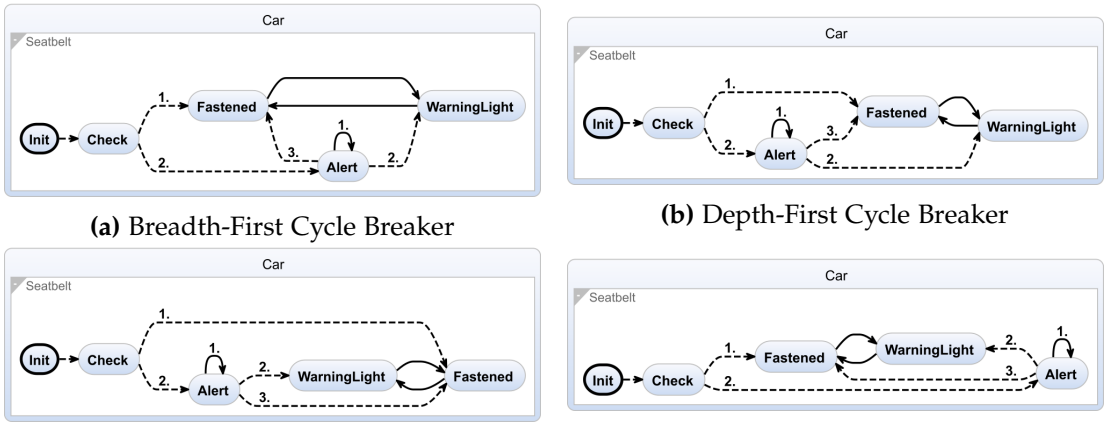
Abbreviations

SCChart(s)	Sequentially Constructive State Charts
KIELER	Kiel Integrated Environment for Layout Eclipse Rich Client
KLay	KIELER Layout Algorithms
CB	Cycle Breaker
ELK	Eclipse Layout Kernel
GrAna	Graph Analyses
BF-CB	Breadth-First Cycle Breaker
DF-CB	Depth-First Cycle Breaker
G-CB	Greedy Cycle Breaker
MO-CB	Model Order Cycle Breaker

Chapter 1

Introduction

Graphical abstraction of textual data is ubiquitous, be it node-based editors instead of textual scripting languages, drag and drop website editors, or Unified Modeling Language (UML) diagrams describing simple or complex processes in a specification book. The objective is to create an abstraction, which is easier to read or gives a better comprehension of a problem. Some visual abstraction languages, like the UML, come with countless rules and guidelines to reduce uncertainties in the graph. However, there are rules that apply to most types of visual representations. Edge crossings, for example, can lead to critical misunderstandings. For example, in a circuit diagram edge crossings if misunderstood can lead to shortcuts.
Manually constructing a layout with minimal edge crossings is time-consuming [Pet95]. This is becoming increasingly problematic as the number of abstractions and, in some situations, the size of these graphs increases. This is particularly problematic for graphical abstractions that are frequently modified, demanding a new layout. The use of automatic layout algorithms removes this tedious task and they have, therefore, become a widely adapted approach for layout creation.
One approach to creating layouts for statecharts is known as the *layered* algorithm, which is explained in Section 2.3. This algorithm is divided into five phases, the first of them is called *cycle breaking*. This thesis proposes different strategies for cycle breaking, and evaluates the differences. The different strategies are the *Breadth-First Cycle Breaker*, *Depth-First Cycle Breaker*, *Greedy Cycle Breaker* and *Model Order Cycle Breaker*. Different strategies lead to different sets of layers, as seen in Figure 1.1



(a) Breadth-First Cycle Breaker

(b) Depth-First Cycle Breaker

(c) Greedy Cycle Breaker

(d) Model Order Cycle Breaker

Figure 1.1. Graph used for the First Set of Questions in the Survey.

The graphs used for the evaluation are state charts, specifically SCCharts [HDM+14a]. These graphs are automatically synthesized from a textual input model, as shown in Figure 2.4. This text file might

hold additional information, as the model creator might have a mental model of the layout while creating this file. Therefore, one research question is if the model creator has a mental model of the diagram while creating the textual model and if a specific strategy represents this mental model better than another alternative. Additionally, the question arose whether there is a cycle-breaking strategy that performs consistently "*better*" than the remaining strategies. Better may be related to any aesthetic criterion mentioned in Section 5.1 or any combination of these.
Two evaluation methods were applied in this work.

- Quantitative Analysis: Evaluation of the aforementioned strategies, with clearly defined metrics. This might expose inherited features for certain strategies. This analysis is presented in Chapter 5.
- Qualitative Analysis: For the qualitative analysis, a survey was conducted. The research question was to expose if certain strategies achieve higher overall acceptance in the test group, despite having some quantitative flaws. Results are presented in Chapter 6.

While the quantitative analysis gives a strong base on which cycle-breaking strategy performs best, the aforementioned empirical survey evaluates if the metrics used for this analysis have an impact on the perception. This survey was designed to see if some of the quantitative criteria may be neglected in favor of a different criterion.

1.1 Related Work

The layered algorithm used here is based on the approach proposed by Sugiyama et. al. [STT81]. This algorithm is also known as the hierarchical or the Sugiyama approach.
There are several aesthetic criteria for determining how well a layout algorithm performs. These criteria are based on well defined metrics [Pur02]. One aesthetic criterion found to have a big impact on clarity is the number of edge crossings [Pur97].
The underlying phase of the cycle-breaking step is also known as the feedback arc set problem [ELS93a].
Another approach to improve the connection of textual model and the synthesized graph is explained in "Preserving Order during Crossing Minimization in Sugiyama Layouts" [DH21].

1.2 Outline

This thesis is structured as follows. Chapter 2 introduces fundamental knowledge, such as terminology and the automatic layout algorithm, and the framework used in this thesis. In Chapter 3, the cycle breaking phase of the algorithm is explained in detail. The aforementioned cycle breaking strategies are introduced in this chapter as well. Chapter 4 shows the implementations produced for this thesis. Chapter 5 presents the quantitative evaluation of the different strategies. This evaluation uses automated graph analysis. In Chapter 6 the results of the survey are evaluated. In Chapter 5, provides a conclusion, some points of criticism and points towards future work.

Chapter 2

Preliminaries

The focus of this chapter is to explain the basic methodologies, principles and frameworks used in this thesis.
Implementations mentioned in Chapter 4 are integrated into the Kiel Integrated Environment for Layout Eclipse Rich Client (KIELER)[1] environment [FH10; HFS11]. Consequently, there will be a quick overview of KIELER. After that, the graph terminology used in the following chapters and sections will be given. The layered algorithm will be introduced. Finally, a quick introduction to SCCharts will be given, as the graphs used in the evaluation in Chapters 5 and 6 are SCCharts.

2.1 KIELER

KIELER is a research project run by the Kiel University's Real-Time and Embedded Systems group. The goal of this project is to improve model-based complex system design. The layered algorithm used for the synthesis of SCCharts is modified for this thesis. While KIELER supports a wider range of graphical languages than SCCharts, this thesis will concentrate on SCCharts.
KIELER allows for on-the-fly modifications to the layout creation. The following sections introduce two powerful assets of KIELER used for the analysis.

2.1.1 KiCodia

KiCodia is a full-featured command-line compiler and is part of the KIELER Compiler and Command-Line Interfaces. It may be used to render graphs in various formats. These images are used in the survey evaluated in Chapter 6. Additionally, with some modifications to it, it is able to export graphical models in the *elkg* format, which is the required format for the graph analysis tool GrAna. The *elkg* format is one of the formats for a *Elkgraph*, the others are *elkt* and *json*. These are convertible using a conversion tool[2].

2.1.2 GrAna

The integrated graph analysis tool GrAna is one of KIELER's most valuable assets for this thesis. This allows the batch analysis of graphs. An in-depth explanation of GrAna may be found in the bachelor's thesis by Martin Rieß [Rie10]. The analysis method receives the node containing the graph, as seen in Listing 2.1. Two additional analysis criteria will be integrated into KIELER, these are covered in Section 4.2 and 4.3. GrAna creates easy-to-use files in the *CSV* format, with the analysis criteria as columns and a row for each graph analyzed. These are then evaluated using R in Chapter 5.

[1] `www.rtsys.informatik.uni-kiel.de/en/research/kieler`
[2] `https://rtsys.informatik.uni-kiel.de/elklive/conversion.html`

```java
@Override
public Object doAnalysis(final ElkNode
     parentNode, final AnalysisContext context,
    final IElkProgressMonitor progressMonitor)
        {
    [...] // Analysis
    return resultObject;
}
```

Listing (2.1) Grana Analysis Method

```
globalResources
 logs "input/path" filter "filename" recurse

globalOutputs
 csv  "output.csv"
execute all

job jobName
 resources
  ref logs
 analyses
  de.cau.cs.kieler.grana.edgeCrossings
 output ref csv
```

Listing (2.2) Example .grana File

Figure 2.1. GrAna Analysis Structure and Example .grana File

2.2 Graph Definitions and Terminology

The majority of the definitions that follow are basic graph theory terminology. They are meant to provide the foundational knowledge used in the following sections.
The following definitions introduce graphs and graph subtypes. Each definition in this chain, in general, refines the previous graph and introduces new rules or restrictions. Some refinements require the introduction of other terms. Therefore these terms are defined in this chain as well.

2.2.1 Definition (Graph)**.** A *graph* G is a pair (V, E), where V is a finite set of nodes and a set of edges $E \subseteq \{(u, v) | u, v \in V\}$ connecting the nodes.

2.2.2 Definition (Directed edges)**.** Let $u, v \in V$ and $e \in E$ with $e = (u, v)$. For *directed* edges the pair (u, v) is ordered and indicates the direction of the edge, starting in u and ending in v. Conversely an *undirected* edge does not have a direction, the pair has no ordering and $(u, v) = (v, u)$.

2.2.3 Definition (Directed graph)**.** A graph $G = (V, E)$ is a called *directed* graph, if all edges $e \in E$ are directed edges. A directed graph is also called *digraph*.

2.2.4 Definition (Path)**.** Let $G = (V, E)$ be a graph. A *path* of length n is defined as a tuple $(v_1, ..., v_n)$ with $v_1, ..., v_n \in V$ and it holds that $\forall i, 0 < i < n : (v_i, v_i + 1) \in E$.

2.2.5 Definition (Cycle)**.** A *cycle* is a path for which $v_1 = v_n$ holds.

2.2.6 Definition (Acyclic graph)**.** A directed graph $G = (V, E)$ is called *acyclic* if the graph does not contain any cycles.

2.2.7 Definition (Layered graph)**.** A *layered* graph is defined by the triple $G = (V, E, L)$, where V, E are defined as in 2.2.1 and L is defined as a finite ordered set, which partitions V into non-empty *layers*. For all $v \in V$ holds that v is assigned to exactly one layer.

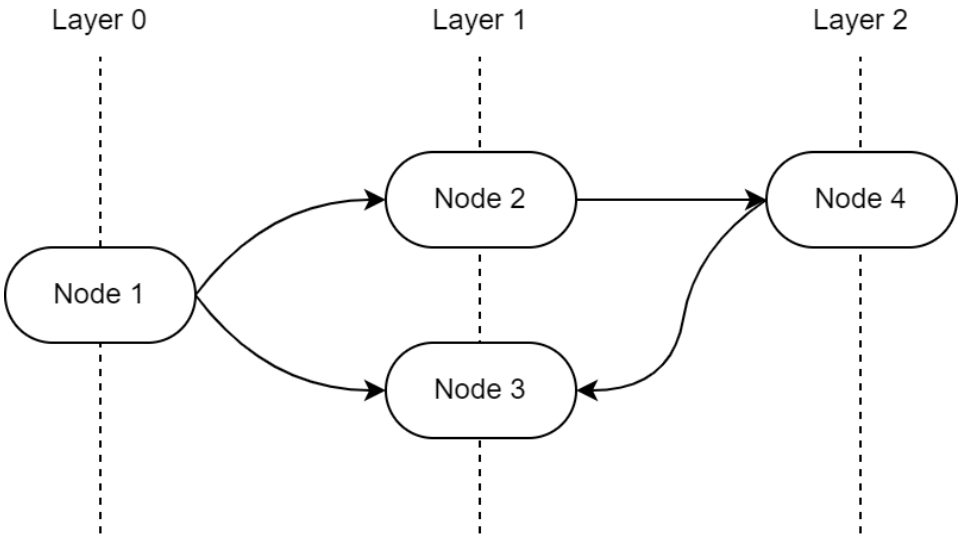


Figure 2.2. Simple Layered Graph This simple example shows a layered graph with 3 layers and 4 nodes.

Some general terms, which are not required for the previous chain, are required in the later sections and chapters.

2.2.8 Definition (Out-degree)**.** The *out-degree* of a node v is the number of edges that start in v.

The out-degree of "Node 1" in Figure 2.2 is 2.

2.2.9 Definition (In-degree)**.** The *in-degree* of a node v is the number of edges that end in v.

The in-degree of "Node 3" in Figure 2.2 is 2.

2.2.10 Definition (Source)**.** A node is called *source*, if its in-degree is 0.

"Node 1" of Figure 2.2 is a source.

2.2.11 Definition (Sink)**.** A node is called *sink*, if its out-degree is 0.

"Node 3" of Figure 2.2 is a sink.

2.2.12 Definition (Hierarchical node)**.** A node is called *hierarchical*, if it contains child nodes. In the graphical representation, this is displayed by drawing child nodes inside of the hierarchical node, which in this case, we call parent node.

2.2.13 Definition (Hierarchical depth)**.** Let $v \in V$ be a hierarchical node in the graph $G = (V, E)$. Let d be the hierarchical depth of v. The children of v have the *hierarchical depth* of $d + 1$. The lowest depth is 0.

Graphs for the KLay layered algorithm are essentially hierarchically nested nodes.

2.2.14 Definition (Root node)**.** The *root* node is a hierarchical node that contains every other node of the graph.

As mentioned in the introduction, the underlying problem of the cycle breaking is known as the feedback set problem:

2.2.15 Definition (Feedback set)**.** Let F be a subset of nodes of a given cyclic graph $G = (V, E)$. F is called *feedback set* if reversing all edges contained in F results in G being acyclic.

2.3 Layered Algorithm

The layered algorithm has been improved to allow for a wider range of graphs to be accepted, e.g. cyclic graphs [SFH09; Sch11]. It has five different phases in the following order: *Cycle Breaking*, *Layer Assignment*, *Crossing Minimization*, *Node Placement*, and *Edge Routing*.
As previously stated, the layered algorithm uses a very modular approach. The modularity of the KLay layered algorithm is improved further, allowing for easy code replacement or the addition of *Intermediate Layout Processors* [SSH14] before, between, or after any phase. For this thesis a new cycle breaking strategy has been implemented, covered in Section 4.1.
The following sections provide a quick overview of the algorithm's phases. The layout direction used here is *RIGHT*, meaning that the layers are created from left to right.

2.3.1 Cycle Breaking

The *Cycle Breaking* phase, as previously stated, is added to allow for a wider range of input graphs and is not part of the Sugiyama approach. This phase is introduced to enable cyclic graphs for the layered approach, by removing cycles. As this phase is the main part for this thesis, it is explained in more detail in Chapter 3.

2.3.2 Layer Assignment

The input to the *Layer Assignment* phase is an acyclic digraph, and the output is a layered digraph. The *Longest Path Layering* method is a simple way to do this [SFH09].

```
1 Find length of longest path lp
2 Assign all sinks to layer Layer L_lp
3 While there are vertices without layer assignment:
4   Find a vertex whose successors have been assigned to layers (l_j,...,l_k)
5   Assign this vertex to layer min(l_i)-1
```

Listing 2.3. Longest Path Pseudo Code

There are other approaches implemented for layering in the KLay layered algorithm, for example, the *Network Simplex Layering* [Döh10] or a layer assignment approach as described by Coffman and Graham [CG72].

2.3.3 Crossing Minimization

This phase focuses on minimizing edge crossings in a layered graph. To achieve this, the node order in each layer is altered using a barycenter approach. The algorithm for this phase is the *Layer Sweep Crossing Minimizer*. Simplified, this works by sweeping the layers from both directions and adjusting the node order, in the so-called free layer, if this results in fewer crossings. The ordering of the free layer is determined by the barycenter of the connected nodes of the previous layer.

(a) Graph with avoidable crossings **(b)** Unnecessary crossing resolved

Figure 2.3. Crossing Minimization Visualization In this example the second layer is the free layer and the sweep is from left to right. If the sweep would be from right to left, the first layer ordering would be adjusted.

2.3.4 Node Placement

This phase decides the final y-position of the node. The final position is given by two coordinate values. For the x-position the Layer Assignment decided the layer, essentially the group of nodes with the same x-position. Later, these layers are moved to create enough space for the edges to be drawn, determining the final x-position. The y-position is constrained by the node order decided in the previous step. There are two aesthetic goals for this phase. The first goal is to create compact drawings, not using more space than necessary. The other goal is to reduce edge bends. An in-depth explanation may be found in the master's thesis by John Carstens [Car12] or in the work "Fast and Simple Horizontal Coordinate Assignment" by Brandes et. al. [BK02].

2.3.5 Edge Routing

After the nodes are placed, the edges are drawn. There are different approaches for this phase, with varying complexity and goals. The simplest way to route edges is the *Polyline Edge Router*, creating sections of straight lines to create the edge.

2.3.6 Intermediate Processors

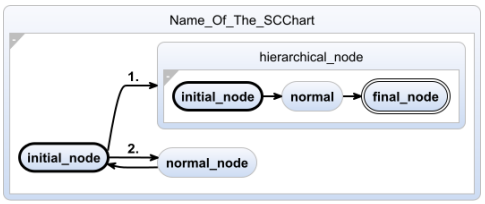
One of the main advantages of the KLay layered algorithm is the extreme modularity of the phases. The phases only perform the core actions for their phase. Additional steps like the edge reversal, used to reverse the edges to their correct direction, are applied in-between phases. The phases may indicate that they depend on certain intermediate processors to be run before or after the phase.

2.4 Sequentially Constructive State Charts

This section covers the basics required to work with Sequentially Constructive State Charts (SCCharts). SCCharts is a visual language [HDM+14b] that uses state chart notation. An example of the textual representation and the synthesized graph are shown in Figure 2.4. The labels used here represent nodes of this graph. Nodes might have child-nodes, which in the visual representation are drawn inside of the parent node, also known as hierarchical nodes, seen in the "hierarchical_node". The layout algorithm is applied to each hierarchical depth independently. If a certain hierarchical depth contains edges this depth requires exactly one initial node, marked with a bold outline, the two nodes named "initial_node". Edges are ordered by the textual representation of the graph, if the appropriate setting

```
1  scchart Name_Of_The_SCChart {
2   bool condition, condition2
3
4   initial state initial_node
5   if condition go to hierarchical_node
6   if condition2 go to normal_node
7
8   state hierarchical_node {
9    initial state initial_node
10   if condition go to normal
11
12   state normal
13   if condition go to final_node
14
15   final state final_node
16  }
17
18  state normal_node
19  if condition go to initial_node
20 }
```

(a) Textual Representation



(b) Graphical Representation

Figure 2.4. SCChart Textual and Graphical Representation The textual representation of the SCChart is in the *sctx* format. The keyword initial marks the node as an initial node for this layer. If a state has a block with curly braces, this node is hierarchical and the states declared inside of this block are its children, as seen in the graphical representation. Finally, connections between nodes are created with if-conditions and the go to keyword.

is used in KIELER. This not only increases the connection of the textual and visual representation, also known as *secondary notation* [Pet95], the edge conditions are processed in the order given by the textual representation. This inherited ordering of the graph is something worth preserving because it increases the link between the textual and graphical representation. SCCharts have a lot more to offer, but for this thesis, the information provided suffices.

Chapter 3

Cycle Breaking

In this chapter, the cycle-breaking phase is explained. Furthermore, there will be an introduction to four different cycle-breaking strategies.
A reason to adjust the cycle breaking is that this step has a major influence on the layers that the next step will form. This phase requires a digraph as input and returns an acyclic digraph. Although removing cycles from a digraph does not appear to be difficult, finding minimal feedback sets has been proven to be NP-complete [CTY07; GJ79]. As a result, the cycle-breaking strategies explained in the following sections work with a heuristic approach. They do not seek a minimal feedback set; instead, they reverse selected edges until there are no cycles left. These edges are remembered and then reversed back to their original state after the layer assignment. Figure 3.1 shows a simple cycle-breaking example.

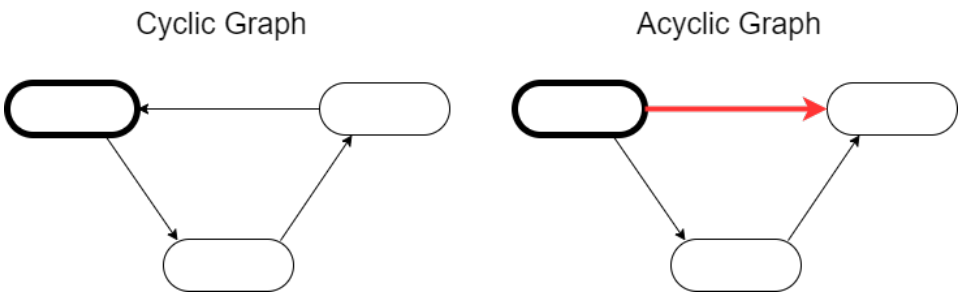


Figure 3.1. Cycle Breaking Example The edge marked red is reversed to break the cycle. For this simplistic example, any edge of the three might be reversed to break the cycle.

The different cycle-breaking strategies form feedback sets containing different edges and therefore the nodes are assigned to different layers. One of the cycle-breaking strategies, the *Model Order Cycle Breaker* (MO-CB), focuses on increasing the connection between textual and graphical representation [DH21]. This is a big difference between this strategy and the remaining three strategies. For the remaining strategies, the altering of the textual model for a graph has minimal, and seemingly arbitrary, influence on the final layout result. This leaves few options for a graph creator to influence a layout.

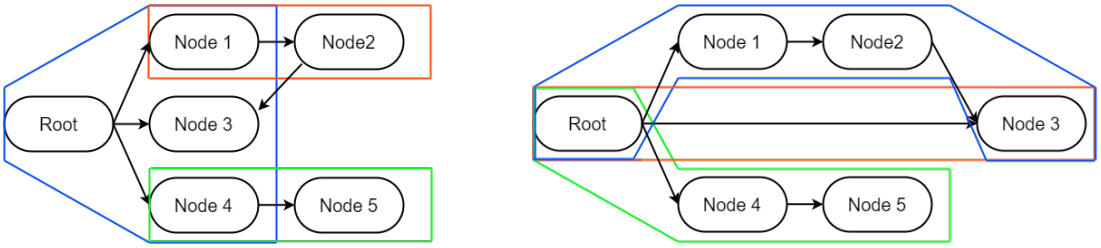
3.1 Breadth-First Cycle Breaker

The *Breadth-First Cycle Breaker* (BF-CB) was implemented as a part of this thesis. The implementation is shown in Section 4.1. As the name suggests, this strategy uses the breadth-first search approach for graph traversal. The heuristic approach utilizes that the breadth-first search remembers previously visited nodes. Edges are reversed if the end node is a node that was previously visited. As a result, this strategy has the same approximate runtime as the breadth-first search, which is $O(V + E)$. If this

strategy and the MO-CB result in the same layout, it would imply that the mental model of the creator was in the style of the breadth-first search, meaning that the mental model is derived from a node with a layer of subsequent nodes, as represented with the colored boxes in Figure 3.2a.

3.2 Depth-First Cycle Breaker

The Depth-First Cycle Breaker (DF-CB), like the Breadth-First Cycle Breaker, is based on the eponymous depth-first search algorithm. Again while checking the outgoing edges of the current node, if the ending node reached by such an edge was already visited, this edge is reversed. Like the BF-CB this also has the runtime approximation of the underlying concept with $O(V + E)$. Likewise, if this strategy and the MO-CB result in the same layout, it would imply that the textual ordering represents a depth-first approach, meaning the mental model might by constructed of nodes with multiple branches rooting in it. This would reflect the graph traversal of the depth-first search, as represented with the colored boxes in Figure 3.2b.



(a) Breadth-First Mental Map **(b) Depth-First Mental Map**

Figure 3.2. Mental Map of a Graph, Breadth-First and Depth-First (a) represents the breadth-first visualization, with groups of layers. **(b)** shows the depth-first visualization, having a root with branches. The blue box is the first group, the red box the second group and the green box the last group. If these two graphs were SCCharts, they would use the same textual input model, but use different cycle breaking strategies. The textual input for this graph starts with the Root node and then the remaining states in alphanumeric ordering.

3.3 Greedy Cycle Breaker

The Greedy Cycle Breaker (G-CB) is implemented in KIELER using the approach described in "A fast and effective heuristic for the feedback arc set problem." [ELS93b], which has a runtime approximation of $O(E)$. The main objective of this strategy is to reduce backward edges. This algorithm uses a metric called *outflow*, which is defined as follows.

3.3.1 Definition (Outflow)**.** Let d_{in} be the in-degree of a node and let d_{out} be the out-degree. The *outflow* is defined as $d_{out} - d_{in}$.

This metric is used to determine the most sensible incoming edges to reverse. For example, reversing the outgoing edges of a source or the incoming edges of a sink should never be considered, as these nodes will never induce a cycle. This approach creates a layering for the graph, where each layer contains exactly one node. These layers are ordered by the nodes *outflow*, ordering the highest outflow to the left and the lowest outflow to the right. Sources are put in the starting layers regardless of their outflow. Sinks are put in the last layers regardless of their outflow. Edges that point to the left are now reversed, therefore, breaking all cycles.

3.4 Model Order Cycle Breaker

The Model Order Cycle Breaker (MO-CB) is designed to increase the correlation between textual and graphical representation, as mentioned this is also known as secondary notation.
This cycle breaker works as follows. Each node is assigned an integer value (the *model order*) according to the placement in the textual representation. If an edge starts in a node with a higher model order than the node it ends in, it is reversed. However, if the designer chooses, there is the possibility to set layer constraints for nodes. These overrule the model order created from the textual ordering. There are five layer-constraint options to achieve the desired ordering, however, it suffices here to distinguish between three: *FIRST* < *NONE* < *LAST*. Therefore, nodes with the constraint *FIRST* are to be placed in the first possible layer. However, let us assume the nodes v, w have the layer constraint *FIRST* and there is an edge $e = (v, w)$. The node with the lower model order is now placed in a prior layer, and if this node is w, e is reversed. This algorithm has a runtime approximation of $O(V + E)$.
The Model Order Cycle Breaker, even without the constraints, is able to produce any desired and possible set of edge reversals. This includes the sets that the other strategies produce. The layer a node is assigned to is directly influenced by the edges that are reversed during cycle breaking, as described in Section 2.3. Therefore, a construction method is given, with which it is possible to create any possible valid edge reversal set by using the MO-CB and altering of the textual ordering. A valid set of edges to reverse mainly is constraint by edges that cannot be reversed independently. Let a, b be nodes in a graph and there are multiple edges that start in a and end in b. If any of these edges is reversed, all of these edges have to be reversed.

Proof: Create any valid set of edges to reverse using the Model Order Cycle Breaker.
Hypothesis: Any valid set of edges to reverse may be created using the MO-CB.
Proof:
Let $G = (V, E, L)$ be a layered graph. The order within a layer is neglected for this proof, as this is decided by the crossing minimization phase. To simplify matters all edges in E that are backward edges are reversed.
Constructing the textual order:
Recall L is a ordered set containing (ordered) sets of nodes. Let $|L| = n$ be the count of layers and $l_0 ... l_n, l_i \in L$ be the layers.
Let M be an empty list of nodes.
Append all nodes, contained in layers $l_i : 0 <= i <= n$, to M. The layers are appended according to their order in L. Now that all nodes are in M, it may be used to create the textual order. Order the textual representation according to the nodes position in M.
Let $m_j \in M$ be the node at position j. Using the MO-CB, all edges e for which the following applies are reversed: $m_a, m_b \in M : e = (m_a, m_b) \wedge a > b$. This results in a graph $G' = (V', E')$
It is to be shown that the edges in E' have the same direction as the edges in E.
Both G and G' only have forward edges, therefore all edges in E start in a layer with a smaller index and end in a layer with a bigger index. All edges in E' start in a node that has a smaller index in M than the ending node. As the order of M is given by the layers of G, it is certain that a node in a previous layer compared to another node in a later layer also has a smaller index in M. □

A visualization of this construction is shown in Figure 3.3.

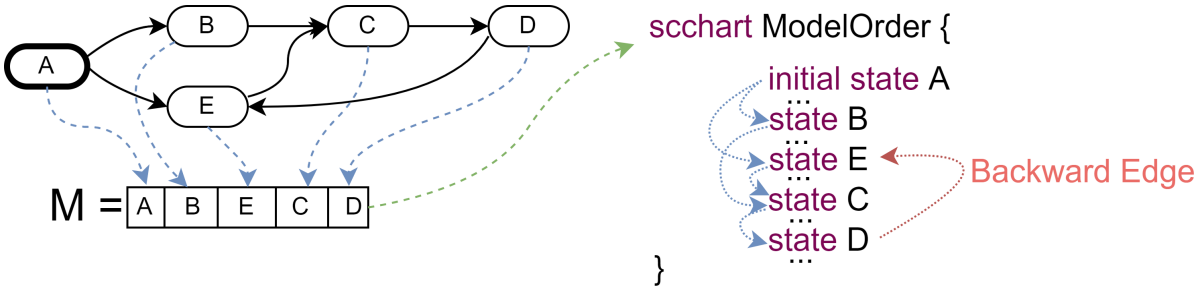


Figure 3.3. Visualization of Model Order Construction Extracting a Model Order that will result in the same Backward Edges using the Model Order Cycle Breaker. The dashed blue lines show the extraction of the model order into the list M. The dashed green line shows the link of position in M and the ordering in the textual definition. The dotted lines represent the edge directions that the Model Order Cycle Breaker would create.

Chapter 4

Implementation

This chapter is separated into two sections. The first section will present the implementation of the *Breadth-First Cycle Breaker* (BF-CB).
Certain types of graphical layouts inherit certain features, which might result in additional aesthetic criteria. Two of those features applicable for layered graphs are presented in the second section. These criteria are part of the quantitative evaluation in Chapter 5.

4.1 Breadth-First Cycle Breaker

As mentioned previously, the BF-CB utilizes the graph traversal of the breadth-first search. This is shown in Listing 4.1.

```
1  private void bfs(final LNode n) {
2     if (visited[n.id]) {
3        return;
4     }
5     visited[n.id] = true;
6
7     for (LEdge out : n.getOutgoingEdges()) {
8        if (out.isSelfLoop()) {
9           continue;
10       }
11       LNode target = out.getTarget().getNode();
12       if (visited[target.id]) {
13          edgesToBeReversed.add(out);
14       } else {
15          bfsQueue.add(target);
16       }
17    }
```

Listing 4.1. Breadth-First Cycle Breaker The *bfs* method is run for each node in a given queue, the *bfsQueue*.

Starting the breadth-first cycle breaking from each source simultaneously or performing it sequentially produces varying results, as seen in Figure 4.2. However, SCCharts and most other graphs contain at most one source per hierarchical depth, the initial node for this graph. If there would be more than one source node or if the source node would not be the initial node, these nodes would not be reachable. Therefore, syntactically correct SCCharts are unaffected by this difference. The sequential approach was chosen here.

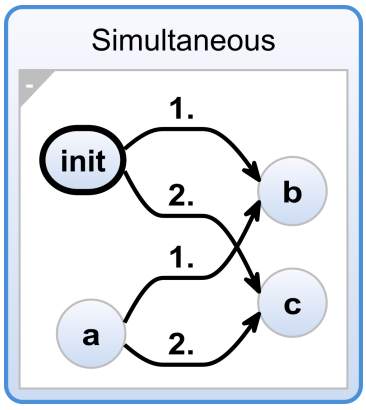
```
for (LNode source : sources) {
   bfs(source);
}
bfsLoop();
```

(a) Simultaneous Breadth-First Search.

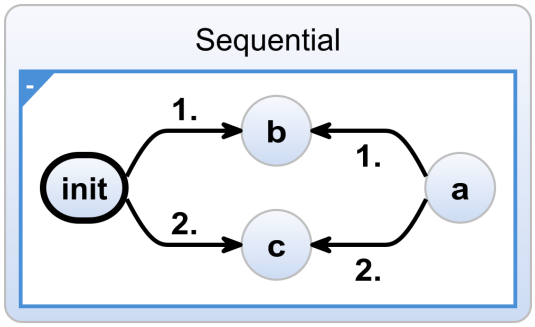
```
for (LNode source : sources) {
   bfsQueue.add(source);
   bfsLoop();
}
```

(b) Sequential Breadth-First Search.

Figure 4.1. Different Approaches to the Breadth-First Cycle Breaking
(a) shows the simultaneous approach, which starts by enqueuing all source nodes, before starting the breadth-first search. **(b)** shows the sequential approach, which runs the breadth-first search for each source sequentially.



(a) Simultaneous Breadth-First Cycle Breaking



(b) Sequential Breadth-First Cycle Breaking

Figure 4.2. Visual representation of Sequential and Simultaneous BF-CB
For this example a syntactically wrong graph is used to show the differences in **(a)** the simultaneous approach or **(b)** the sequential approach.

4.2 Backward Edge Analysis

The first additional aesthetic criterion is the number of backward edges. The layered algorithm has a layout direction, let the direction be right for this example. A backward edge in this example would be an edge where the start point would be to the right of the endpoint, meaning the starting node is in a layer with a higher id than the node where the edge ends.
The analysis, which is presented in Listing 4.2, computes the sum of the backward edges for each hierarchical depth, starting with the highest depth. There are some validity checks, as this analysis does not deal with some types of edges, e.g. hyperedges (edges with multiple starting or ending nodes).
This criterion is not only an aesthetic criterion, as it may also be used to determine the count of edge reversals in the cycle-breaking stage. Finding a significant and reliably better cycle breaker concerning edge reversals would indicate that this strategy might be a better heuristic for the feedback set problem.

```java
public int evalBackwardsEdges(final ElkNode node) {
   int backwardsEdges = 0;
   if (node.isHierarchical()) {
      for (ElkNode child : node.getChildren()) {
         int backwardsEdgesInChild = evalBackwardsEdges(child);
         switch (backwardsEdgesInChild) {
            [...] // Checks for validity
            default:
               backwardsEdges += backwardsEdgesInChild;
         }

      }
   }
   for (ElkEdge edge : node.getOutgoingEdges()) {
      [...] // Checks for validity
      ElkNode src = (ElkNode) edge.getSources().get(0);
      ElkNode target = (ElkNode) edge.getTargets().get(0);
      [...] // Check if LAYERING_LAYER_ID is set
      if (src.getProperty(LayeredOptions.LAYERING_LAYER_ID) > target.getProperty(LayeredOptions.
          LAYERING_LAYER_ID)) {
         backwardsEdges++;
      }
   }
   return backwardsEdges;
}
```

Listing 4.2. Backward Edge Analysis This analysis uses the layer id assigned to the nodes. For every edge in the graph it is evaluated if the layer id of the starting node is bigger than the layer id of the ending node. If this is the case, this edge is a backward edge.

4.3 Layer Count Analysis

Another criterion for layered graphs is the number of layers created for the graph layout. For this analysis, layers are counted for each hierarchical depth individually and summed up. For this, we find the highest *layer id* in each hierarchical node. This analysis is similar to the area analysis. A lower layer count means that the layers have more nodes in them. This in turn means that the graph has higher compactness. The implementation is shown in Listing 4.3.

```java
public int evalLayerHierachical(final ElkNode node) {
   int maxLayerID = 0;
   Queue<ElkNode> hierarchicalChildren = new LinkedList<ElkNode>();

   if (node.isHierarchical()) {
      for (ElkNode child : node.getChildren()) {
         if (child.isHierarchical()) {
            hierarchicalChildren.add(child);
         }
         if(maxLayerID < child.getProperty(LayeredOptions.LAYERING_LAYER_ID)) {
            maxLayerID = child.getProperty(LayeredOptions.LAYERING_LAYER_ID);
         }
      }
      while (!hierarchicalChildren.isEmpty()) {
         ElkNode child = hierarchicalChildren.poll();
         int sumDeeperLayers = evalLayerHeiracical(child);
         switch (sumDeeperLayers) {
            [...] // Validity Checks
            default:
               maxLayerID += sumDeeperLayers;
         }

      }
   }
   return maxLayerID;
}
```

Listing 4.3. Layer Count Analyses
The analysis starts by calling this method with the root of the graph as the argument. This method recursively sums the layercount for each hierachical depth.

Chapter 5

Quantitative Evaluation

In this chapter, the different aesthetic criteria evaluated will be introduced. Following that, there will be a brief introduction to the dataset. The evaluation methodology will be explained. Following that the results of the analysis will be provided, for which the R-project[1] was used. The quantitative evaluation results are presented, starting in Section 5.4. The first analysis presented is the *Node Count Analysis*, as this yields the same results for all cycle breaking strategies. This analysis shows the variety of the dataset. The remaining quantitative analysis will be presented in alphabetical order. At last, the strategies are compared to the MO-CB. A conclusion will be drawn in Chapter 7. Some noteworthy critics for this analysis will also be presented.

5.1 Aesthetic Criteria

The following table presents the analysis names, a short explanation, and the return values of the analysis. The results are presented in the later sections of this chapter.

Table 5.1. Aesthetic Criteria Names and Explanations

KIELER Name	Explanation
Area	This analysis evaluates the area consumption of a layout, which is the same as the size of the root node. Returns width, height and their product.
Backwards Edges	Counts edges that go against the layout direction. Indicates the edges reversed during cycle breaking, as described in Section 4.2.
Edge Crossings	Counts the edge crossings in a graph.
Edge Length	Measures the length of edges. Returns minimum, maximum and average edge length.
Layer Count	Counts the layers of the layout, as described in Section 4.3.
Node Count	Counts all nodes in the graph, including the root.

5.2 The Dataset

The dataset contains 419 SCCharts. Most of these are models created as part of lectures by the Real-Time and Embedded Systems group. These graphs contain as little as three nodes and up to around 300 Nodes. There are three outliers with 2300, 7000, and 29000 nodes. There are two things to know about graphs with really small or high node counts. Graphs with very few nodes are usually too small to result in different layout results based on cycle breaking strategy. Graphs with high node count are often combinations of different subgraphs. The differences in these graphs in most cases are only in small subgraphs. There are also graphs that do not contain any cycles and are, therefore,

[1] `https://www.r-project.org`

of no use for this analysis. Therefore, for the analysis presented in this chapter, graphs that achieve identical layouts for all cycle-breaking strategies are cut from the dataset. This leaves the dataset with 265 graphs, for which at least one strategy creates a different result. Of these 265 graphs, 47 are unique for each cycle-breaking strategy. These are the most interesting graphs for the survey evaluated in Chapter 6. The current default cycle breaking strategy is the *Greedy Cycle Breaker* and most SCChart models have been created using this strategy.

5.3 Evaluation Methodology

There are three evaluation methods used for this chapter. This section will briefly introduce them.

5.3.1 Normalization

As the dataset varies a lot in graph size, the number of edges, and node count it is sensible to *normalize* the results. The differences found in smaller graphs are otherwise outweighed and overshadowed by the differences in larger graphs. This is especially noticeable for a metric like the edge crossing count, as there are a lot of graphs that do not have any edge crossings. A graphical representation like a *box-plot* would only show an indistinguishable line close to the x-axis.
For each graph G up to four **different** layouts are created, using the different cycle breaking strategies. Let v_1, v_2, v_3, v_4 be the values returned by a given analysis for the different strategies. Normalizing v_i in this case would mean:

$$norm(v_i) = \frac{v_i}{mean(v_1, v_2, v_3, v_4)}$$

Most graphical representations presented in this section use this method. Raw data may be found in Appendix A.
A normalized representation no longer shows the direct values for a specific metric but rather how it compares to an average baseline of the four strategies, meaning the value *1* on the y-axis is the average for this metric taken from all strategies. A value greater than this means that this sample performed worse than the average and a value below means it performed better.
Some metrics might evaluate to zero, e.g. edge crossings. There are two conditions to differentiate here. If at least one cycle breaking strategy (A) resulted in a value >0, then 0 means that this strategy (B) was able to remove, e.g. edge crossings, while the other strategy required them. Mathematically this would mean that strategy B is infinitely better than the other strategy, which is accurate for a singular metric. The other situation comes with a problem. This problem arises if all strategies evaluate to zero, which is the case for the majority of edge crossing samples (59%). Just keeping the zero for all strategies would not be accurate, as this would mean all of them are performing infinitely better than their combined average. This is not accurate, as all of them perform just like the average, without e.g. edge crossings. There are two approaches to deal with this problem. Firstly, it would be possible to use the following approach for normalization:

$$norm(v_i) = \begin{cases} \frac{v_i}{mean(v_1, v_2, v_3, v_4)} & , mean(v_1, v_2, v_3, v_4) > 0 \\ 1 & , otherwise \end{cases}$$

Using this approach these cases would be counted as if they all perform like the average, which would be accurate. Consequently, the number of ones in the result would increase greatly, removing the ability to see how often strategies perform on average if they produce different results.
The second approach is filtering the dataset, removing the examples where all strategies evaluate

to zero. This feels less intrusive for the final result, as important data is not overshadowed by these, mostly uninteresting, cases. As this problem only exists for edge crossings, this analysis will be done on the smaller dataset, as detailed further in Section 5.7. As to not lose data, the number of cases where this problem arises is given in the analysis.

5.3.2 Kruskal-Wallis Test

In addition to the graphs presented for each analysis, the results are evaluated with the Kruskal-Wallis test [KW52]. The Kruskal-Wallis test by ranks is a non-parametric[2] method for testing whether samples originate from distibutions with the same median. It is used for comparing two or more independent samples of equal or different sample sizes. It extends the *Mann–Whitney U test* [3], which is used for comparing only two groups. It is used to compare the four datasets, to determine if there is a statistically significant difference in these samples, starting with the null hypothesis: There is no difference in the samples. If the *p-value* is smaller than 0.05, the test indicates there is a significant difference for this metric.

5.3.3 Willcoxon Test

If the Kruskal-Wallis test indicates significant differences in a metric, the pairwise *Willcoxon* signed rank test [Wil45] is used as a post hoc analysis test. The Willcoxon test may be used to analyze paired data, which is the case for this dataset. To account for multiple comparisons when comparing multiple sets of pairwise data, the *Bonferroni* correction [DWC+13] of the p-values was used. This means that the significance between a set of data is given if the p-value is smaller than the p-value of the Kruskal-Wallis test divided by the number of comparing tests (in this case six). The results of this test do not return a relation, which is better or worse, they simply indicate a difference.

5.4 Node Count Analysis

Figure 5.1 shows the node count analysis of the dataset as a violin plot. The plot's y-axis is limited to roughly 300 nodes. Therefore, five graphs are not included in Figure 5.1. These graphs have a node count of 29085, 6979, 2308 and 625. The median node count of the entire dataset is 17 and the mean is 182. Not including the outliers the mean drops to 46.27 and the median remains at 17 nodes. These outliers deform the statistical graphs to a point where information extraction would not be possible. Therefore, the statistical graph of this analysis does not contain these graphs. As the remaining graphs utilize normalization they are unaffected by this problem. The raw data is presented in the appendix.

5.5 Area Analysis

KIELER allows for easy adjustments regarding the spacing between graph components. An example setting is the node-node spacing which determines the minimum distance between two nodes. The default settings are used for this analysis. Figure 5.2 presents area consumption using normalized values. Area consumption is closely related to node count. Usually, the higher the node count the more space is required. Therefore, not normalizing would void the results of smaller graphs. 1.0 on the y-axis is the average space consumption of each layout for this graph. The data represented in

[2] `https://en.wikipedia.org/wiki/Nonparametric_statistics`

[3] `https://en.wikipedia.org/wiki/Mann-Whitney_U_test`

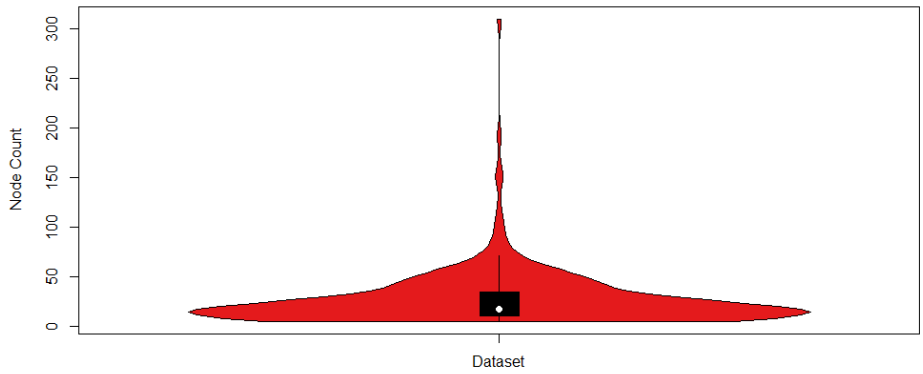


Figure 5.1. Node Count Analysis
This graphs shows the node count distribution in the dataset, excluding the four highest values.

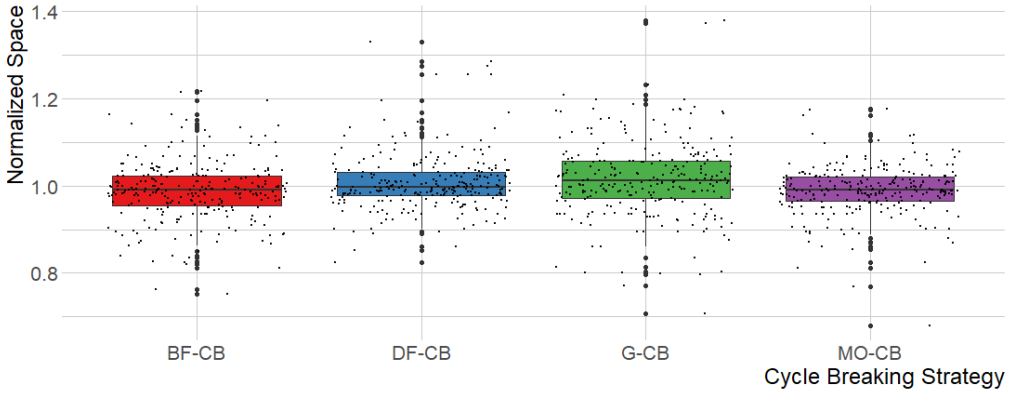


Figure 5.2. Normalized Area Consumption
Comparison of the normalized space consumption per cycle breaking strategy.

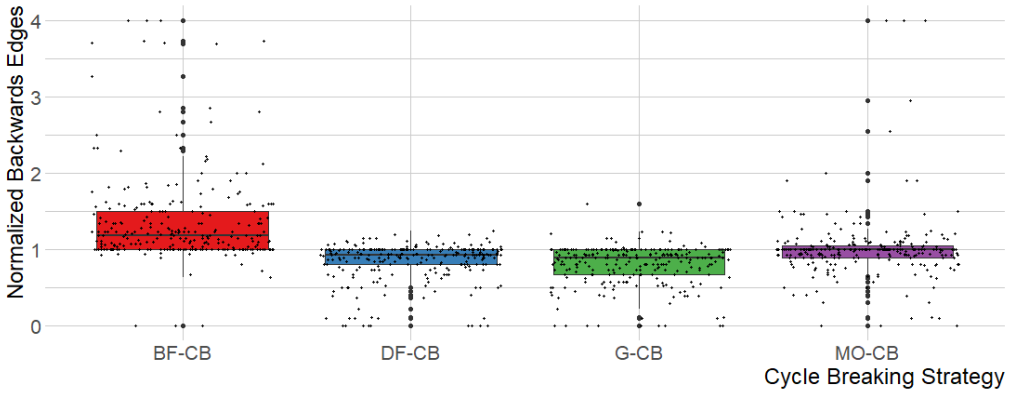


Figure 5.3. Normalized Backward Edge Analysis
Comparison of the normalized backward edge analysis per cycle breaking strategy.

Table 5.2 are the normalized key values of the analysis. Additional data, for this and all other metrics, may be found in Appendix A.

Table 5.2. Raw Statistical Data for Normalized Area Consumption

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	0.990	0.996	1.012	0.990
mean	0.987	1.008	1.015	0.989
min	0.751	0.825	0.706	0.680
max	1.215	1.330	1.379	1.176

The data shows there is very little variance in area difference, which is supported by the Kruskal-Wallis test, with a p-value of >0.98.

5.6 Backward Edge Analysis

Backward edges hinder the readability of a graph, as they break the linearity of the layout direction. Therefore, the fewer backward edges the better. Additionally, fewer backward edges directly correlate with a smaller feedback set. Consequently, this metric also evaluates the heuristic approach of the cycle-breaking strategy. Figure 5.3 presents the normalized backward edge count.
The Kruskal-Wallis test indicates there is a clear difference between at least one set of samples, with a p-value of $3.14 \cdot 10^{-13}$. The Willcoxon test fails to show significant differences in the DF-CB, G-CB, and MO-CB. However, the BF-CB scores significantly worse than the others.
This means that there is no significantly and reliably better heuristic for the minimal feedback set, for this dataset. Further raw data, as well as the results of the Willcoxon test, may be found in the appendix in Section A.2. As previously mentioned, problems could arise if a metric could return zero for all strategies. This is possible for this metric as well, however, the entire dataset does not have an example where this is the case. Therefore, this problem has no impact on this analysis.

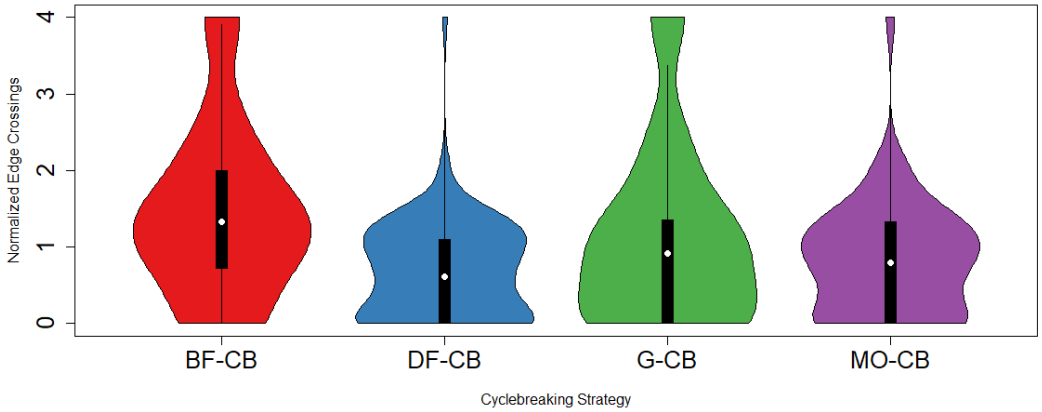


Figure 5.4. Normalized Edge Crossing Analysis
Comparison of the normalized edge crossing analysis per cycle breaking strategy.

5.7 Edge Crossing Analysis

As previously mentioned the edge crossing count is a criterion with a very high impact on clarity. Edge crossings have been minimized by the corresponding phase of the layered algorithm. The Kruskal-Wallis test fails to indicate a significant difference with a p-value of 0.302. However, the majority of graphs (159 of 265) do not induce any edge crossings for any strategy. Therefore this analysis is evaluated again for the remaining 106 graphs, for which at least one strategy contains crossings. The Kruskal-Wallis test now has a p-value of 0.092, again failing to indicate that there is a significant difference. Figure 5.4 shows the normalized evaluation of the smaller dataset. Despite the results of the Kruskal-Wallis test, Figure 5.4 shows that the DF-CB and the MO-CB perform better than the BF-CB and the G-CB. It additionally shows that there are samples for every strategy, where they perform much worse than the average for this sample. There is no single best strategy for all graphs, however, on average the DF-CB performs the best. Raw data may be found in Section A.3.

5.8 Edge Length Analysis

The KIELER *Edge Length Analysis* evaluates the input for three values minimum, average, and maximum edge length. Figure 5.5 presents the normalized results of the average edge length analysis. The raw data is given in Section A.4. Shorter edges tend to be easier to follow, therefore, the shorter the average edge length the better. The Kruskal-Wallis test indicates a difference in the datasets with a p-value of 0.022. The Willcoxon fails to show differences in the DF-CB and MO-CB.
Figure 5.5 shows the best performing strategy is the BF-CB, followed by equally good DF-CB and MO-CB. The G-CB performs the worst.
As previously mentioned, edge length might be influenced by settings in KIELER, e.g. the spacing between layers.

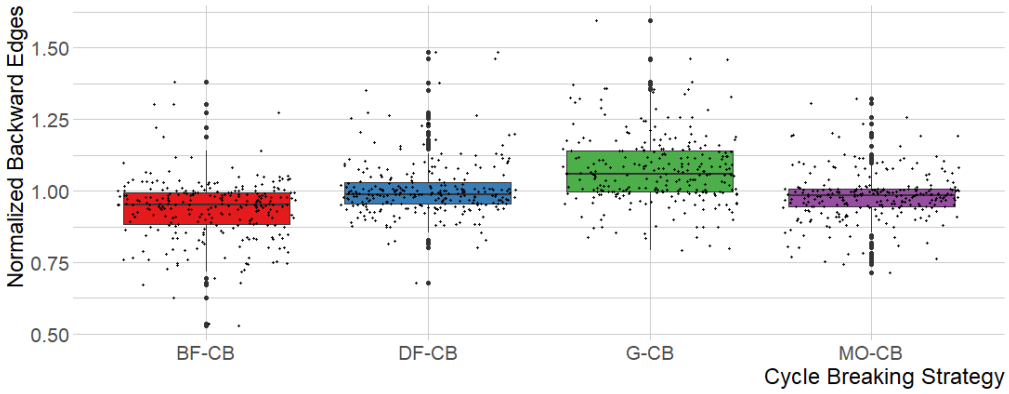


Figure 5.5. Normalized Edge Length Analysis
Comparison of the normalized average edge length analysis per cycle breaking strategy.

5.9 Layer Count Analysis

The layer count is another metric for the layered algorithm. Fewer layers mean that the created layers have a higher node count. For most SCCharts this is desirable, as this means compact drawings. However for some models and other graph types layers with a too many nodes may seem overwhelming. There are some assumptions one could think of regarding the layer count, given by the nature of some cycle-breaking strategies. The nature of the *Breadth First* approach should lead to a small number of layers, while the nature of the *Depth First* approach should lead to a higher layer count.
The Kruskal-Wallis test fails to indicate are significant differences between the strategies, with a p-value of 0.41. Figure 5.6 shows the normalized results of the layer count analysis. It also shows that the DF-CB rarely performs better than the average, and on average performs worse.

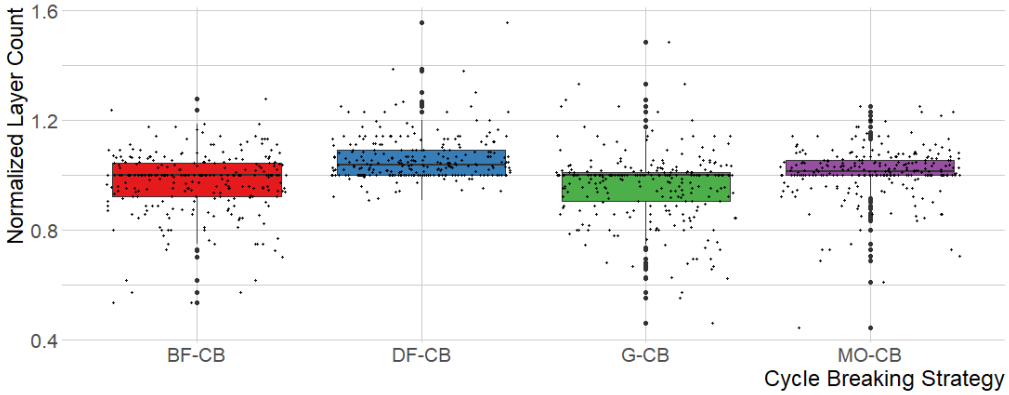


Figure 5.6. Normalized Layer Count Analysis
Comparison of the normalized layer count per cycle breaking strategy.

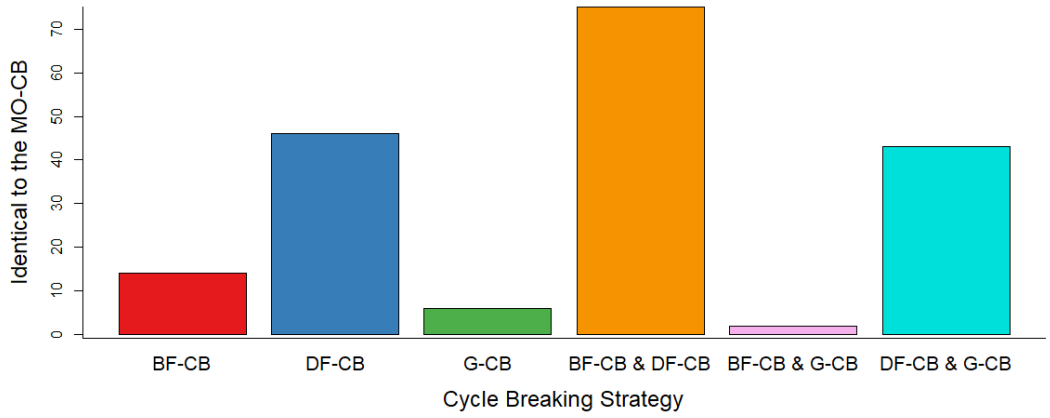


Figure 5.7. Strategy Comparison for Identical Layouts only with the MO-CB The first three bars show how often each strategy creates identical layouts only with the MO-CB. The final three Bars show how often the different strategies produce identical layouts with the MO-CB and one other strategy.

5.10 Model Order

As previously hinted, the question arose if the modeler has a mental model of the layout that is represented in the textual representation. For this, the model order cycle breaker will be compared to the other strategies.
Figure 5.7 shows how often a cycle-breaking strategy created an identical layout only with the MO-CB. The last three bars show how often two strategies produce the same result as the MO-CB. As expected the G-CB shows little to no correlation between the layout it creates and what the MO-CB creates from the textual ordering. The G-CB is not designed to show this behavior. The BF-CB creates very few identical layouts with the MO-CB as well. Both the BF-CB and G-CB, for the majority of the examples, only create identical results with the MO-CB if they also create identical results with the DF-CB. The DF-CB builds identical results as the MO-CB in over 61% of the graphs. As a reminder, this does not include graphs where all the cycle breakers resulted in the same graph. Building this statistic across the entire dataset results in identical graphs in over 75.8% of the graphs.
As previously mentioned, this could mean that a model creator thinks of the graphs in depth-first search nature.

5.11 Combined Analyses

This section was planned to present findings like “Strategy A performs better for smaller but worse for larger graphs than strategy B.” However, the results for such analysis were not providing any reliable or consistent results. As SCCharts allows for multiple edges from one starting node to the same end node, neither the node count nor the edge count is sensible for this kind of analysis. An additional analysis that might be of interest for this would be an analysis that counts distinct node connections and evaluates the previous analyses results against this. A reason for the lack of any correlation might also be that an arbitrary increase in node count does not automatically result in larger graphs or in graphs with more edge crossings.

5.12 Unique Layout Analysis

As mentioned, graphs that produced identical layouts for all strategies were cut from the dataset. While this eliminated over 50% of the dataset, this was done to emphasize the differences produced by the different strategies. However, as Section 5.10 showed and as mentioned in the introduction to the dataset, there are still many examples where at least two strategies create identical layouts. Each analysis shown in this chapter was additionally run for the 47 graphs that result in unique layouts for all strategies. As these results have very little power due to their small sample size and their results are very similar to the results already presented in this chapter, these results will not be shown here.

Chapter 6

Empirical Study

This chapter presents the structure and analysis of the aforementioned empirical study. The survey is split into three parts, starting with some questions about the participants, followed by three detailed graph examination tasks and five short question rounds. The following sections present these stages and present an exemplary question structure. Finally, evaluations about the entire set of responses will be given. For this section a lot of data is available. Each question type will be evaluated in depth at least once. For this, a statistical graph is given. Additional raw data and some graphs may be found in Appendix B.

6.1 The Survey and Participants

The survey was created with the tool LimeSurvey[1], which was provided by Kiel University. It contained 35 questions with an average time to completion of 30 minutes. Of the 33 participants, 27 finished the survey. The remaining 6 partial submissions will be disregarded for the analysis. Of the 27 participants 17 work or study in the IT sector. 14 participants answered that they previously worked with statecharts. 8 participants worked with the layered layout approach prior to this survey. The final question about the participants was how confident they feel in the topic of automated graph layout, using a five-level Likert scale. The results are shown in Table 6.1. After the questions about the participants, some terminology was introduced. As this survey was also targeted towards participants without any prior knowledge, most terms were simplified.

Table 6.1. Results of the participants confidence in the topic of automated layout creation.

	No confidence	Not confident	Neither nor	Confident	Very confident
Answers	9	4	6	4	4

6.2 Question Structure

The graphs used in this survey are graphs from the dataset presented in the previous chapter. However, since the research goal is to differentiate between the different cycle-breaking strategies, only graphs with a different layout for each strategy were used. All options for a graph were presented at the same time in a grid. The position of the strategies in this grid was randomized. Figure 1.1 shows how the options were presented in the survey. To compare the graphs, the participants were asked to order the options. The survey contained an optional text response field for each graph.

[1] `https://www.limesurvey.org/de/`

6.3 Introductory Graph Questions

The first three graphs were used to introduce the topic. For these graphs, the following five questions were asked.

- ▻ First Impression - Take a quick look (≈30s) at the different layout options and rank them according to your preference, starting with the best.
- ▻ Readability I - Do you feel like the states are too crowded or too far away from each other?
- ▻ Readability II - Evaluate for each graph: Edges are easy to follow?
- ▻ State Grouping - Take a look at the node labels. Do you feel like any layout option creates groups of nodes that are thematically related?
- ▻ Final Impression - After working with this graph repeat the evaluation of your preference. Starting with the best.

The first and last questions were ordering questions. The questions concerning readability were five-level Likert scales. For Readability I the scale had the following labels: 1 - Too crowded , 2 - Crowded , 3 - Good, 4 - A little too far, 5 - Too far. For Readability II the scale was labeled as follows: 1 - Strongly disagree, 2 - Disagree , 3 - Neither agree nor disagree, 4 - Agree, 5 - Strongly agree. The question about thematically related grouping was a multiple-choice question. For the ordering a performance metric may be defined as follows:

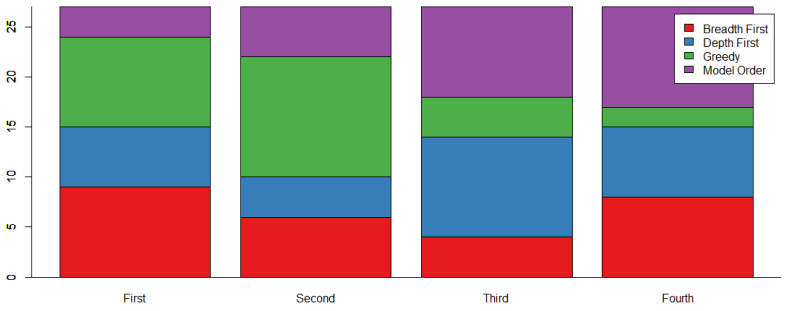
6.3.1 Definition (Ordering Performance)**.** Let $r_i, 1 \leqslant i \leqslant 4$ be the amount of votes a certain strategy s has received in rank i, with 1 representing the best rank and 4 the worst. Therefore, the average (mean) performance of a strategy is given by

$$performance(s) = \Sigma_{i=1}^{4} \frac{r_i * i}{\text{total votes}}$$

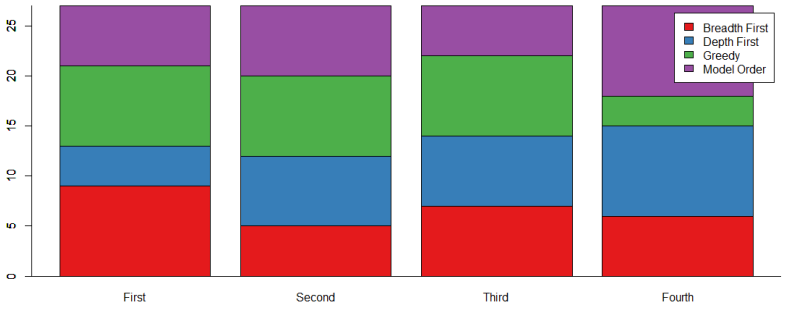
Even though there was no hard limit on how much time participants could spend on a question, after a certain amount of time (30 seconds for the first and last question, 1 minute for the remaining questions) a message would display that the intended time had passed. This time limit was meant to help with the survey completion rate, because this should decrease the time to get more progress, which was shown at all times.

6.4 Graph 1

The first graph is shown in Figure 1.1. The examples differ in the number of backward edges and the size. This graph was chosen because of its small size and its lack of complexity. Figure 6.1a shows the responses to the first question (ordering the options) and Figure 6.1b shows the responses to the final question of Graph 1. The performance taken from the first impression shows that the G-CB performs the best with an average performance of 1.96. This also performs the best concerning backward edges, with only one backward edge (together with the DF-CB). The smallest layout, which was created by the MO-CB, performs the worst with an average of 2.96. Figure 6.1b shows how the ordering has changed after asking further questions. The performance values of this ordering are much closer together. The G-CB still performs the best, however, the difference to the remaining strategies is much smaller. Of the 27 participants, only nine ordered the graphs the same way as before. The results of Question 3 directly oppose an objective metric. The smallest option appeared too crowded for the majority of the



(a) First Impression Rating.



(b) Final Impression Rating.

Figure 6.1. Comparison of the Initial Rating and the Final Rating

participants, scoring an average of 2.07 on the Likert scale.

6.5 Graph 2

The examples presented in Figure 6.2 were one of the starting points for this thesis. The BF-CB option of this example was highly debated in the Real-Time and Embedded Systems group. This option introduces four additional edge crossings compared to the remaining options, however, one could argue that the BF-CB creates thematic node groups. For some, this is an acceptable trade-off. For others, this breaks principle too much and the introduced edge crossings are unacceptable. The survey results mirror this big disagreement for this example. The votes are almost entirely split 50-50 between rank one and rank four, as seen in Figure 6.3. The final impression may be found in the appendix in Figure B.1. This disagreement may be found in both groups of participants, the group working in the IT sector and the ones who are not. Actually, there is little to no difference between these two groups for this example. The BF-CB performed the worst for both Readability I and Readability II, with 2.37 and 2.59 respectively. 18 participants answered that the BF-CB created thematic groups and 11 of those chose the BF-CB as first, confirming that for some this break in principle is justified by introducing

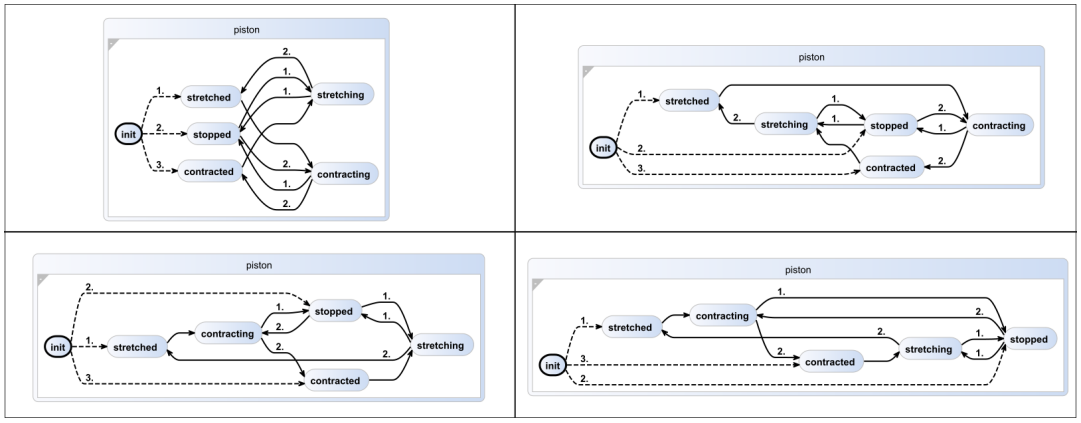


Figure 6.2. Second Graph used in the Survey. Top Left - BF-CB, Top Right - MO-CB, Bottom Left - DF-CB, Bottom Right - G-CB

thematic groups. The best performing option objectively and subjectively is the G-CB.

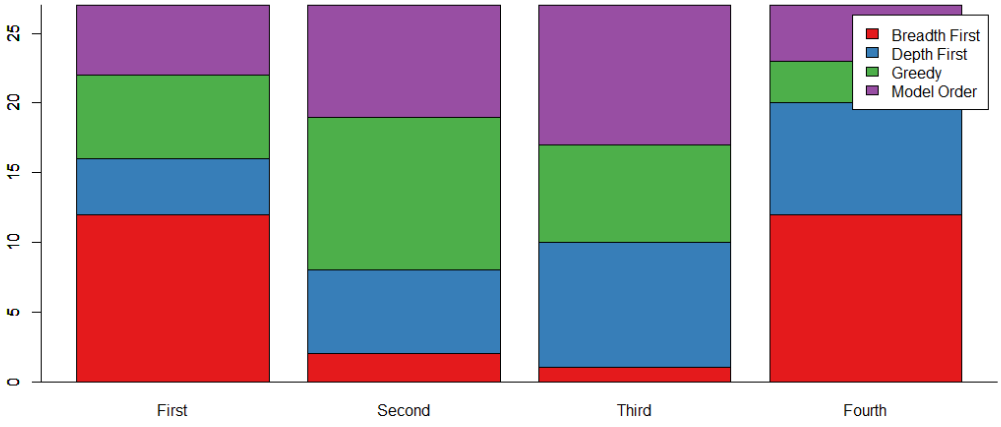


Figure 6.3. First Impression Ordering of Graph 2

6.6 Graph 3

The third graph used in the survey was chosen to introduce bigger graphs with higher complexity. This example is too big to be presented well here, therefore the individual options are shown in the appendix in Section B.2. Many free text responses mention that it is aesthetically pleasing to have error states on the far right, which is something the G-CB and the DF-CB will do automatically with all sinks. Additionally, the text responses showed that the initial state should be in the first layer, as this was a big criticism for the G-CB. The ordering results presented in Figure 6.4 are very different compared to those of Graph 1 and Graph 2. While the choices for the first two graphs seemed somewhat evenly

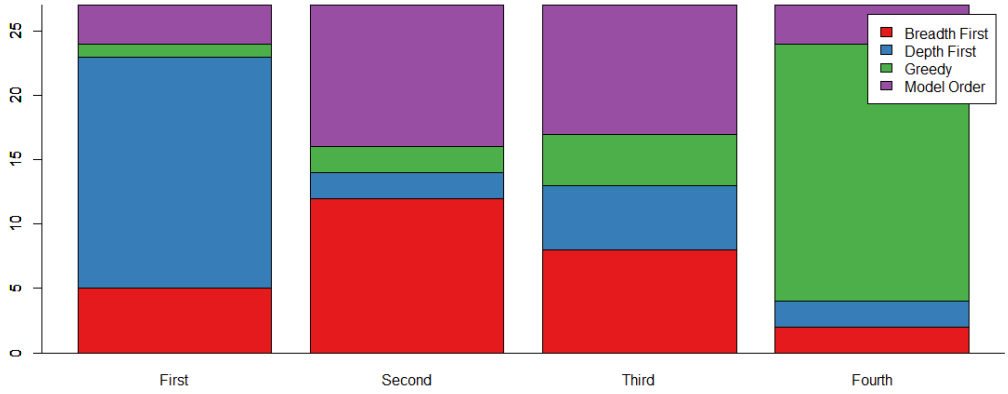


Figure 6.4. Final Impression Ordering of Graph 3

distributed the opposite is true for the third graph. For this graph, there is a clearly winning and a clearly losing graph regarding performance. With an average of $1.\overline{6}$ the DF-CB scores best with a performance difference of over 0.5 points ahead of the BF-CB with 2.25 points. Even larger is the gap between the G-CB, which performs worst, with an average of 3.59 points, and the MO-CB with 2.48 points. This is in line with the objective evaluation of this graph. The DF-CB scores best for backward edge count and for edge crossings, while the G-CB scores by far worst for edge crossings. The MO-CB and the BF-CB score fairly similar. The Readability I & II answers reflect the ordering presented in Figure 6.4 as well, as the G-CB scores worst in both. This graph introduces a feature for the G-CB, which most participants seem to dislike. This feature is called dangling nodes and defined as follows:

6.6.1 Definition (Dangling Nodes)**.** A node is called *dangling* node if a node, which is not the a source, is placed in a layer without any incoming edges from a previous layer (a layer with a lower layer id).

Even though the free text responses do not mention this for this example it is reasonable to assume that this also had a negative impact on this graph. In the later graphs this is mentioned frequently in the free text responses.

6.7 Evaluation of the Remaining Graphs and Comparison with the First Three Graphs

The following sections will present the results for Graphs 4 to 8. For these graphs, only the first question (First Impression) of the five mentioned questions was asked. This was done to evaluate a larger number of models without prolonging the survey too much. The assumption was taken that the participants would incorporate the mentioned criteria after using them three times. Additionally, for all the graphs a free text response was optional.
To compare the following graphs with the first graphs a decision has to be made to either take the First Impression or the Last Impression of the first three graphs. On average 18 participants changed their ordering for the first three graphs, however, there were no clear trends in this reordering. For

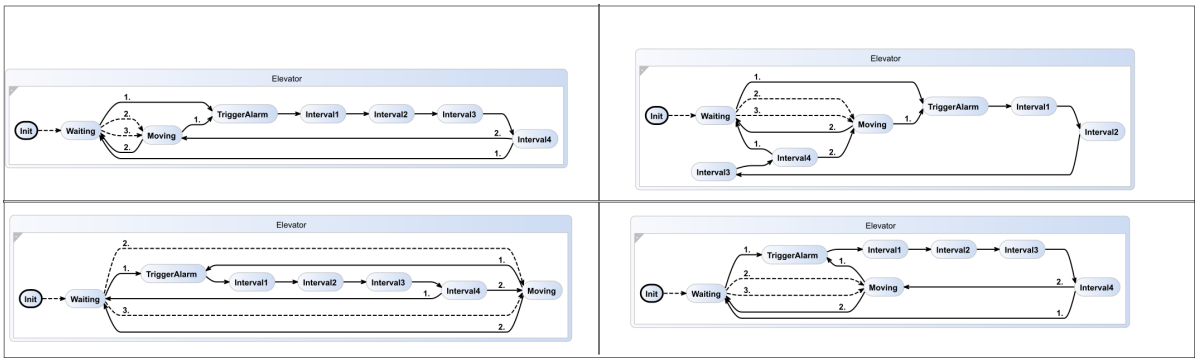


Figure 6.5. Fourth Graph used in the Survey Top Left - MO-CB, Top Right - G-CB, Bottom Left - DF-CB, Bottom Right - BF-CB

analysis across the entire dataset, the Final Impression results of the first three graphs will be used. The following graph orderings are similar to the ordering of Graph 3 in their decisiveness.

6.8 Graph 4

The fourth example, which is presented in Figure 6.5, does not have any edge crossings and all have the same backward edge count. The options differentiate mainly in edge length. Additionally, two layouts (G-CB and BF-CB) create one layer less than the other two. The ranking, which may be found in Figure B.6 in the appendix, shows that the G-CB and the BF-CB perform worse despite creating fewer layers. As seen in Figure 6.5 the G-CB has a node dangling node “Interval3”. This is something many free text responses give as a reason for their low rating for this option.
Generally, a consistent direction is something that most participants feel is important. The DF-CB and the MO-CB perform very similarly, an indication that pure edge length has little impact on the final decision.

6.9 Graph 5

The fifth graph introduces a new layout direction, from top to bottom. The options differentiate a lot for this graph. As this graph is rather large the options are, together with the ordering, presented in the appendix in Section B.4. Graph 6 shown in Figure 6.6 also has the *DOWN* layout direction. The G-CB creates a small layout, however, it creates more edge crossings than the rest. The DF-CB creates a tall layout with high symmetry in some parts of the graph. The result of the MO-CB is similar to the result of the G-CB in size, however, it produces only one edge crossing like the DF-CB and it has higher symmetry. The BF-CB creates a similar result to the DF-CB, however, it creates two edge crossings and the textual responses mainly differentiate these two because of the lack of symmetry in the BF-CB. For this example again the G-CB performs significantly worse than the remaining strategies, with 80% of the participants selecting this strategy in the last place. This is the highest percentage for any graph and any position. It achieves a performance of 3.74 points. The G-CB in creates multiple dangling nodes for this example. Interestingly, despite dangling node (“empty”) that the MO-CB has it still performs better than the BF-CB, with a respective performance of 2.04 points and 2.52 points. The DF-CB performs the best with a performance of 1.70 points.

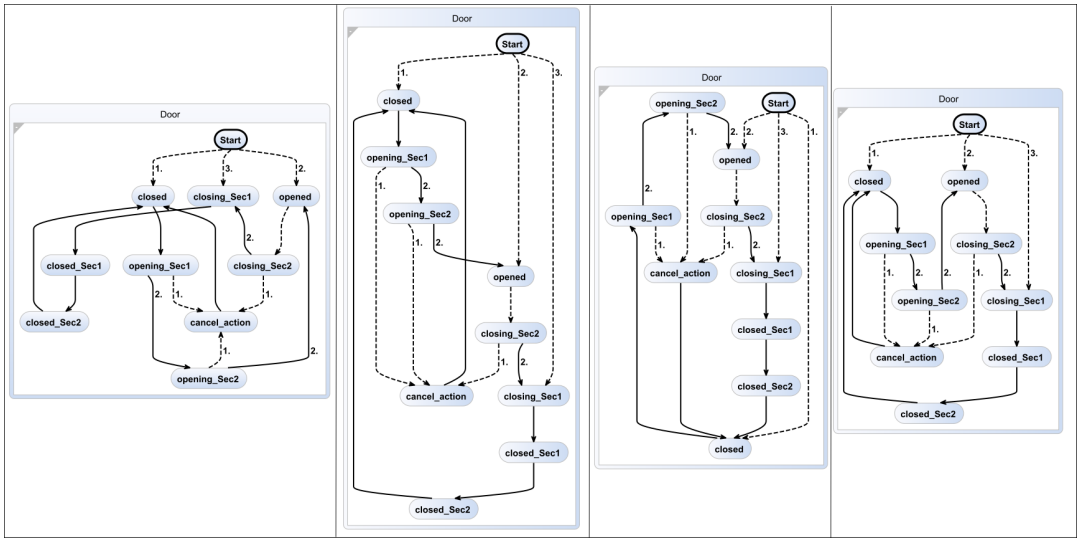


Figure 6.6. Graph 6 of the Survey The second graph in the survey that uses the *DOWN* layout direction. First - BF-CB, Second - DF-CB, Third - G-CB, Fourth - MO-CB

6.10 Graph 6

The sixth graph shown in Figure 6.6 is, as mentioned, the second graph with the *DOWN* layout direction. The G-CB and the MO-CB create layouts without any edge crossings, while the layout created by the DF-CB has one edge crossing and the BF-CB creates two edge crossings. The worst performing strategy is the BF-CB with an average of $3.\overline{4}$ points. The best performing option the DF-CB, despite creating an unnecessary edge crossing. The edge crossings in this graph are, however, at a great angle and therefore do not interrupt the flow too much. With a performance of 1.81 points, this option is roughly 0.4 points ahead of the MO-CB with 2.19 points. The G-CB has a performance of $2.\overline{5}$. Something the free-text responses mention again is that the G-CB has a dangling node in the same layer as the initial node (the node “opening_Sec2”), which is something most participants seem to dislike. The ordering may be found in Figure B.10

6.11 Graph 7

The seventh graph used in the survey, which is shown in Figure 6.7, emphasizes symmetry. The option with the highest symmetry, the MO-CB, clearly wins this ranking with a performance of 1.59 points. The textual responses clearly indicate that the thematic groups created in this option had a big impact. This is confirmed by taking a look at the BF-CB. This shows a little less symmetry and has a performance of 2.00 points. This option is lacking the thematic group in the third layer, however, this option only has four edge crossings instead of seven, which the MO-CB has. For this example, the additional edge crossings are a reasonable trade-off for higher symmetry and better thematic grouping, as the angle of these crossings is big enough to reliably follow the flow. The DF-CB shows very little symmetry, however with a performance of $2.\overline{8}$ points it still performs significantly better than the G-CB with 3.52 points. The G-CB again has additional nodes in the same layer as the initial node.

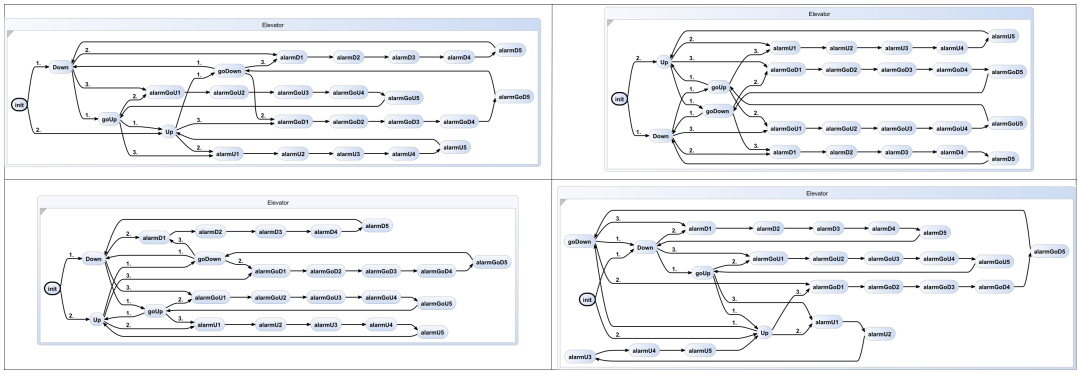


Figure 6.7. Graph 7 of the Survey Top Left - DF-CB, Top Right - MO-CB, Bottom Left - BF-CB, Bottom Right - G-CB

6.12 Graph 8

The last graph of the survey has an unusually high edge count compared to the node count. This is because for most nodes multiple connections to the same node exist. This graph is too big to be shown in a grid, therefore the individual options, together with the ordering, may be found in Section B.6. The G-CB is quite different from the other strategies, as it does not strictly has a flow from left to right. It again creates a dangling node (“evalDigit”), while the others do not. This option again performs the worst with a performance of 3.37 points. The remaining options mainly differ in the placement of the final node (“stop”) and the error node. The BF-CB has these nodes in the middle of the graph and ranks third with a performance of 2.85 points. The MO-CB places the error node in the last layer and has a total of 13 edge crossings. It performs second best with a value of 2.00 points. The DF-CB performs the best with an average of $1.\overline{7}$ points. This option has the error state in the layer before the final node. Textual responses show that it is a big benefit if an error state is somewhat segregated from the remaining graph.

6.13 Analysis for All Graphs

In this section, some analysis for all graphs used in this survey will be given, first combining all graph orderings, without looking at the quantitative statistics of the different graphs. This unbiased combination ranking is shown in Figure 6.8. This shows that the average performance of the MO-CB is the best, with a performance value of 2.11 points. This is very closely followed by the DF-CB with an average performance of 2.17 points, which is a difference smaller than 3%. This small difference does not indicate anything in a sample size as small as this survey provides. Therefore, the MO-CB and the DF-CB share the first place. This is followed by the BF-CB with a performance of 2.62 points. And finally, the G-CB with an average performance of 3.11 points. Figure 6.8 shows the results for simply adding all ranking values per graph. This representation still holds information about the ratio for each individual graph, Figure 6.9 simplifies this information. For this graph, a ranking has been created from the average performance values. This second representation shows how often which strategy ended up in what rank, ignoring the magnitude between ranks. It shows that the MO-CB, which in this case has an average performance of 1.875 points, was never the worst-performing graph. Also the BF-CB, with an average performance of 2.75 points, despite never creating the best option only

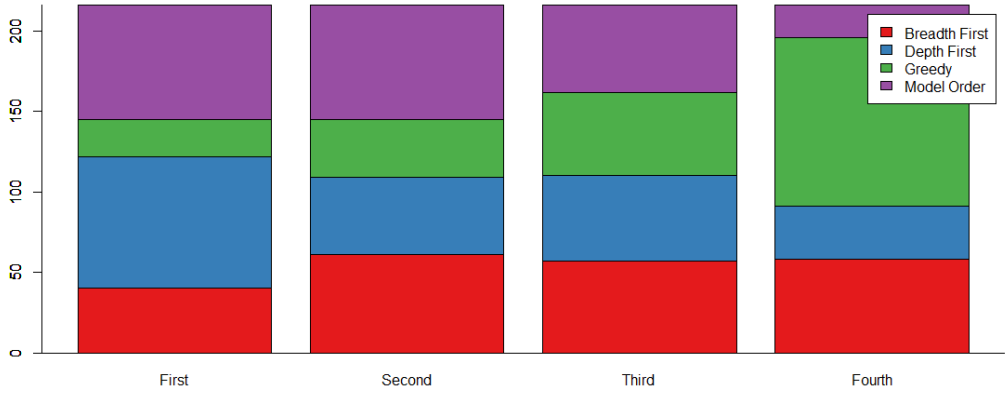


Figure 6.8. Unbiased Ranking of All Graphs Ignoring the quantitative differences and adding the ranking votes of each option per graph.

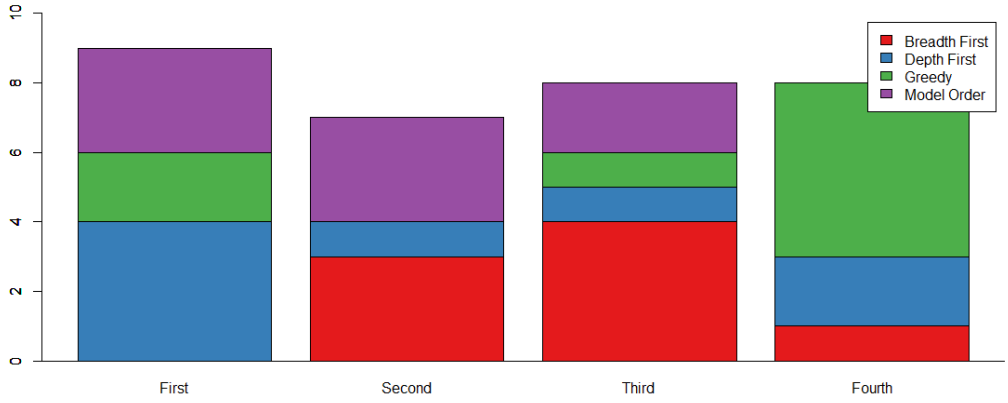


Figure 6.9. Strategy Rank per Graph Taking the average performance and creating a ranking from this per graph. The height difference is because the G-CB and MO-CB perform identical in Graph 2.

once created the worst option, leading to third place. The DF-CB shows mixed results with an average of 2.125 points. It creates the highest count of best-performing options, however, it also creates some worst-performing options. Interesting are the results for the current default cycle-breaking strategy, the G-CB. This has the worst average performance of 3.125 points and it creates the worst options for the survey in the majority of the graphs. Interestingly, the first places it creates are the first two graphs, which are rather small. For the larger graphs, this strategy does not seem to create visually pleasing results. Additionally, one of the first places of the G-CB is shared with the option created by the MO-CB. This double first place is the reason for the height difference in the bars, as for this example, two first places were assigned and no second place.

Table 6.2. Importance of the Introduced Criteria

	Very Unimportant	Unimportant	Neither nor	Important	Very Important
Clear Edges	1	0	3	15	8
Crowdedness	2	4	8	8	5
Thematic Grouping	2	5	1	11	8

6.14 Importance Ranking of the Criteria

The final two questions concerned the decision driving criteria as perceived by the participants. Firstly the participants were asked to rate how important the criteria mentioned in the first three graphs were to them, using a five-level Likert scale. The results to this question are presented in Table 6.2. These results show that while clear edges are the most valued metric for the majority of people, it is also important that the nodes that have a similar theme should be grouped. This could be a reason why the MO-CB performed as well as it did in this survey and why the G-CB performed as badly as it did. Finally, the last question of the survey was a free text question asking for additional criteria, which influenced the decision-making of the participant. The following list represents the top answers to this question, ordered by the count of answers mentioning them:

- Initial Node in the First Layer and Final Node(s) in the Last Layer (7 Responses)
- Symmetry (5 Responses)
- Edge Crossings (4 Responses)
- Understandable Node Labels (3 Responses)

Chapter 7

Conclusion

This chapter presents some conclusions from the previous two chapters. First, however, some points of criticism concerning the evaluation will be given.

7.1 Points of Criticism

For quantitative analysis to be meaningful, the dataset has to be very large. As previously stated, of the 419 SCCharts of the original dataset, only 265 are useful for the quantitative analysis. Especially as these graphs are vary so much in size it would be nice to rerun this analysis once an even larger dataset is available. The same is true for the survey results. With 27 submissions the data is very limited. A larger test audience would be great, however, it is very difficult to have a balance in this group between people with and without prior knowledge in automatic graph layout. People with prior knowledge are hard to find, and for people without prior knowledge the interest is very limited. As stated previously, of the graphs in the dataset, only 47 graphs resulted in unique layouts for each strategy. For most of these graphs, the different options have multiple differences, making it very hard to distinguish what was the driving factor in the decision. The free text responses helped a lot for this. The biggest point of criticism, however, concerns the creation of the graphs. They have been created using the *Greedy Cycle Breaker*. This could lead to a better performance of the G-CB. A dataset with a different default strategy or even in a dataset with evenly distributed used strategies the results might be very different. There currently is no convenient option to change the cycle-breaking strategy in KIELER.

7.2 Conclusion of the Comparison

The first strategy discussed here is the ***Greedy Cycle Breaker***, as this is currently the default strategy for SCCharts. The G-CB is designed to create minimal backward edges, which it does reliably, as the quantitative analysis shows. However, this comes with some drawbacks, for the remaining quantitative metrics, which showed some significant differences, this strategy performs poorly. On average it creates more edge crossings than the MO-CB and the DF-CB, and on average it creates the longest edges. The last major point of criticism is the total lack of secondary notation, as the edge reversal seems rather arbitrary to a human. The model order of the textual input has no reliable influence on the final layout. It performed significantly worse in the survey than the remaining options. This is especially interesting regarding the criticism brought forth in the previous section. Additionally, this strategy created layouts where nodes are placed in a layer without a connection to any previous layer, the survey results show that this has a really big influence on the clarity of a layout.
Following that, the next strategy discussed is the ***Breadth-First Cycle Breaker***. The only benefit of this strategy is compactness. It creates graphs with fewer layers and shorter edges. However, this compactness comes at a cost, resulting in more backward edges and more edge crossings. This strategy

might be best described as mediocre. It never creates the best and only once creates the worst option in the survey. For most participants, compactness seems to have little effect on the decision, if it was mentioned in the responses, it mostly was because the graph felt too crowded. However, there might be some use-cases where space consumption is critical. The option for the second graph of the survey, which sparked the idea for this strategy, may be created with the MO-CB and the correct ordering.
Next up are the ***Depth-First Cycle Breaker*** together with the ***Model Order Cycle Breaker***. The DF-CB and the MO-CB perform very well in the quantitative analysis, in most cases performing better than the other options. As previously evaluated, these two strategies create identical layouts in the majority of the graphs for this dataset. Again this could hint towards the hypothesis that a model creator thinks about the graph in a depth-first way. These two strategies additionally perform very well in the survey. It is again worth noting that the MO-CB never created the worst option in the survey. The major benefit of the MO-CB over the DF-CB is the direct influence the creator can have on the layout. This of course enables the creation of thematic groups of nodes.
Finally, the results of the quantitative analysis and the results of the survey show that the *Greedy Cycle Breaker* may not be a good default option. While it is nice to decrease backward edges, these did not seem to have such a big impact on the decision. The evaluation data presents a clear direction towards choosing the *Model Order Cycle Breaker* as the default option for SCCharts.

7.3 Future Work

As mentioned, it would be interesting to repeat this analysis with a larger dataset, or even with other models and not only SCCharts.
It would be nice to have clear differences in the different options. This could then be used to create a ranking of the different criteria with a higher accuracy.
As the conclusion here presents some clear advantages for the Model Order Cycle Breaker, it would be very interesting to see if this strategy would perform even better, if the graphs would have been created using the MO-CB.
If there was an easy way to set the cycle breaking strategy in KIELER, an additional analysis would be how often a model creator actively chose a certain strategy, which is not the default. It was not possible to see if a certain strategy performs better for models with certain features like a higher node count, as mentioned in Section 5.11. It would be interesting to test if there are different features that show such a performance difference. Features that come to mind are connectedness of the graph or the (average) number of edges per node.

Appendix A

Additional Data for the Objective Analysis

In this chapter of the appendix raw data of some quantitative analysis criteria is given.

A.1 Area Raw Data

Table A.1. Raw Statistical Data for Area Consumption

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	345011	351554.2	351554.2	349111.3
mean	23691232	23330748	34013900	24231488
min	46533.6	46533.6	48164.03	46533.6
max	4728596748	4612521029	7431536979	4877245845

A.2 Backwards Edge Raw Data

Table A.2. Raw Statistical Data for Backwards Edge Analysis

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	5	4	4	4
mean	7.392308	4.342308	4.111538	5.026923
min	0	0	0	0
max	42	20	25	22

Table A.3. Raw Statistical Data for Normalized Backwards Edge Analysis

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	1.176471	0.9230769	0.8888889	1
mean	1.35234	0.84074	0.8015928	1.005328
min	0	0	0	0
max	4	1.238095	1.6	4

Table A.4. Evaluation Results for the Willcoxon Test for the Backwards Edge Analysis with a Kruskal-Wallis p-value of $3.14 \cdot 10^{-13}$, adjusted with the Bonferroni correction. Significant differences are marked with green cells. The value does not express any relation for better or worse, it just indicates a significant difference.

	BREADTH FIRST	DEPTH FIRST	GREEDY
DEPTH FIRST	$< 2 \cdot 10^{-16}$	-	-
GREEDY	$< 2 \cdot 10^{-16}$	0.61	-
MODEL ORDER	$4.1 \cdot 10^{-16}$	$1.1 \cdot 10^{-12}$	$2.5 \cdot 10^{-10}$

A.3 Edge Crossing Raw Data

Table A.5. Raw Statistical Data for Edge Crossing Analysis

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	0	0	0	0
mean	17.70189	8.173585	13.22264	9.864151
min	0	0	0	0
max	2541	682	1793	1012

Table A.6. Raw Statistical Data for Normalized Edge Crossing Analysis

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	0	0	0	0
mean	0.5752295	0.2679906	0.4305159	0.326264
min	0	0	0	0
max	4	4	4	4

A.4 Edge Length Raw Data

Table A.7. Raw Statistical Data for Average Edge Length Analysis

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	127.7435	148.5468	155.4871	143.7422
mean	198.1576	211.5645	221.7893	207.9055
min	45.5601	45.5601	54.82275	45.5601
max	2834.549	2834.549	2834.549	2835.743

Table A.8. Raw Statistical Data for Normalized Average Edge Length Analysis

	BREADTH FIRST	DEPTH FIRST	GREEDY	MODEL ORDER
median	0.9520167	0.9883364	1.058654	0.9832882
mean	0.9371895	1.005848	1.073359	0.9836027
min	0.5302196	0.6795208	0.7935748	0.7150965
max	1.381669	1.48605	1.594989	1.320479

Table A.9. Evaluation Results for the Willcoxon Test for the Average Edge Length Analysis with a Kruskal-Wallis p-value of 0.22, adjusted with the Bonferroni correction. Significant differences are marked with green cells. The value does not express any relation for better or worse, it just indicates a significant difference.

	BREADTH FIRST	DEPTH FIRST	GREEDY
DEPTH FIRST	$< 6.8 \cdot 10^{-10}$	-	-
GREEDY	$< 2 \cdot 10^{-16}$	$< 3.7 \cdot 10^{-8}$	-
MODEL ORDER	$6.2 \cdot 10^{-6}$	0.028	$2.8 \cdot 10^{-11}$

Appendix B

Additional Data for the Survey

In this chapter some additional data of the survey is presented. This includes graphs that are too big to include in the main part of this thesis.

B.1 Graph 2

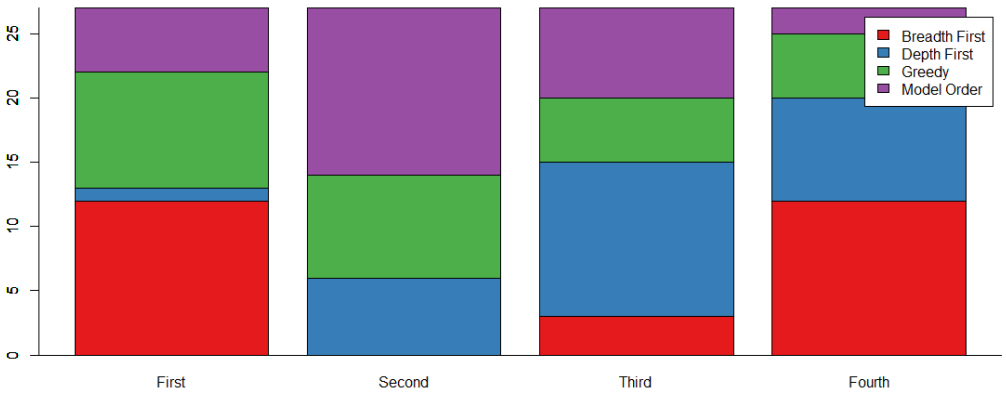


Figure B.1. Final Impressions of the second Graph

B.2 Graph 3

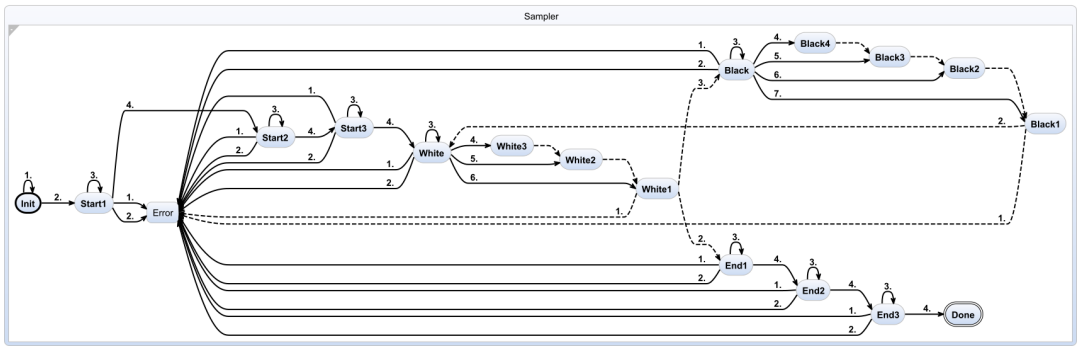


Figure B.2. Graph 3 BF-CB

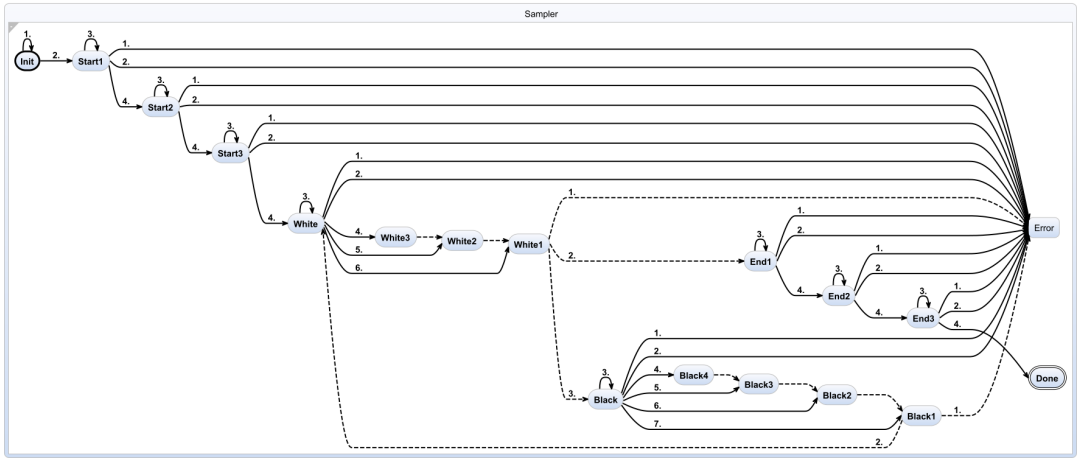


Figure B.3. Graph 3 DF-CB

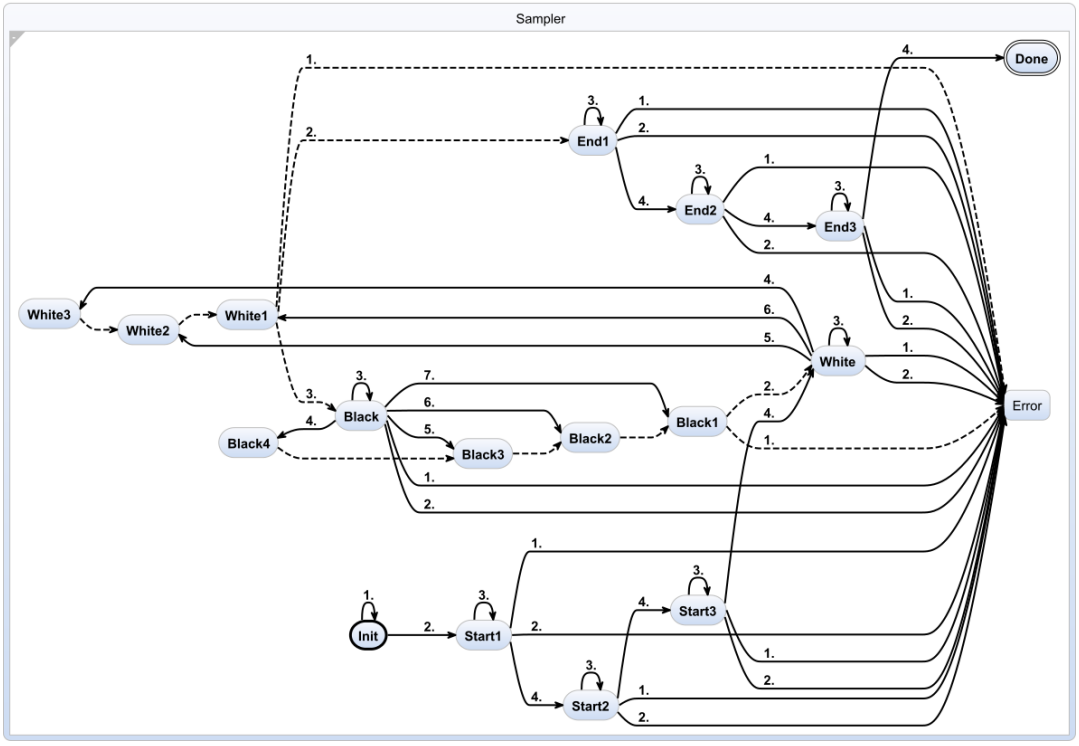


Figure B.4. Graph 3 G-CB

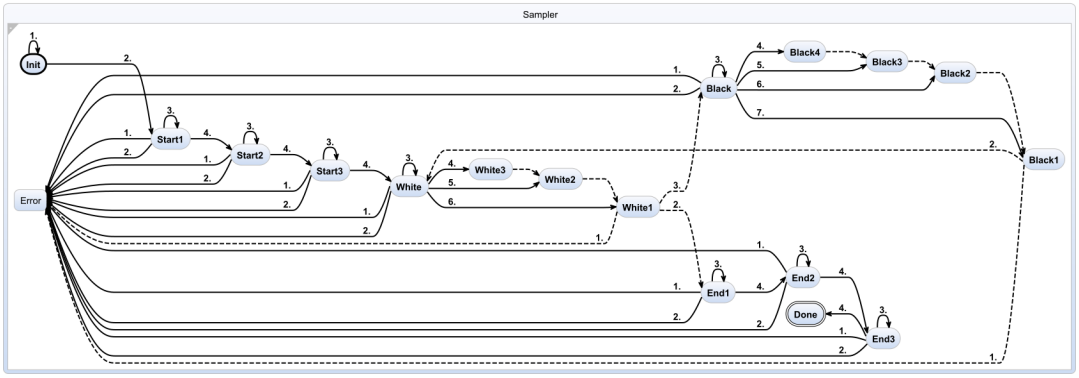


Figure B.5. Graph 3 MO-CB

B.3 Graph 4

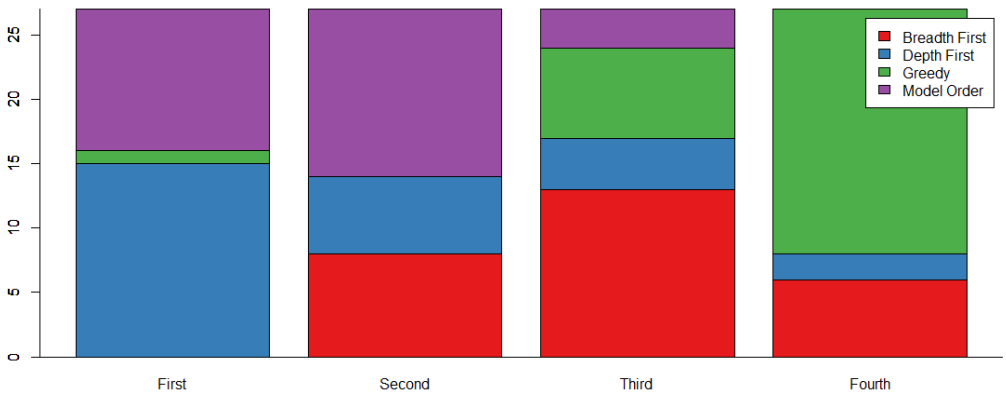
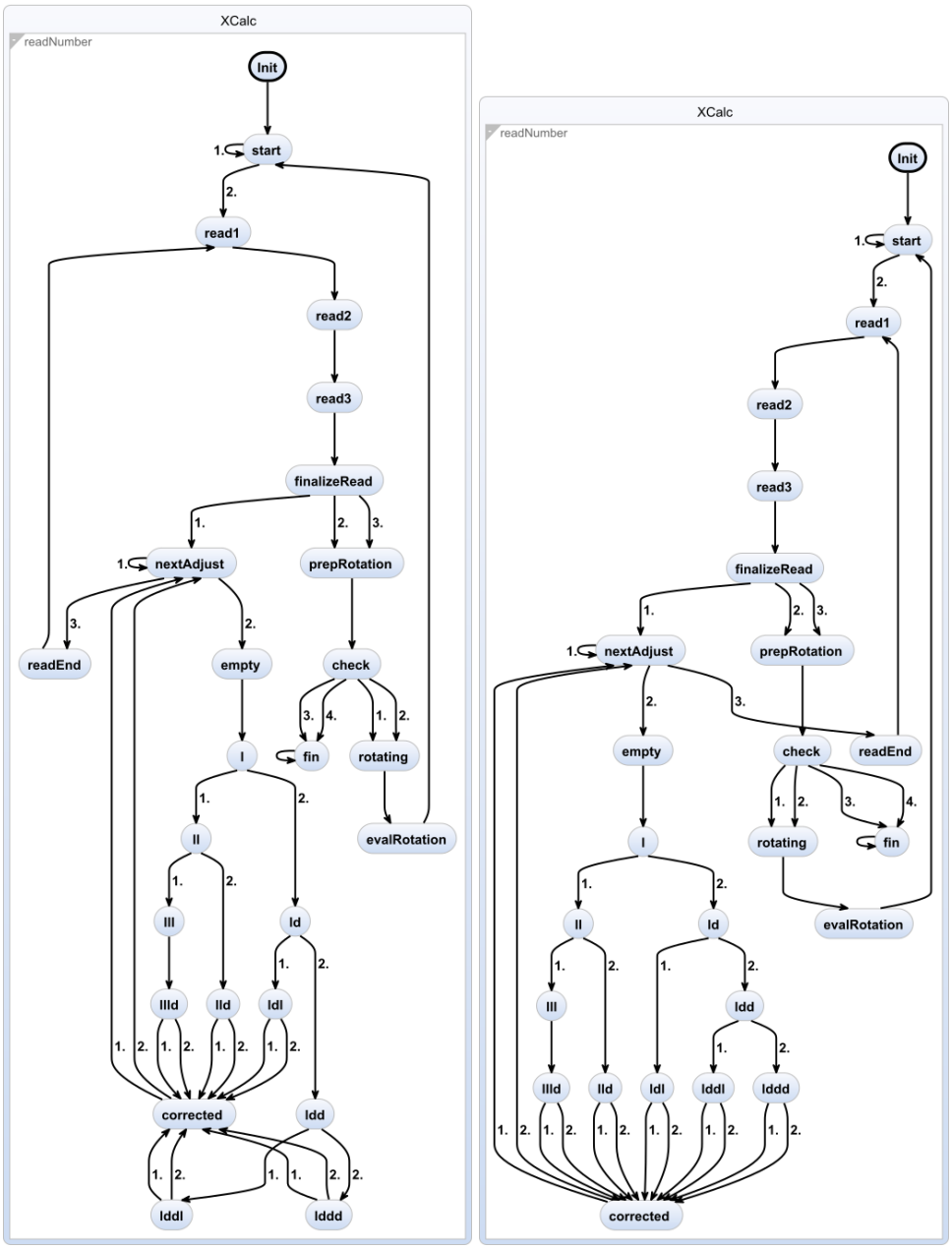
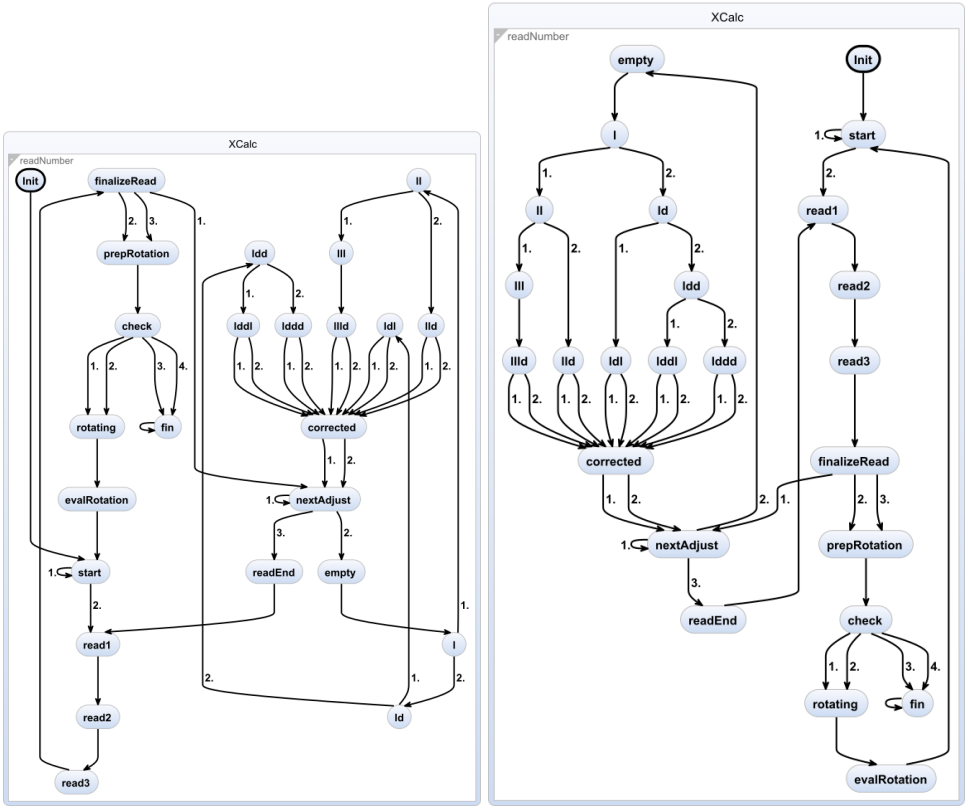


Figure B.6. Graph 4 Impression Ranking

B.4 Graph 5



(a) Graph 5 BF-CB **(b)** Graph 5 DF-CB



(a) Graph 5 G-CB **(b)** Graph 5 MO-CB

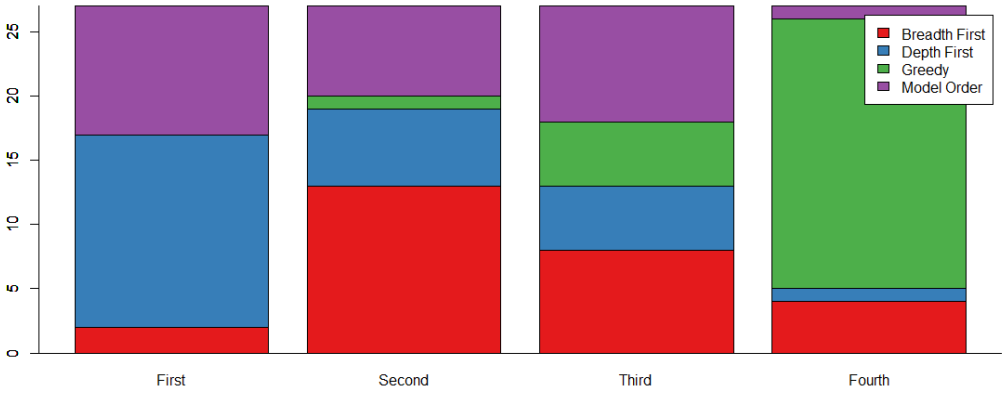


Figure B.9. Graph 5 Impression Ranking

B.5 Graph 6

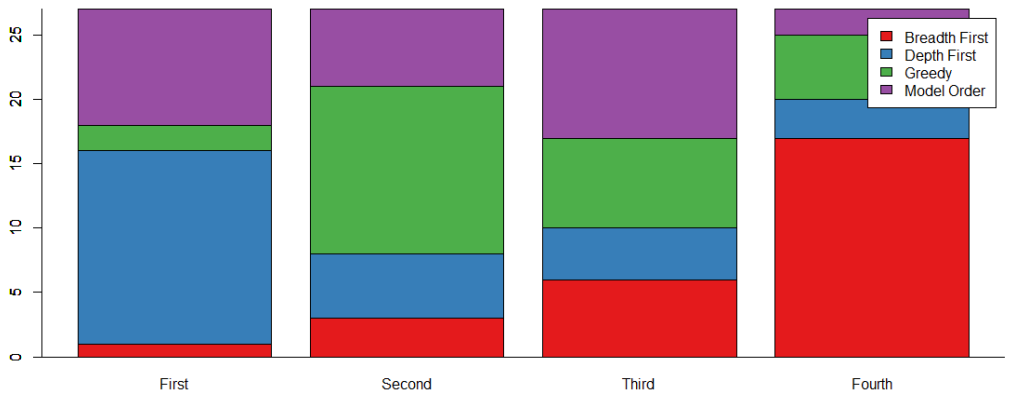


Figure B.10. Graph 6 Impression Ranking.

B.6 Graph 8

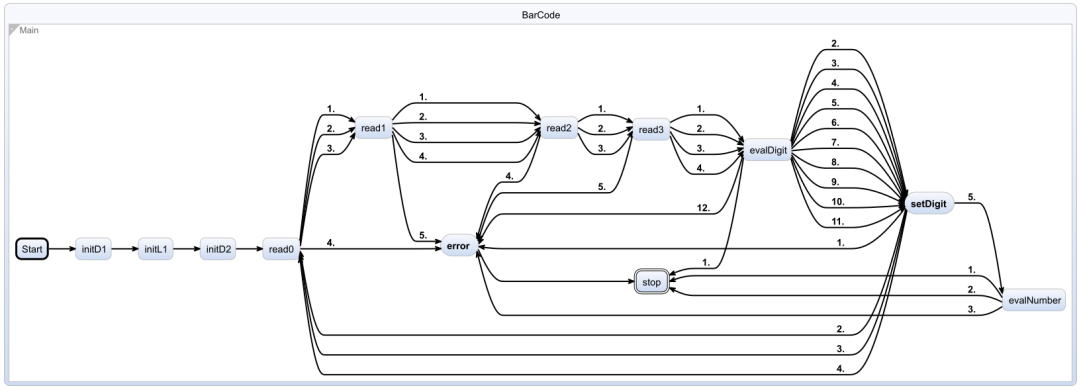


Figure B.11. Graph 8 BF-CB

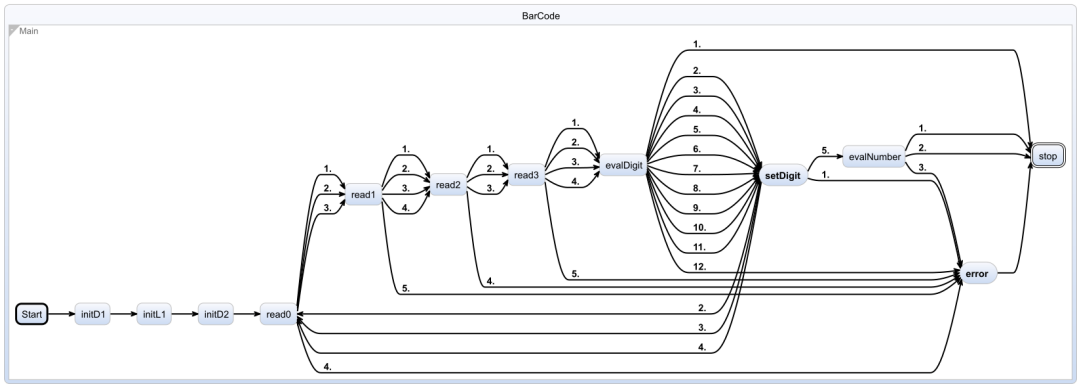


Figure B.12. Graph 8 DF-CB

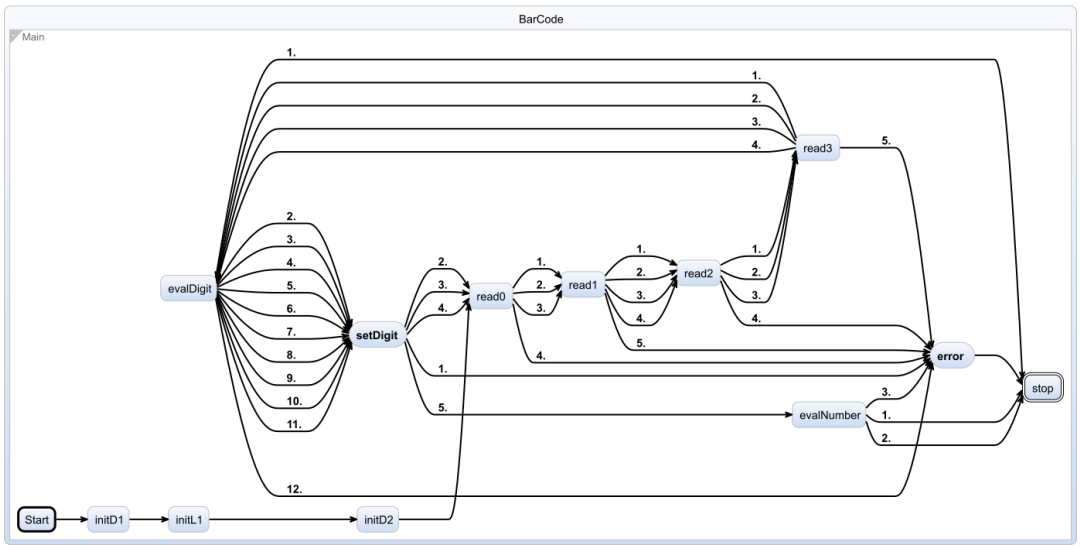


Figure B.13. Graph 8 G-CB

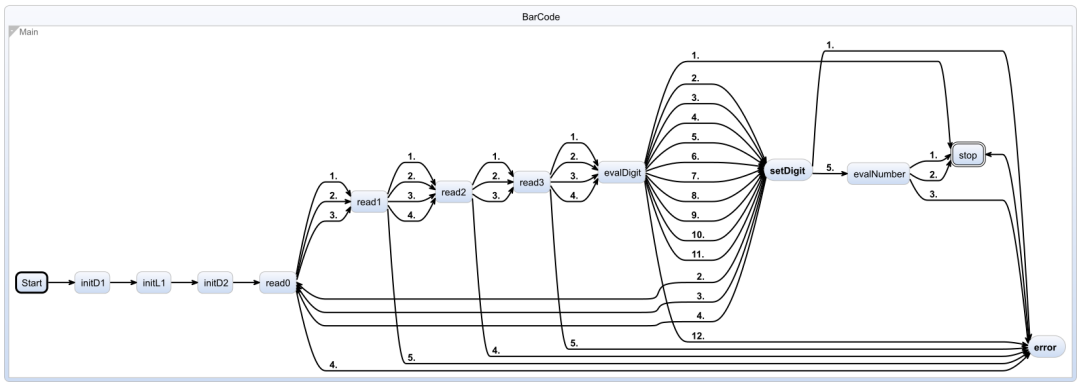


Figure B.14. Graph 8 MO-CB

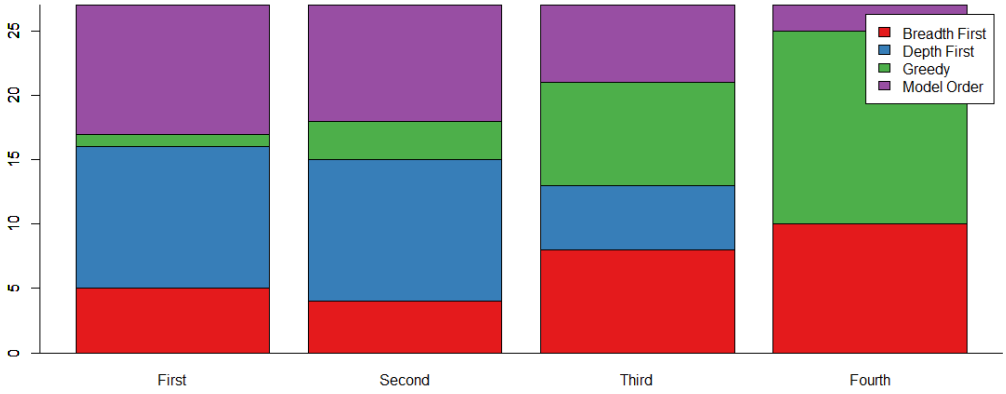


Figure B.15. Graph 8 Impression Ranking

Bibliography

[BK02] Ulrik Brandes and Boris Köpf. "Fast and simple horizontal coordinate assignment". In: *Proceedings of the 9th International Symposium on Graph Drawing (GD '01)*. Ed. by Petra Mutzel, Michael Jünger, and Sebastian Leipert. Vol. 2265. LNCS. Springer, 2002, pp. 33–36. ISBN: 978-3-540-43309-5. DOI: 10.1007/3-540-45848-4.

[Car12] John Julian Carstens. "Node and label placement in a layered layout algorithm". https://rtsys.informatik.uni-kiel.de/~biblio/downloads/theses/jjc-mt.pdf. Master's thesis. Kiel University, Department of Computer Science, Sept. 2012.

[CG72] Edward G. Coffman. and Ronald L. Graham. "Optimal scheduling for two-processor systems". In: *Acta Informatica* 1.3 (1972), pp. 200–213. ISSN: 0001-5903. DOI: 10.1007/BF00288685.

[CTY07] Pierre Charbit, Stéphan Thomassé, and Anders Yeo. "The minimum feedback arc set problem is np-hard for tournaments". In: *Combinatorics, Probability & Computing* 16.1 (2007), pp. 1–4. DOI: 10.1017/S0963548306007887.

[DH21] Sören Domrös and Reinhard von Hanxleden. *Preserving order during crossing minimization in sugiyama layouts*. Technical Report 2103. ISSN 2192-6247. Christian-Albrechts-Universität zu Kiel, Department of Computer Science, Nov. 2021.

[Döh10] Philipp Döhring. "Algorithmen zur layerzuweisung". Bachelor thesis. Kiel University, Department of Computer Science, Sept. 2010.

[DWC+13] Werner Dubitzky, Olaf Wolkenhauer, Kwang-Hyun Cho, and Hiroki Yokota, eds. *Encyclopedia of systems biology*. New York, NY and s.l.: Springer New York, 2013. ISBN: 978-1-4419-9862-0. DOI: 10.1007/978-1-4419-9863-7. URL: http://dx.doi.org/10.1007/978-1-4419-9863-7.

[ELS93a] Peter Eades, Xuemin Lin, and W. F. Smyth. "A fast and effective heuristic for the feedback arc set problem". In: *Information Processing Letters* 47.6 (1993), pp. 319–323. ISSN: 00200190. DOI: 10.1016/0020-0190(93)90079-o.

[ELS93b] Peter Eades, Xuemin Lin, and W. F. Smyth. "A fast and effective heuristic for the feedback arc set problem". In: *Information Processing Letters* 47.6 (1993), pp. 319–323. ISSN: 0020-0190. DOI: 10.1016/0020-0190(93)90079-O.

[FH10] Hauke Fuhrmann and Reinhard von Hanxleden. "On the pragmatics of model-based design". In: *Proceedings of the 15th Monterey Workshop 2008 on the Foundations of Computer Software. Future Trends and Techniques for Development, Revised Selected Papers*. Vol. 6028. LNCS. Budapest, Hungary: Springer, 2010, pp. 116–140. DOI: 10.1007/978-3-642-12566-9.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and intractibility: a guide to the theory of NP-completeness*. New York: W. H. Freeman & Co, 1979.

[HDM+14a] Reinhard von Hanxleden, Björn Duderstadt, Christian Motika, Steven Smyth, Michael Mendler, Joaquín Aguado, Stephen Mercer, and Owen O'Brien. "SCCharts: Sequentially Constructive Statecharts for safety-critical applications". In: *Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '14)*. Edinburgh, UK: ACM, June 2014, pp. 372–383.

[HDM+14b] Reinhard von Hanxleden, Björn Duderstadt, Christian Motika, Steven Smyth, Michael Mendler, Joaquín Aguado, Stephen Mercer, and Owen O'Brien. "Scccharts: sequentially constructive statecharts for safety-critical applications". In: *ACM SIGPLAN Notices* 49.6 (2014), pp. 372–383. ISSN: 0362-1340. DOI: 10.1145/2666356.2594310.

[HFS11] Reinhard von Hanxleden, Hauke Fuhrmann, and Miro Spönemann. "KIELER—The KIEL Integrated Environment for Layout Eclipse Rich Client". In: *Proceedings of the Design, Automation and Test in Europe University Booth (DATE '11)*. Grenoble, France, Mar. 2011.

[KW52] William H. Kruskal and W. Allen Wallis. "Use of ranks in one-criterion variance analysis". In: *Journal of the American Statistical Association* 47.260 (1952), pp. 583–621. ISSN: 0162-1459. DOI: 10.1080/01621459.1952.10483441.

[Pet95] Marian Petre. "Why looking isn't always seeing: Readership skills and graphical programming". In: *Communications of the ACM* 38.6 (June 1995), pp. 33–44.

[Pur02] Helen C. Purchase. "Metrics for graph drawing aesthetics". In: *Journal of Visual Languages and Computing* 13.5 (2002), pp. 501–516.

[Pur97] Helen C. Purchase. "Which aesthetic has the greatest effect on human understanding?" In: *Proceedings of the 5th International Symposium on Graph Drawing (GD '97)*. Vol. 1353. LNCS. Springer, 1997, pp. 248–261.

[Rie10] Martin Rieß. "A graph editor for algorithm engineering". Bachelor Thesis. Kiel University, Department of Computer Science, Sept. 2010.

[Sch11] Christoph Daniel Schulze. "Optimizing automatic layout for data flow diagrams". Diploma Thesis. Kiel University, Department of Computer Science, July 2011.

[SFH09] Miro Spönemann, Hauke Fuhrmann, and Reinhard von Hanxleden. *Automatic layout of data flow diagrams in KIELER and Ptolemy II*. Technical Report 0914. Christian-Albrechts-Universität zu Kiel, Department of Computer Science, July 2009.

[SSH14] Christoph Daniel Schulze, Miro Spönemann, and Reinhard von Hanxleden. "Drawing layered graphs with port constraints". In: *Journal of Visual Languages and Computing, Special Issue on Diagram Aesthetics and Layout* 25.2 (2014), pp. 89–106. ISSN: 1045-926X. DOI: 10.1016/j.jvlc.2013.11.005.

[STT81] Kozo Sugiyama, Shojiro Tagawa, and Mitsuhiko Toda. "Methods for visual understanding of hierarchical system structures". In: *IEEE Transactions on Systems, Man and Cybernetics* 11.2 (Feb. 1981), pp. 109–125.

[Wil45] Frank Wilcoxon. "Individual comparisons by ranking methods". In: *Biometrics Bulletin* 1.6 (1945), p. 80. ISSN: 00994987. DOI: 10.2307/3001968.