INSTITUT FÜR INFORMATIK

Edge Label Placement in Layered Graph Drawing

Christoph Daniel Schulze, Nis Boerge Wechselberg,
and Reinhard von Hanxleden

Bericht Nr. 1802
February 2018
ISSN 2192-6247

CHRISTIAN-ALBRECHTS-UNIVERSITÄT

ZU KIEL

Institut für Informatik der
Christian-Albrechts-Universität zu Kiel
Olshausenstr. 40
D – 24098 Kiel

Edge Label Placement in Layered Graph Drawing

Christoph Daniel Schulze, Nis Boerge Wechselberg,
and Reinhard von Hanxleden

Bericht Nr. 1802
February 2018
ISSN 2192-6247

e-mail: E-mail: {cds,nbw,rvh}@informatik.uni-kiel.de

An abridged version of this work is published at the
10th International Conference on the Theory and Application of Diagrams,
Edinburgh, Scotland, June 2018.

This work has been supported in part by the German Science Foundation,
as part of the Compact Graph Drawing with Port Constraints project
(ComDraPor,DFG HA 4407/8-1).

Abstract

Many visual languages based on node-link diagrams use edge labels. We describe different strategies of placing edge labels in the context of the layered approach to graph drawing and investigate ways of encoding edge direction in labels.

We evaluate the label placement strategies based on both common aesthetic criteria and a controlled experiment. We find that placing labels on their edge can lead to more compact diagrams. Also, placing labels with additional arrows indicating edge direction can help users navigate in large diagrams and is generally preferred by participants of our experiment, outperforming other ways of indicating edge direction.

1 Introduction

Visual programming languages based on *node-link diagrams*, such as Sequentially Constructive Charts (SCCharts) [21] (a synchronous state charts dialect, see Figure 1.1), have become mainstream in several industries. Languages such as *ASCET* (ETAS Group) or *Simulink* (MathWorks) are popular choices in the automotive industry, and even 3D designers without any background in programming have been using visual languages for quite some time now. Autodesk's *3ds Max*, for instance, allows its users to manipulate geometry through its *Max Creation Graph* language. Many of these languages share a number of similarities: first, being based on a notion of either data flow (data is produced, processed, and consumed by *nodes* and transmitted between them through *links* or *edges*) or control flow (nodes represent *states* that can be active or not, with *transitions* transferring control between them); second, deriving some of their semantics through textual labels, be it node labels that specify the types of nodes or edge labels that define when control transfers from one node to the next; and third, requiring users to spend a considerable amount of time on laying out their diagrams [12] for them to properly readable [14], giving rise to automatic layout algorithms [20].

A popular layout approach for flow-based diagrams is the layered approach introduced by Sugiyama et al. [19], which tends to emphasize data or control flow by making the majority of edges point in the same direction. The original description of the layered approach did not mention edge labels. Not taking them into account, however, will lead to layouts with too little space available for their placement, resulting in overlaps with other diagram elements—something that may well cause users to refrain from using automatic layout in the first place. This paper is about making labels first-class citizens during automatic layout.

Contributions. We show different ways of placing labels within the layered approach, including the selection of layers to place labels in and the side of their edge to place them on. We also investigate ways of encoding an edge's direction through label placement or additional decorations, intended to be of particular help in use cases where only parts of a diagram can be displayed on screen. We evaluate the introduced techniques both with an evaluation based purely on aesthetic criteria as well as with a controlled experiment.

Related Work. Label placement in general has a long history in cartography. In a classic paper [8], Imhof lays down six principles for good map labeling, which Kakoulis and Tollis [10] apply to edge labeling as the following three rules:

1. No overlaps between labels and other diagram elements.

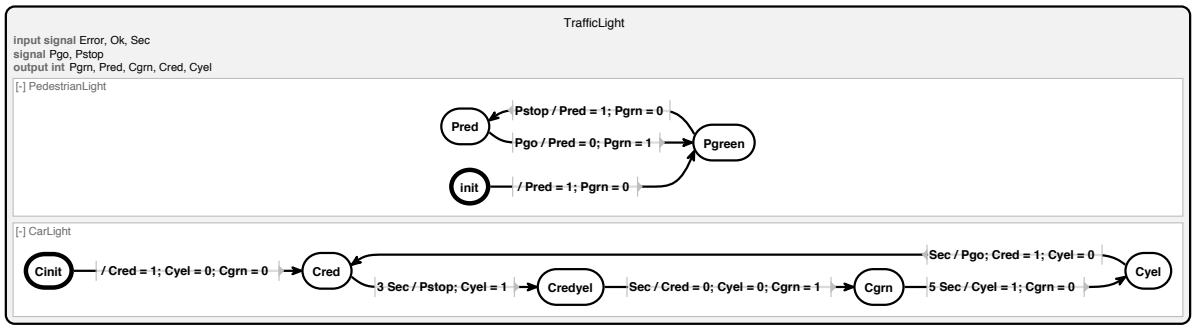


Figure 1.1: An SCChart laid out with the methods we propose in this paper. This drawing uses a horizontal layout with edges routed as splines.

2. It should be clear which diagram element a label belongs to. Imhof calls this "clear graphic association".

3. Among all acceptable positions, a label should be placed in the best possible.

Kakoulis and Tollis also provide a definition of the *edge label placement problem*, which is about placing edge labels in diagrams whose elements have already been placed. Existing algorithms, of which Kakoulis and Tollis provide an overview [20, Chapter 15], usually either run the risk of violating rules 1 or 2 or may resort to hiding or at least scaling down labels to avoid violations—both undesirable for visual programming languages.

In this paper we consider label placement a part of automatic layout, thereby ensuring that there will always be enough space available to satisfy rules 1 and 2. There are algorithms that follow the same approach. Klau and Mutzel [11] for instance integrate node label placement into the topology-shape-metrics approach to graph drawing, although the results they show do not always seem to satisfy rule 2. The *Graphviz dot*[1] algorithm, an implementation of the layered approach, handles edge labels by introducing dummy nodes [4], an approach we follow as well. However, they do not describe any strategies regarding where edge labels end up with regard to their edge. Castelló et al. [1] place labels on edges, which is also one of our label placement strategies. However, they do not discuss graphical design considerations and do not evaluate whether doing so may have a negative impact on the ability of users to read the resulting drawing.

There have been more radical proposals, most notably by Wong et al. [23] who replace an edge by its label. That approach would not work with long edges or orthogonal edge routing, but our on-edge label placement strategy to be introduced in chapter 4 can be seen as a less extreme version of this technique.

There have been investigations into how edge direction can be communicated. Xu et al. [24] found that indicating edge direction through curvature is inferior to using straight arrows. Holten and van Wijk [7] additionally investigate methods such as changing edge thickness or color from tail to head. While methods like these can be successful in small graph drawings, our use cases include diagrams that are too large to fit on a single screen. Since edges can grow rather long in such diagrams, it would be hard for users

[1] `http://www.graphviz.org/`

to spot continuous changes in color or thickness since they are distributed over a larger area. In a subsequent paper, Holten et al. [6] also investigate animating edges to indicate direction and rendering them as a sequence of arrows. While both can be valid solutions, they have two drawbacks. First, they arguably increase visual clutter more than our methods do. And second, they require the rendering of edges to be changed, which may not always be possible. For example, *LabVIEW* (National Instruments) distinguishes different types of edges through their rendering.

Outline. We start with a closer look at the layered approach in chapter 2 before describing ways to choose which layer and which side of an edge a label ends up in in chapter 3 and chapter 4, respectively, before introducing directional decorators in chapter 5. We evaluate the techniques in chapters 6 and 7 and conclude in chapter 8.

2 The Layered Approach

The *layered approach* to graph drawing was first introduced by Sugiyama, Tagawa, and Toda in 1981 [19]. Given an acyclic directed graph, its goal is to produce drawings in which all of the graph's edges point in the same direction. In the authors' terms, this is achieved by establishing what they call a *hierarchy* between the nodes, partitioning them into distinct *levels*, or *layers*, such that edges always point from lower to higher levels in the hierarchy (see Figure 2.1). Note that this usage of the term "hierarchy" differs from how we use it in this thesis, namely to refer to the concept of establishing parent-child relationships between nodes. We thus refrain from calling the layered approach "hierarchical layout", otherwise used as a synonym in graph drawing literature.

The layered approach consists of five phases, of which the first and last were somewhat neglected in the original paper:

1. Cycle breaking
2. Layer assignment
3. Crossing minimization
4. Node placement
5. Edge routing

We will look briefly at each phase in the following section before turning to an important addition to the approach as implemented in the ELK Layered algorithm that ships with the Eclipse Layout Kernel (ELK) project [17]. For a more comprehensive introduction to the topic, we refer to Healey and Nikolov [20, chapter 13].

2.1 Phases of the Layered Approach

Phase 1: Cycle Breaking. It is trivial to see that in order to produce a drawing whose edges all point in the same direction, the input graph must be acyclic. The first phase is thus in charge of turning a graph with cycles into an acyclic graph for the duration of the algorithm by reversing the direction of some of its edges. In the final drawing, these edges will end up as feedback edges, opposing the prevalent layout direction. Formally, the main problem to be solved during cycle breaking is to find a *feedback arc set*, to adopt the terminology used by Eades et al. [2].

Healy and Nokolov note that the original direction of reversed edges can be restored after the second phase [20, chapter 13]. Only that phase solves a problem that strictly

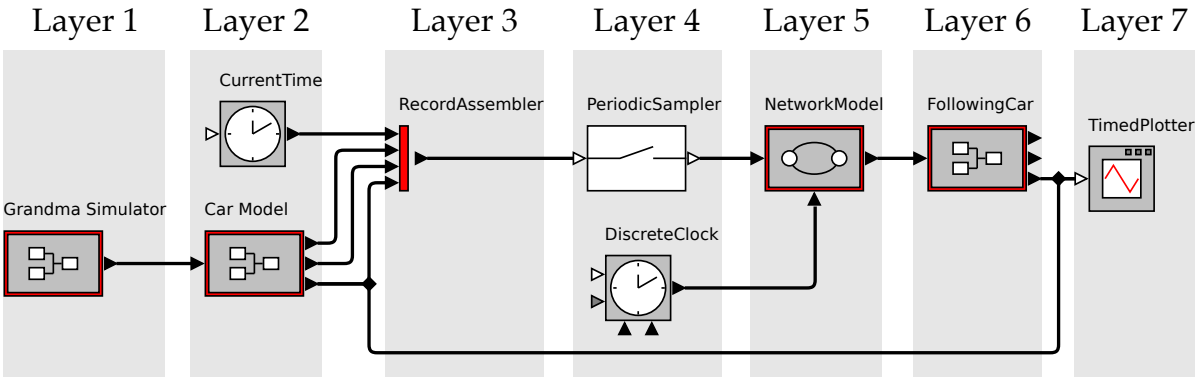


Figure 2.1: A diagram as laid out with ELK Layered, a layer-based algorithm. The different layers are highlighted.

requires the graph to be acyclic. While this is certainly true, the implementation of subsequent phases can be simplified if they, too, can operate under the premise of an acyclic graph.

Phase 2: Layer Assignment. The layered approach would hardly deserve its name if it did not compute a *layering* at some point. Doing so is the responsibility of its second phase. A layering is a partition of a graph's nodes into distinct *layers* $L_1, \ldots, L_n$ such that for all edges that run from a node in layer L_i to a node in layer L_j it holds that $i < j$. A *proper layering* requires $i = j - 1$. We can turn a layering into a proper layering by inserting *dummy nodes* to break long edges and thereby turn them into a sequence of short ones. The dummy nodes are removed again once the algorithm has finished, thereby restoring the original long edges.

The layers can be thought of as columns, placed from left to right in the final drawing according to their position in the layering, while the nodes in each layer are placed below one another.

Phase 3: Crossing Minimization. With nodes assigned to layers, the third phase computes the order of nodes inside each layer in an attempt to reduce edge crossings in the final drawing. Finding a node order that minimizes the amount of crossings is NP-complete [5], and remains so even if the graph consists of only two layers and the order of one of them is fixed [3]. While there are methods to compute optimal solutions, for example by Jünger et al. [9], people usually resort to heuristics.

Phase 4: Node Placement. The fourth step of the layered approach assigns y coordinates to all nodes. As a side-effect, this also determines the height of the final drawing. There are different goals one can strive for in the process.

One such goal is to minimize the height of the drawing. A trivial method which does so places all nodes inside each layer as close together as spacing constraints allow, and then centers these groups of nodes vertically. While this method is interesting in that

it defines a lower bound on the drawing's height, it does not fare well in the legibility department—something users usually do care about.

Other goals include producing a *balanced* placement (centering nodes with respect to their neighbors in other layers), minimizing the length of edges, and straightening as many edges as possible. Node placement methods differ not only in performance or in whether they solve the problem approximately or optimally, but also in the goals they try to achieve in the first place.

Phase 5: Edge Routing. The final phase of the layered approach routes edges between each pair of consecutive layers. The routing determines the amount of space that needs to be left between the layers. A byproduct of this phase thus is that it computes the x coordinates of all nodes and thus the width of the whole graph.

There are different styles of edge routing. *Orthogonal edge routing* inserts bend points such that edges are routed as a sequence alternating between horizontal and vertical edge segments. *Spline edge routing* abolishes sharp bend points by smoothly routing edges through the diagram as splines.

2.2 Ports

At its inception the layered approach did not have any concept of restricting where edges could connect to nodes—the important thing was that they did connect. This is not enough, however, to lay out data flow diagrams such as the one shown in Figure 2.1 which assign semantics to where an edge connects to a node.

Over time, ports crept up in some of the research. Gansner et al. [4] have ports with fixed positions built into their node placement algorithm. The port side cannot be influenced, however: incoming and outgoing edges will always connect to the node's left and right side, respectively. Sander alleviated this restriction by proposing different levels of *port constraints* that define where ports can be placed [15]. Spönemann et al. [18] added yet more levels of constraints to meet the requirements of data flow diagrams.

For the remainder of this report, it is enough to assume that ports have a fixed position.

3 Layer Selection

The aim of integrating edge label placement into the layout algorithm is to reserve enough space for the edge labels to be placed without overlaps and with clear graphic association. Similar to Graphviz dot, we break each edge that has labels by introducing a *label dummy node* to represent them. We compute the size of the dummy node such that all edge labels fit into it, stacked upon each other with a configurable amount of space between them, plus spacing to be left between the labels and their edge. Once edge routing has finished, label dummies are replaced by the labels they represent.

Label dummies need to be inserted before the layer assignment step to ensure that each dummy is assigned to a layer (which might end up existing only because of the label dummy). It does place a burden on the layer assignment algorithm, though: if there is any qualitative difference in where exactly along an edge a label dummy is placed, it seems to be the layer assignment algorithm's responsibility to place it in the optimal spot. Since layer assignment is hard enough as it is, we make no assumptions as to which layer it assigns a label dummy node to (apart from that it will appear somewhere between the layers of its edge's end points, of course). Instead, we can move each label dummy to a layer of our choice after the layer assignment step, if necessary.

That choice is obvious if the edge is so short that there is only one layer to choose from. If the edge is longer, however—such as the edge from `cyel` to `cred` in Figure 1.1—we need a strategy that defines what constitutes the best choice. We define two basic groups strategies can belong to: *simple strategies* and *size-aware strategies*. Simple strategies base their decisions only on structural information, while size-aware strategies also take information about node sizes into account. For the remainder of this chapter, we will look at examples of each group in turn, all of which are illustrated in Figure 3.1.

3.1 Simple Strategies

Possibly the simplest strategy one could imagine (apart from not doing anything at all, of course) is the *median strategy*, which places a label dummy in the median of all the layers its edge spans. If the width of these layers is approximately uniform, the median strategy will end up placing labels near the center of their edge. It can be argued, however, that this may not be the optimal place for all visual languages.

Let us turn to SCCharts to investigate this claim. As mentioned before, edges represent transitions from a source to a target state that are eligible to be taken based on some condition. That condition is part of the edge label, as is a hint regarding the order in which the conditions are tested if multiple transitions leave a state. If edge labels are placed near the edge center, a user might have to search a large area of an SCChart to

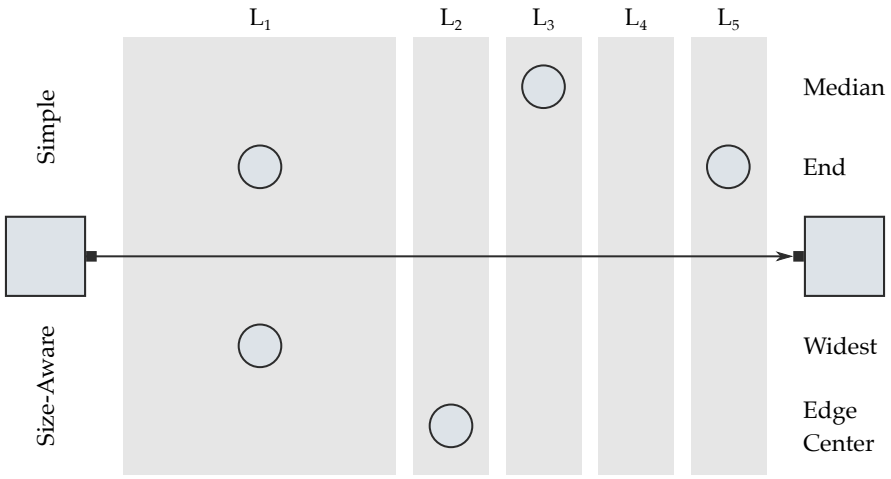


Figure 3.1: Different layer selection strategies would place a label dummy node in different layers spanned by a long edge, in this case layers L_1 through L_5 (with L_2 located at the edge's physical center). What should be considered the best result depends on the design and the requirements of the visual language.

assemble the labels of all outgoing transitions in order to understand how the SCChart works.

A second strategy, the *end layer strategy*, targets exactly this kind of situation. It places a label dummy in the layer closest to either the source or the target node of an edge.

Note that both of these simple strategies may end up placing a very wide label dummy in a layer that contains only comparatively narrow nodes, thereby enlarging that layer even if another one already provided enough space.

3.2 Size-Aware Strategies

Such considerations lead us to strategies that consider the size of nodes and label dummies. First up is the *widest layer strategy*, which places a label dummy in the widest of all layers that its edge spans. For the purposes of this strategy, the width of a layer is defined as the width of its widest non-dummy node. Label dummy nodes in particular do not contribute to a layer's width here since it is not clear whether they are already in their final layer—if they are not, moving them to another layer may change layer sizes and thereby invalidate decisions made earlier. Among all strategies, the widest layer strategy ideally leads to the least amount of wasted white space. To that goal though it sacrifices any obvious rule regarding where along an edge to start looking for its label, which may or may not be acceptable.

The latter point may be an advantage of the simple strategies, but a closer look reveals that the median strategy may in fact suffer from a similar problem. Recall that the first layer in Figure 3.1 is a lot wider than the others. While it is true that the median strategy places the label dummy in the median layer, in this case that layer is not the

one closest to the edge's center. This problem is due to the fact that the median strategy does not take layer sizes into account. The *edge center strategy* tries to improve upon the median strategy by doing exactly that: place the label dummy in the layer likely to be closest to the edge's center in the final drawing. For this to work one needs a way of determining the length of the edge. To do so, we simply sum up the width of all layers it spans. This can of course only be a heuristic, because of two reasons. First, switching label dummy nodes around may influence the size of layers, in turn influencing both the length of the edge and the layer closest to its center. And second, the edge length thusly calculated does not take the amount of space into account that is required for edge routing between layers since that only becomes known during edge routing. This is not problematic if that space is approximately uniform, but can adversely affect the strategy's decisions otherwise.

4 Label Side Selection

The subject of our discussions in the previous chapter was where to place center edge labels along the horizontal axis. In this chapter, we will be concerned with placement along the vertical axis. An edge label can be placed above, below, or even on the edge it belongs to. Implementation-wise, this decision influences where we place the ports of the label's dummy node, as Figure 4.1 shows—but how do we make this decision? Let us work our way through different strategies and examine their advantages and disadvantages.

4.1 Same-Side Strategy

An obvious strategy is the *same-side strategy*, which places all labels either above or below their edge. The simplest strategy to implement, it may also be the easiest for users to understand due to its consistency, which can make it work even when other layout properties do their best to sabotage it.

Consider the example in Figure 4.2a. It is clear which edge each label belongs to due to the fact that it is placed nearer to its edge than to any other edge, in accordance with the Gestalt principle of *perceptual grouping* [22]. Donald Norman would call this "knowledge in the world" [13] in that the diagram can stand on its own and does not require further information to be deciphered (note that this is not a statement about how easy this kind of placement is for users, only about whether it is possible for them to use it at all). Now consider Figure 4.2b. Here, the associations between labels and edges are ambiguous due to unfortunate spacings. Knowing that labels are always placed below their edge resolves any ambiguity and makes the same-side strategy work even in unfortunate circumstances. Norman, however, claims that such additional information required to understand the world—what he calls "knowledge in the head"—should be avoided when possible. Although the same-side strategy works even with unfortunate spacings, the preferred way is to support it with properly chosen spacings to make graphic association as clear as possible. Still, a convention regarding the side labels are always placed on may be part of a visual language.

It might strike one as odd that only some of the edges in the example are labeled. However, this is a common situation since edges usually span different sets of layers and thus have their labels placed in different ones. Note that the different layer spans imply that this situations would occur even with a layer selection strategy that tried to prevent them.

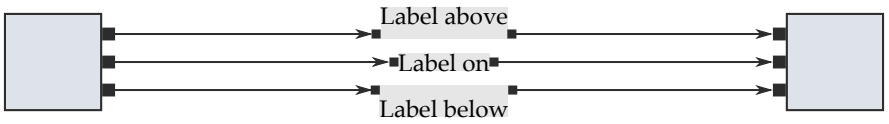
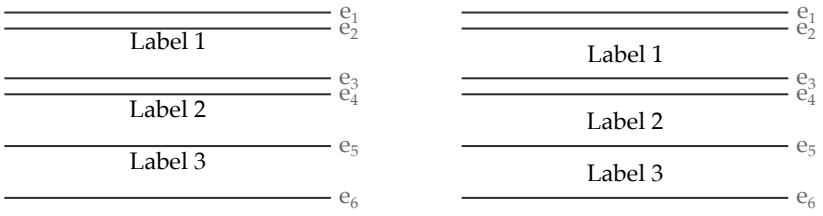


Figure 4.1: Whether an edge label will end up above, below, or on its edge is a function of where its dummy node connects to the edge. Note that the size of the dummy nodes includes the space to be left between the label and its edge, except in the on-edge case.



a: With clear spacings.

b: With ambiguous spacings.

Figure 4.2: Placing all labels above or below their edge yields maximum consistency. **(a)** With proper spacings between labels and related or unrelated edges, a label placement is unambiguous regardless of the label side selection strategy. **(b)** With unfortunate spacings, the same-side strategy still yields unambiguous results if users are aware of the used strategy.

4.2 Directional Strategy

While the same-side strategy works well in terms of clear graphic association, it does not encode additional information, such as the direction an edge is heading towards. Since a label may be far removed from the end points of its edge, any clue as to the edge direction may help a user navigate the diagram. The *directional strategy* aims to do just that by always placing a label to the left or to the right of an edge (looking towards its head).

Figure 4.3 shows an example of this strategy in action. Knowing that labels are always placed to the left of an edge lets us deduce that e_2 is headed rightwards while e_4 and $e5$ are going off to the left. If spacings are chosen well, this additional piece of knowledge is not required for clear graphic association, but offers additional information to advanced users of a visual language who know about the convention.

If spacings are chosen badly, the directional strategy ceases to work. Referring back to Figure 4.2b, it would for example not be clear whether "Label 1" belongs to e_2 (which would then be headed leftwards) or to e_3 (which would then go to the right).

Of course, this strategy requires knowledge in the head to be able to derive edge directions, which we will improve upon in chapter 5.

Figure 4.3: Directional label side selection lets users deduce where a labeled edge is heading without having to look for its end points.

4.3 Augmented Same-Side Strategy

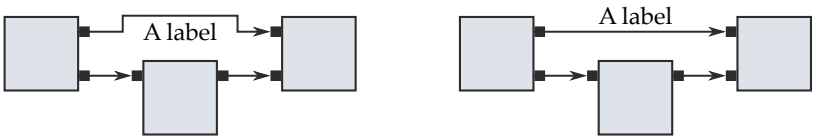
Arguably one of the things most frustrating to users is hitting the layout button and being served a result with deficiencies that they immediately see how to resolve. Schelten calls graphs that contain such abominations Obviously Non-Optimal (O-No) [16], an abbreviation which seems particularly appropriate. Here, we use the term not for complete graphs, but for the deficiencies themselves. While the same-side and directional strategies already seem like decent methods for selecting label sides (particularly with sensible spacings), in practice they can produce O-Nos. Based on observing examples of this, we will derive a set of rules with which to augment the same-side strategy to arrive at what we unimaginatively call the *augmented same-side strategy*. All labels not matched by one of the rules will be assigned a default label side. Our goals are both good graphic association and improving drawings with regard to aesthetic criteria, but when in doubt we will choose to be conservative in that good graphic association will never be sacrificed to improved aesthetics.

Before we start examining examples, however, we should take a look at the general algorithm that implements the augmented same-side strategy. Algorithm 4.1 is executed for every layer in the graph. It iterates over the layer's nodes and looks for consecutive runs of long-edge and label dummy nodes (called `dummyGroup` in the code). Once one such run ends because another type of node is found or because the layer does not have any more nodes, it is processed by calling `process()` in lines 1 and 1. The definition of that function is deviously missing in the pseudo code because it is this function which implements the set of rules which we will be discussing in a moment. The algorithm also does a bit of bookkeeping to keep track of the number of actual label dummy nodes in the current run as well as of whether the current run contains nodes at the top or at the bottom of the layer. These information will be used in our rules, so let us start looking at the examples from which we will derive them.

The first example concerns the length of edges. Consider the example in Figure 4.4a. What we have here is a label of a layer's topmost edge and a label side selection algorithm which thought it a good idea to place the label below the edge. This of course causes the edge to have to make its way around the label, causing its length and in this case also the number of its bend points to increase. Simply placing the label above the edge, as in Figure 4.4b, improves edge routing while perfectly well retaining graphic associations, thus leading to our first rule.

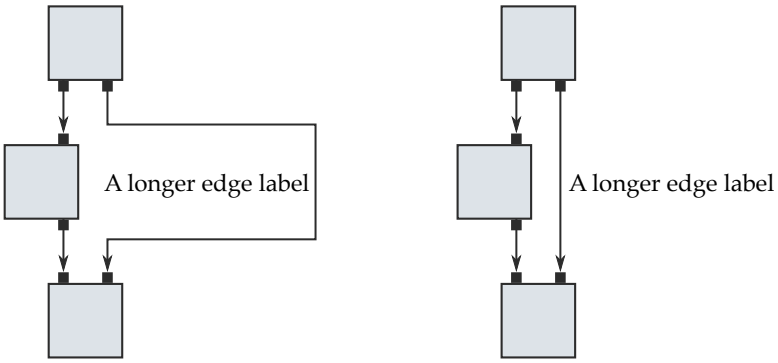
```
Input: layer, a list of nodes in the layer to be processed
Output: Assignment of label side to each label dummy in layer

dummyGroup ← empty list
labelCount ← 0
topInLayer ← true
bottomInLayer ← false

foreach node in layer do
    if isLabelDummy(node) then
        Add node to dummyGroup
        labelCount += 1
    else if isLongEdgeDummy(node) then
        Add node to dummyGroup
    else
        if |dummyGroup| > 0 then
            process(dummyGroup)
            dummyGroup ← empty list
            labelCount ← 0
        end
        topInLayer ← false
    end
end

if |dummyGroup| > 0 then
    bottomInLayer ← true
    process(dummyGroup)
end
```

Algorithm 4.1: The *augmented same-side strategy* is run for every layer of a graph. It basically looks for consecutive runs of long-edge and label dummy nodes and processes them.



a: Without rule 1. b: With rule 1.

Figure 4.4: Simple label side selection methods can easily produce edges that are longer and have more bend points than necessary. **(a)** The label causes its edge to be routed around it. **(b)** Changing the label sides reduces edge length and the number of bend points.



a: Without rule 1. b: With rule 1.

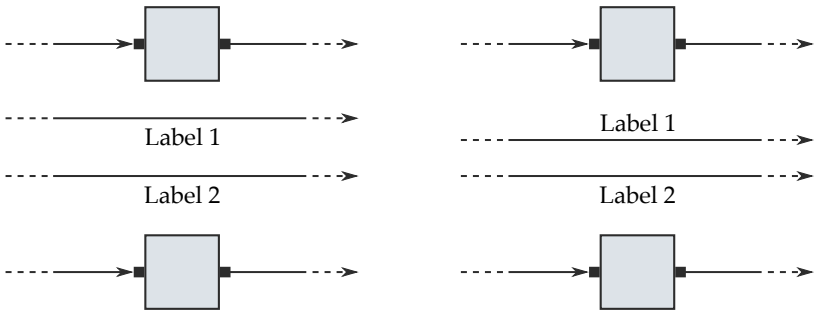
Figure 4.5: A graph similar to the one in Figure 4.4, but with a vertical layout direction. **(a)** Without applying rule 1, the problem becomes even worse since the label's width has more of an impact. **(b)** Applying rule 1 again improves the result.

Rule 1
If `labelCount` = 1 and either `topGroup` is true and the first node in `dummyGroup` is a label dummy or `bottomGroup` is true and the last node in `dummyGroup` is a label dummy, configure the corresponding label to be above or below its edge, respectively.

Note that we are being conservative here: if more than the topmost edge has a label in the layer, we do not apply this rule in order to avoid any confusion on the user's side (although one of the subsequent rules may apply).

The usefulness of rule 1 becomes more apparent for vertical layouts, as in Figure 4.5. Here, the width of the label has much more of an impact on the layout than its height did in the horizontal case.

The second example concerns cases where a run of dummy nodes consists of exactly two nodes, as in Figure 4.6. The same-side strategy will place one of the labels between the edges, while the directional strategy may even end up doing that to both labels. While this may already be clear enough, especially if the label-side selection strategy is known and spacings are chosen sensibly, we can still improve graphic association by placing none of the labels between the edges. To repeat a point made during our



a: Without rule 2. b: With rule 2.

Figure 4.6: Example of a diagram where only two edges run between two regular nodes. **(a)** The same-side strategy will put one of the two labels between the edges. The directional strategy may even end up putting both labels there. **(b)** Placing the labels around the edges improves graphic association.

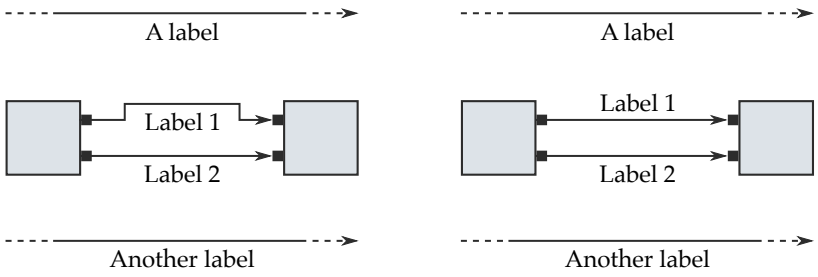
discussion of port labels in **??**, placing the labels around the edges does not introduce even the possibility of mistaking the association of one of the labels. This leads us to our second rule.

> **Rule 2**
> If |`dummyGroup`| = 2, configure the group's first dummy for above and the second for below placement (if it is a label dummy node).

The final example highlights a construct that frequently appears in SCCharts: two states connected by two transitions, one in each direction, as shown in Figure 4.7. We call this a *tight loop*, although being a loop is not really a requirement—one can easily imagine situations in data-flow languages where one actor sends two signals to the next actor instead of just one, effectively producing the same situation. This of course causes the same problems we already discussed for the previous rule, but worse. If two nodes are connected by two edges, those edges will usually be spaced rather tightly if they are not labeled. Introducing a label into the space between them may not only introduce the possibility of ambiguity, but also requires them to be moved apart, introducing bend points and elongating them (Figure 4.7a). Placing the labels around the edges improves the situation (Figure 4.7b), leading us to the final rule.

> **Rule 3**
> When encountering exactly two dummy nodes that belong to edges that connect the same two nodes, configure the upper one's label to be placed above and the lower one's label to be placed below the edge.

This rule seems contradict our basic goal of being conservative in our label side decisions. After all, there may be more label dummy nodes in the current run, which would so far cause us to fall back to the default label side for all of them. However, in our experience such tight loops are often placed at a bit of a distance from surrounding



a: Without rule 3. b: With rule 3.

Figure 4.7: Edges of a "tight loop" tend to be close to one another, which causes longer edges and bend points if labels are placed between them. This is a generalization of what Figure 4.6 was about to tight loops possibly surrounded by other edges.

elements, thus preserving graphic association.

4.4 On-Edge Strategy

We have thus far focused on placing labels next to their edge, which is the standard edge labeling strategy in the vast majority of graphical modeling tools. This makes perfect sense in that the principles of legibility and minimization of disturbances seem to call for labels to not overlap their edges. There is a case to be made, however, for placing them *on* their edge.

When placing labels next to their edge, one of our main concerns has to be clear graphic association. The perception of their association improves as they get closer to each other and further away from unrelated elements, an example of the principle of *proximity* in what Gestalt psychology calls *perceptual grouping* [22] (graphic designers will use the same principle to group elements or distinguish them from other elements). Given properly configured spacings, this is not too hard to achieve, but will cause a drawing to grow in size. As spacings shrink, drawings do get smaller, but the possibility for ambiguous graphic association grows.

Wong et al. [23] eliminate the need for perceptual grouping by going so far as to replace the edge itself with its label, gradually changing the font size from tail to head to indicate edge direction. We will not follow their proposal, due to several reasons. First, for the approach to work without introducing distortion or very different font sizes, the length of an edge would have to be a function of the text it is labeled with—a prerequisite quite obviously not compatible with the layered approach. Second, we allow edges that share a common end point to be drawn as hyperedges, which essentially lets them share parts of the routes they take through the diagram. This would have a decidedly negative impact on the legibility of labels. And finally, the orthogonal edge routing style (or any routing style that employs bend points, for that matter) would degrade label legibility even further.

On-edge label placement achieves optimal graphic association without completely replacing edges by their label. If the layout direction is horizontal, we may also reduce

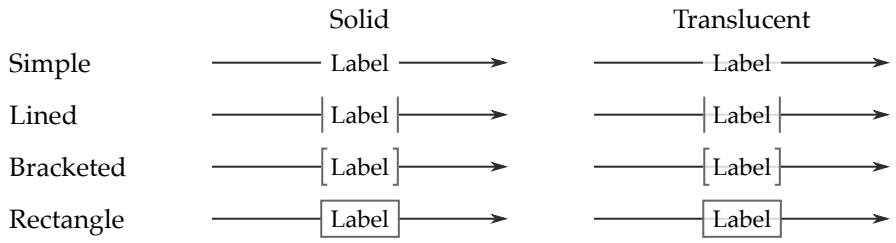


Figure 4.8: Four examples on-edge label designs.

the diagram's height slightly because there is no edge-label spacing anymore, and since each label sits on its edge the space between the label and unrelated edges can be much smaller than it could otherwise.

For on-edge label placement to work, the graphical representation of edge labels has to be designed accordingly. Labels must have either a solid background or at least cause the background to be sufficiently faded for the edge not to interfere with the text's legibility. This requirement is easy to meet, and many designs for on-edge labels are possible, which may even reflect different edge semantics. Figure 4.8 shows four simple examples of on-edge label representations. Castelló et al. [1] use a simple solid design when drawing statecharts, but do not discuss their motivation for doing so. Interrupting the edge, however, may cause users to have a harder time following it through the diagram. We will investigate this in chapter 7.

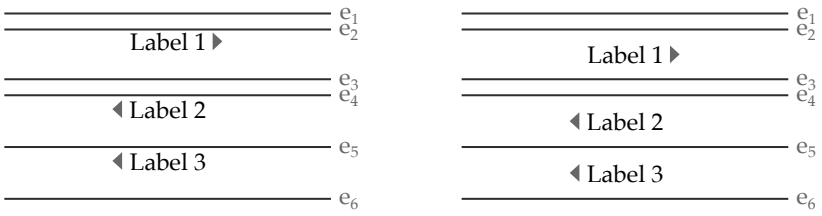
5 Directional Decorators

The directional label side selection strategy had the advantage of encoding information about the direction of an edge, but suffered from both potential graphic association problems as well as knowledge in the head for its proper interpretation. If the label side is unavailable or unwanted as a means to communicate additional information, what remains is the possibility to communicate through the label's design. Figure 5.1 shows examples of on-edge labels decorated with an arrow which points towards the edge's head.

Compared to the directional strategy, this way of indicating edge direction has the advantage of representing knowledge in the world, not in the head, thus not requiring user to know about any label side conventions that would otherwise have to be learned. Such decorations consequently work with any label side selection strategy, thus allowing the same-side strategy to communicate the same amount of information as the directional strategy does while being slightly clearer in terms of graphic association (Figure 5.2).

An interesting problem concerns the implementation of directional decorators. Whether the arrow should point leftwards or rightwards is subject to the diagram's layout, which implies that the viewing framework needs to support changes to the visualization after automatic layout has run. How this can be done depends on the viewing framework and is outside the scope of this paper.

Figure 5.1: Labels can be decorated with arrows to point at where the edge is heading. While this example only shows on-edge labels, such decorations can of course also be added to labels placed next to their edge.



a: With clear spacings.

b: With ambiguous spacings.

Figure 5.2: The same edge label placement as in Figure 4.2, but with directional decorators added to the labels. **(a)** With arrows, the same-side label selection strategy can clearly communicate edge direction. **(b)** With badly chosen spacings, the directional label side selection strategy would cease to work due to ambiguous graphic association. The same-side strategy still works and can communicate the same amount of information when augmented with directional decorators.

6 Aesthetic-Based Evaluation

We have explained how to place edge labels regarding two main problems: which layer to place a label in and which side of its edge to place it on. In this chapter we will compare our solutions by comparing their influence on common aesthetic criteria.

6.1 Layout Impact of Side Selection Strategies

The choice of a side selection strategy will influence different aesthetics of the drawing. The aim of the first evaluation is to get an idea of the extent of that influence. The following questions seem of particular need to be answered:

1. Does the on-edge strategy result in smaller drawings? It requires less edge-edge spacing due to the obvious graphic association between labels and the edges they belong to. This would seem to cause horizontal drawings to vary in height and vertical drawings to vary in width, but the question is by how much. We did not expect the differences to be very large.

2. Does the augmented side selection strategy yield shorter edges? It does attempt to reduce obvious O-Nos where a label's placement causes its edge to take a longer detour, leading us to hypothesize that it would indeed produce shorter edges than its simple counterparts, although again only slightly. However, since labels are wider than they are high the effect would be more pronounced in vertical layouts.

3. Does the augmented side selection strategy increase the number of straight edges? The way the augmented strategies choose label sides leads us to believe that it does, but again the increase must be expected to be small.

To help us in our quest for answers, we obtained 315 SCCharts produced by students. Some were created as part of their homework assignments in courses on synchronous languages and real-time systems that used SCCharts as an example of a graphical synchronous language. Others were derived from a large SCChart that controls an elaborate model railway, produced by students during a recent practical. In all cases the students used a textual language to describe the SCChart which was then rendered as a diagram. Since SCCharts usually consist of several levels of hierarchy that are laid out by different layout algorithms, we extracted simple graphs (only a single level of hierarchy) and removed diagrams that had no edge labels or less than three nodes. We thus ended up with 641 diagrams averaging 6.47 nodes, 9.37 edges, and 8.93 edge labels.

	Relative Height		Relative Width	
	AVG	SD	AVG	SD
Down	0.907	0.091	0.883	0.093
Up	0.897	0.093	0.887	0.093
Dir. Down	0.891	0.099	0.915	0.113
Dir. Up	0.892	0.097	0.880	0.096
Aug. Down	0.894	0.089	0.876	0.096
Aug. Up	0.896	0.090	0.880	0.104

Table 6.1: Height of horizontal drawings and width of vertical drawings (in pixels) obtained using the on-edge side selection strategy expressed as a fraction of the size produced by the other strategies.

Before being analyzed the diagrams were laid out by ELK Layered. Each diagram was laid out twice for each side selection strategy, once with a horizontal layout direction (left-to-right) and once with a vertical layout direction (top-to-bottom). We used the same settings throughout, except for the on-edge strategy where we reduced the edge-edge spacing from 10 to 5 due to clearer graphic association.

Let us start with the first question: the impact of the on-edge side selection strategy on a horizontal drawing's height and a vertical drawing's width. Table 6.1 shows the size of each drawing made using the on-edge strategy expressed in terms of the size produced by the other strategies and is, on average, consistently smaller. A look at a plot of the relative data in Figure 6.1 however reveals that this is not always the case. In fact, the different strategies will yield slightly different node placements, which in turn can lead to bigger diagrams. Due to the fact that labels are wider than they are high this effect is more pronounced in vertical layouts. As Table 6.2 shows, however, the number of cases where this happened in our set of diagrams is very small.

In summary we can conclude that using the on-edge label side selection strategy will yield smaller diagrams than the other strategies in the vast majority of cases.

The second question was concerned with whether the augmented side selection strategies resulted in shorter edges compared to their simple counterparts. Table 6.3 shows the differences in edge length when switching from the always down and always up strategies to the augmented down and augmented up strategies, respectively. While the differences are negligible in horizontal layouts, they are indeed much more pronounced in vertical layouts due to the fact that labels can be rather long, forcing considerable detours onto their edges. However, the standard deviations are rather large as well.

Table 6.4 shows the percentages of drawings where switching to an augmented strategy made edges shorter, left them unchanged, or made them longer (both the average length of edges in a drawing as well as its longest edge). Here the pattern becomes clearer: switching to an augmented strategy improves average edge length in about 60%

a: Horizontal drawings.

b: Vertical drawings.

Figure 6.1: Box plots of the data underlying Table 6.1.

Strategy	Height			Width		
	Smaller	Unchanged	Larger	Smaller	Unchanged	Larger
Down	73.5	25.0	1.6	92.5	2.3	5.1
Up	75.8	23.4	0.8	92.4	2.5	5.1
Dir. Down	74.4	24.8	0.8	80.7	2.3	17.0
Dir. Up	76.3	23.4	0.3	92.4	2.3	5.3
Aug. Down	77.7	21.5	0.8	92.5	2.2	5.3
Aug. Up	76.3	22.8	0.9	92.4	2.3	5.3

Table 6.2: The number of cases where switching from a given side selection strategy to the on-edge strategy led to smaller, unchanged, or larger drawings (regarding height in horizontal and width in vertical layouts). All values are given in percent of all 641 analyzed drawings.

Direction	Strategy	Average Length		Maximum Length	
		AVG	SD	AVG	SD
Horizontal	Down	−4.73	13.47	−14.31	82.64
	Up	−5.06	12.05	−9.51	52.56
Vertical	Down	−75.05	157.74	−221.89	549.19
	Up	−76.30	124.14	−228.32	461.88

Table 6.3: Change in edge lengths (in pixels) when switching from a simple side selection strategy to its augmented counterpart. Negative change means shorter edges in the augmented case. Our data points for the *average length* column are the average length of all edges in a diagram.

Direction	Strategy	Average Length			Maximum Length		
		Shorter	Unchanged	Longer	Shorter	Unchanged	Longer
Horizontal	Down	47.3	37.1	15.6	45.9	43.4	10.8
	Up	54.4	38.4	7.2	41.0	48.5	10.5
Vertical	Down	61.0	31.7	7.3	53.4	35.7	10.9
	Up	59.3	34.9	5.8	53.2	37.6	9.2

Table 6.4: The number of cases where switching from a simple side selection strategy to its corresponding side selection strategy led to shorter, unchanged, or longer average and maximum edge lengths in a drawing. All values are given in percent of all 641 analyzed drawings.

Strategy	Bend Points	
	Horizontal	Vertical
Down	2014	1,900
Up	1967	2,065
Dir. Down	1989	2,031
Dir. Up	2060	2,060
Aug. Down	2055	2,013
Aug. Up	2041	2,076
On Edge	2064	1,989

Table 6.5: The number of bend points generated across all analyzed drawings by label side selection algorithm

of vertical drawings. More importantly, it turns out to be a bad idea only in less than about 10% of our examples.

We thus feel it is safe to say that the augmented strategies will indeed often lead to shorter edges, although the impact is much larger for vertical than for horizontal layouts.

The final question is whether the augmented strategies reduce the number of bend points as well. As it turns out we can answer this question rather easily if we look at the sum of bend points across all drawings by label side selection algorithm, as shown in Table 6.5 The different strategies produce very similar numbers of bend points, including the augmented strategies, which do not even fall at the lower end of the spectrum. Investigating the data further confirms that we must answer this final question with no.

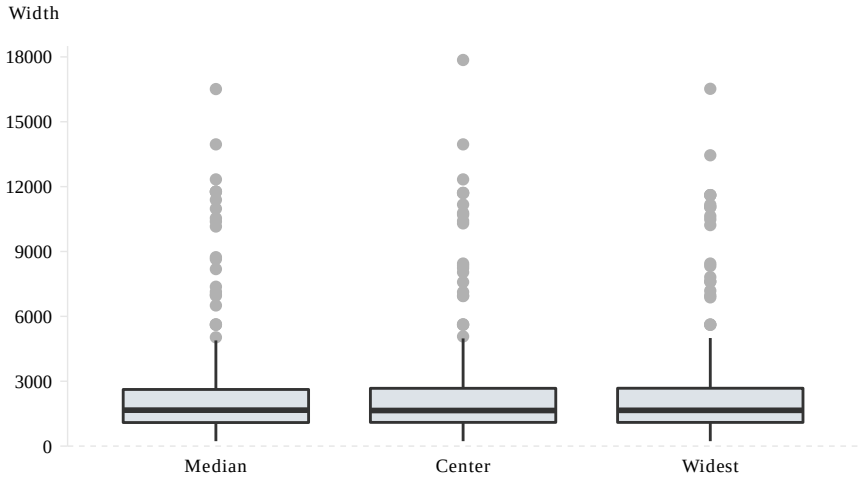


Figure 6.2: The diagram width resulting from the application of different layer selection strategies.

6.2 Layout Impact of Layer Selection Strategies

The impact of choosing a layer selection strategy on the aesthetics of the resulting layouts may not be the first criterion to base that choice on. Picking the head or tail layer strategies, for example, can simply be based on the visual language's usability requirements. SCCharts are an obvious case in point: it is probably more helpful to display transition conditions near the source state than to worry about what that may do to the aspect ratio of the resulting drawing. What the aesthetic impact is more helpful for is to judge the effectiveness of the size-aware layer selection strategies. It is with this in mind that we want to answer the following questions:

1. Does the widest layer strategy succeed at reducing the width of diagrams compared to the other strategies?

2. Which strategy is best at placing edge labels closest to the edge's physical center (excluding the head and tail layer strategies, of course)?

We based this evaluation on the same diagrams as the previous one, but removed diagrams where there was no choice regarding which layer to place labels in since the layer selection strategies would not make a difference there. We thus ended up with 366 diagrams totaling 2758 nodes (with an average of 7.54), 4525 edges (12.36), and 4357 edge labels (11.9).

Figure 6.2 shows the width of diagrams depending on the chosen layer selection strategy. The widest layer strategy does not succeed to reduce the width of diagrams. This is likely due to the fact that the way it estimates the width of layers—not taking label dummies into account—is an insufficient approximation. More research is required to develop a method that chooses layers for all labels at once.

	AVG	SD
Median	0.493	0.085
Center	0.510	0.098
Widest	0.494	0.122

Table 6.6: Relative positions of the center point of labels along the horizontal span of the edge they label. A value of 0.5 would indicate a label being placed at the center of its edge. The performance of each algorithm is described in terms of the Average (AVG) and the Standard Deviation (SD).

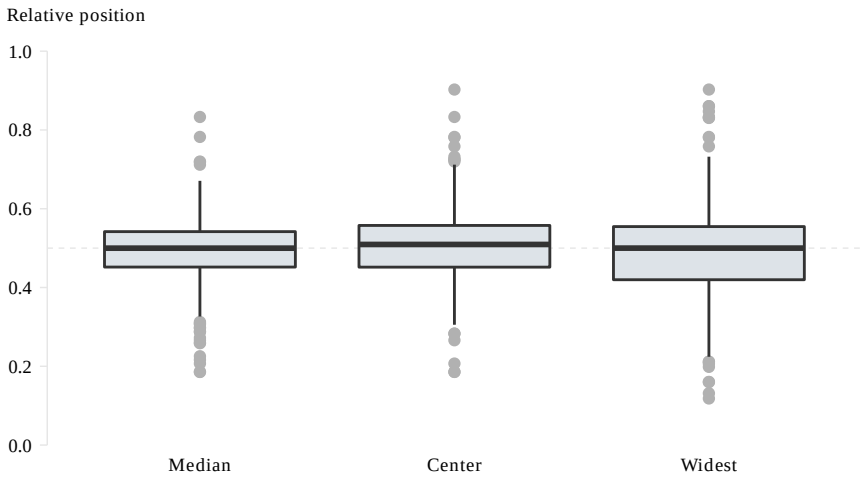


Figure 6.3: Relative positions of the center point of labels along the horizontal span of the edge they label. A value of 0.5 would indicate a label being placed at the center of its edge.

To answer the second question (which layer selection strategy is best at placing labels at the center of their edge), we looked at the horizontal coordinate span of each edge and measured where along that span the center points of its labels ended up. Values of 0.0 and 1.0 would indicate that a label's center point lies on the edge's left and right end points, respectively. Since the head and tail layer selection strategies are of little interest for this question, we performed the measurements for the median, center, and widest layer algorithms.

Table 6.6 shows a summary of the measurements while Figure 6.3 plots the data as a box plot. Interestingly, the median strategy cannot be said to perform worse than the center strategy, which explicitly tries to take layer sizes into account. Again, this must be attributed to the fact that layer sizes change while moving label dummies to other layers.

In summary, it seems necessary to develop more sophisticated size-aware layer selection methods.

7 Experimental Evaluation

We wanted to answer the following research questions:

1. Which strategy is better at indicating edge direction, traditional label placement with direction-dependent label side selection or on-edge label placement with directional decorators?

2. Does on-edge label placement have a negative impact on the ability of users to follow edges through a diagram compared to traditional label placement?

As for the first question, it seems obvious that adding directional decorators makes edge direction more apparent than following direction-dependent label side selection conventions, again referring back to Norman's distinction between information in the world as opposed to information in the head [13]. The question remains, however, how much of a difference there really is.

It is the second question, though, that seems most important: the objective advantages of on-edge label placement (clear graphic association and reduced space) would be of little value if it meant that users were not able to effectively follow edges through their diagrams anymore. We hypothesized that this might be the case due to the visual interruptions caused by on-edge labels as opposed to the solid lines edges are rendered as with traditional label placement.

We designed a controlled experiment that consisted of three parts: one for each question, and a concluding interview where participants were asked to rank label placement strategies according to personal preference. The first two parts were performed using an application developed specifically for that purpose. Since the tasks to be solved during the experiment were not too time-consuming, we used a within-participants design throughout (every participant was subjected to all of the conditions).

The software used to conduct the experiment as well as all of the data necessary to repeat the experiment can be obtained by contacting the authors. The collected data and any scripts used to conduct the subsequent analyses are available online.[1]

We start by describing the experiment's overall procedure and then describe and analyze each part separately.

7.1 Procedure

48 participants between 18 and 33 years (averaging 23,35) were recruited for the experiment, 9 of them female. All participants studied computer science either as a major

[1] `http://rtsys.informatik.uni-kiel.de/~biblio/downloads/papers/report-1802-data.zip`

or as a minor (36 undergraduate students, 11 graduate students, and one PhD student) and were recruited via e-mail to the institute's central student mailing list. None were in any way involved in the research that was the subject of this experiment. Participation was completely voluntary, but we compensated participants by paying them 5 Euros. All experiments were conducted by a single experimenter, always in the same room. Most experiment runs took place between 1pm and 5:30pm, except for two that were conducted at 10am and 11am. A full run would take about 30 minutes.

Participants were sat down in front of a computer screen with a resolution of 1920x1200 pixels. Except for a demographics questionnaire, the experiment was controlled purely through the keyboard.

Participants were first asked to read a document that explained the experiment and the data to be collected, and then signed a declaration of consent. The experimenter then briefly explained the structure of the experiment before they were directed to follow the instructions on the computer screen, which led them through the first two parts of the experiment. A full run through the experiment would take a participant about 30 minutes to complete.

Thanks to five pilot runs, no severe problems arose during the experiments. One session was slightly disturbed by someone knocking and opening the door, but the participant did not feel that this was a significant problem, and we did not find obvious anomalies in their data.

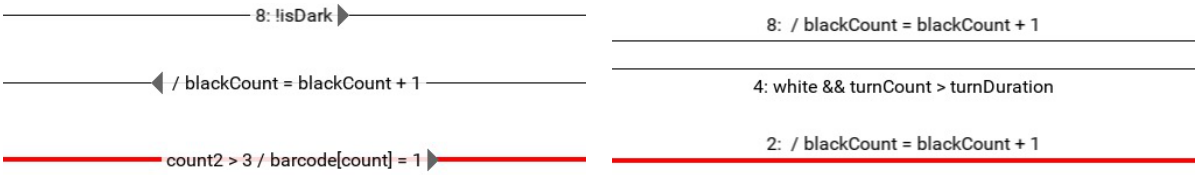
7.2 Effectiveness of Directional Label Placement

The first part of the experiment was meant to help answer the question of whether on-edge label placement with directional decorators (OED) or next-to-edge label placement with direction-dependent label side selection (NED) is more effective at conveying edge direction. These, then, also served as the two conditions. Our hypothesis was that explicit directional decorators would be significantly more effective than implicitly encoding direction through placement side, in terms of both reaction time and error rate.

7.2.1 Experimental Method

The experimental objects were designed to only show parts of edges to simulate a situation where users would have to rely on edge labels to infer where an edge is headed, as shown in Figure 7.1. This was to simulate typical use cases the placement strategies were designed for: focusing on a particular part of the diagram that does not include the end points of an edge, or zooming into the diagram enough that they are not even on screen. Each experimental object was generated randomly, with 5 to 10 edges placed at a distance of 20 to 70 pixels, decorated with labels randomly generated based on labels found in SCCharts. One of the edges was highlighted in red and drawn as a thicker line.

The task was to infer the direction of the highlighted edge and indicate the direction by pressing the left or right control key. The control keys were chosen over the arrow keys



a: Condition 1: OED

b: Condition 2: NED

Figure 7.1: Excerpts of experimental stimuli for the two conditions of the experiment's first part. Participants were asked to infer the direction of the highlighted edge as quickly as possible. The correct answer would be "right" in both examples.

to reduce the probability of accidentally pressing a wrong key due to close proximity. We did not impose a time limit, but instructed participants to solve the task as quickly as possible.

The software first introduced participants to both edge label placement strategies and included five practice trials per condition to practice the task to be laid upon them, with visual feedback indicating whether or not a particular answer was correct. At this time they also had the opportunity to ask the experimenter for clarification on anything they did not understand.

Participants then performed three blocks of 30 trials each and we measured the reaction time and recorded the correctness of each answer. Each block consisted of 15 trials of each condition in randomized order.

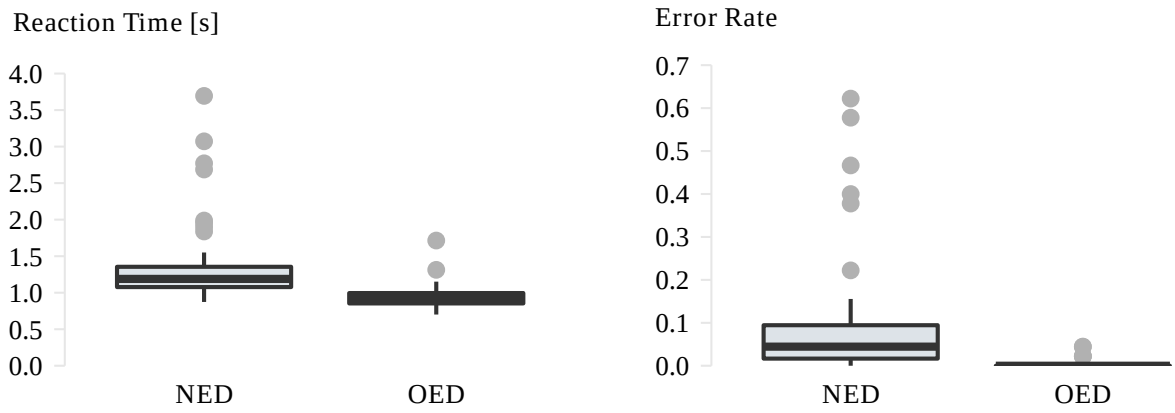
7.2.2 Data and Analysis

Figure 7.2 shows box plots of the mean reaction time and error rate of each participant according to condition. Since neither of the two sets of data could be considered to follow a normal distribution as well as the facts that the experiment used a within-participants design and exactly two conditions, we used a Wilcoxon signed rank test with a typical significance level of 0.05 and found significant differences ($p < 0.01$) between the conditions, with the OED condition taking less time and producing fewer errors than the NED condition.

7.2.3 Conclusions

The results are clearly (and unsurprisingly) in accordance with our hypothesis: the OED condition outperforms the NED condition in both metrics. The error rate in particular also seems to support Norman's preference for information in the world: the error rate was generally higher for NED than it was for OED, and some participants seem to have gotten confused as to which edge direction a given label placement side indicates. As one participant put it, "This completely confused me, I always had to think about how it worked."

Of course, the task forced participants to focus only on edge direction, which in real-world scenarios is only part of what users do to read and work with diagrams. We believe



a: Reaction time

b: Error rate

Figure 7.2: Box plots of the mean reaction times and error rates of each participant according to condition for the first part of the experiment.

it to be likely that focusing on this task may actually have improved results for the NED condition in this experiment since there were no distracting tasks here that may have removed the (arbitrarily chosen) semantics of NED from working memory. Even so, 30 participants explicitly stated that inferring edge directions with the help of NED was confusing, hard to remember, or required more thought in the concluding interview.

7.3 Perceptual Impact of On-Edge Label Placement

The second part was designed to answer the question of whether on-edge label placement has a negative impact on the ability of users to follow edges through a diagram. Our hypothesis was that it might indeed have a negative impact, but hoped that it would be small.

In addition to the two conditions already present in part one, part two added always-down next-to-edge label placement (NE) as a third condition. Since that strategy does not encode directional information, any significant performance difference (or lack thereof) would allow us to draw conclusions about the extent to which participants used those information to solve the task to be described soon.

7.3.1 Experimental Method

This time, the experimental objects were diagrams that fit completely on a single screen. In the interest of realism, we based the diagrams on nine excerpts from SCCharts developed by students as part of their homework during lectures throughout the past semesters, but modified them to fit onto a single screen. The diagrams averaged 16 nodes and 22.8 edges and all but one had edge crossings. We obtained nine additional objects by reversing all edges of the original excerpts and switching node names around,

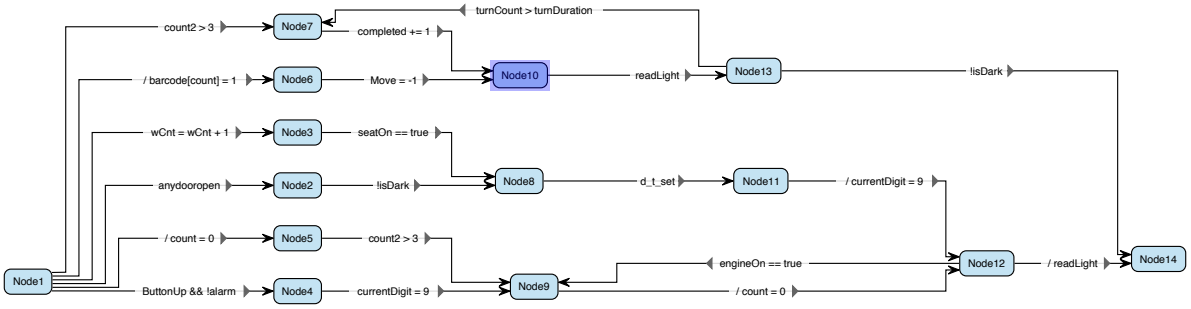


Figure 7.3: Experimental stimulus of the experiment's second part, in this case with on-edge labels with directional decorators. All objects were laid out using a left-to-right layout direction and drawn using orthogonal edges.

resulting in diagrams that looked significantly different.

The task was the fairly typical one of counting how many different nodes could be reached from a highlighted start node by traversing exactly two edges. This required participants to follow edges through the diagram, allowing us to draw conclusions about our research question. Again, we did not impose a time limit, but asked participants to solve the task as quickly as possible. Each of the 18 experimental objects had two possible start nodes defined, yielding a total of 36 different stimuli. Figure 7.3 shows an example of what a typical stimulus would look like.

Each trial began with drawing the experimental object. The start node was highlighted one second later to avoid the need for participants to spend time searching the diagram for it. We measured reaction time (starting with highlighting the start node) and recorded the correctness of each answer.

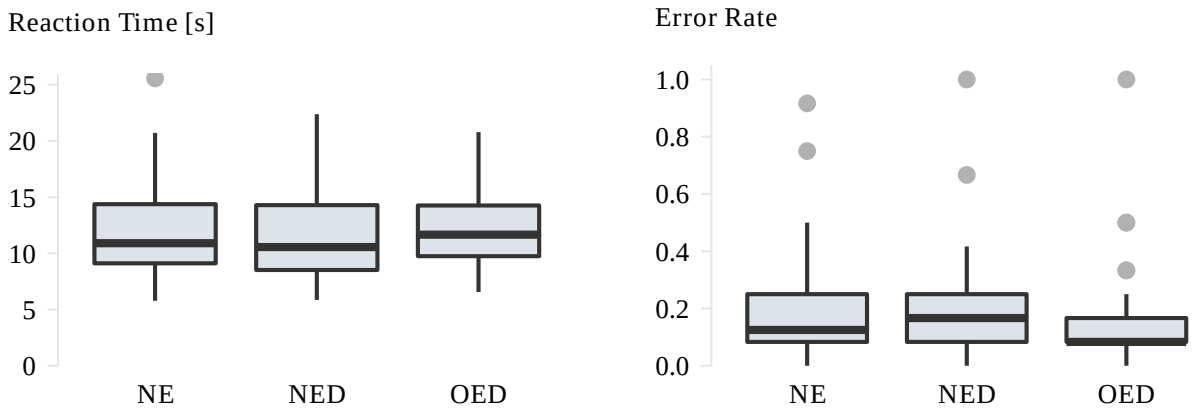
The software first introduced participants to the task that would be their charge and allowed for three practice trials, with visual feedback indicating which nodes are reachable in exactly two steps from the highlighted node. At this time they also had the opportunity to ask the experimenter for clarification on anything they did not understand.

Participants then performed three blocks of 12 trials, one block for each condition. To average out any learning effects, the order of blocks was assigned based on a Latin square and each of the six possible permutations appeared the same number of times across the 48 participants. Each of the 18 experimental objects appeared at most once in the same block to minimize situations where participants recognized a diagram and would be able to react more quickly than they otherwise could.

7.3.2 Data and Analysis

Figure 7.4 shows box plots of the mean reaction time and error rate of each participant according to condition. It is obvious that there are no significant performance differences in either of the two data sets.

A single participant was responsible for the worst error rate in each condition. It is unclear why the error rate was this high since the participant mentioned nothing



a: Reaction time

b: Error rate

Figure 7.4: Box plots of the mean reaction times and error rates of each participant according to condition for the second part of the experiment.

during the concluding interview that might explain their problem. However, even upon removing this outlier, the data do not become any more significant.

7.3.3 Conclusions

The lack of significant differences in the results does not allow us to claim that either of the three conditions is better or worse regarding a user's ability to follow edges through a diagram. This is still an interesting observation, however: given 48 participants, if there was a large performance difference between the three conditions, we believe it to be reasonable to assume that we would have had a fair chance of finding it.

A possible threat to the validity of the experiment can be seen in the fact that participants spent quite different times on solving their task depending on the stimulus. While we believe that the way we assigned stimuli made results of different participants comparable, future experiments should control for this.

7.4 Concluding Interview

The concluding interview was meant to shed light on which label placement strategies users preferred, and why. We also wanted to give participants an opportunity to give feedback on the experiment itself. In addition to the side selection strategies they already knew from the first two parts, the interview added on-edge label placement (OE) (*without* directional decorators).

7.4.1 Experimental Method

Participants were presented with the graph shown in Figure 7.3, drawn with the four side selection strategies which they were asked to order from best (1) to worst (4). We asked them to try to assign each strategy a distinct preference rank, but accepted if they could not settle on an order between two or more strategies.

We made an audio recording of each interview—to which no participant objected—and asked them to explain their thought process while ranking the four strategies. The audio recordings were later transcribed to be coded into several categories. The coding was done separately by the experimenter and a researcher that was otherwise not involved in the experiment. Divergent opinions on how statements should be coded were resolved by going through the transcription together and discussing arguments for and against a particular coding.

7.4.2 Data and Analysis

Out of 48 preference rankings we collected, only 6 had at least two strategies that shared the same ranking. Table 7.1 shows the 42 submissions that assigned unique ranks to each strategy, and Table 7.2 shows the remaining six.

For the purpose of this analysis, we define *consistent rankings* as rankings that follow consistent preferences regarding the underlying techniques. One such consistent ranking could be to always prefer next-to-edge placement to on-edge placement and to break ties by preferring directional decorators; another could be to prefer directional decorators and to break ties by preferring on-edge placement to next-to-edge placement. A first interesting observation is that consistent rankings are the minority (18 out of 42). Among the remaining 24 inconsistent rankings, 18 rank OED highest and 13 of these rank NED lowest, possibly due to problems when inferring edge directions with the latter strategy.

Figure 7.5 visualizes the sums of the ranks assigned to each side selection strategy, which already indicates that the OED condition may have an advantage over the others. The OE condition also seems to have been ranked slightly better than both NE and NED, but the difference is far from being as pronounced. Figure 7.6 shows histograms of ranks assigned to each strategy. The histogram for the OED condition seems to support the former statement. The second statement, however, is not obvious.

Since graphics do not confirm the existence of statically significant ranking differences, we performed a Friedman test on the data and found that significant differences do in fact exist ($p < 0.01$). We proceeded with a pairwise sign test and found that the only significant differences in preference rankings involved the OED condition, which was ranked significantly better than all other conditions (always with $p < 0.01$). The OE condition did not perform significantly better than either NE and NED.

Regarding the interviews, Table 7.3 shows a summary of how often the most important statement categories occurred for each condition. For the most part, the results are not surprising. 24 participants mentioned clear graphic association as an advantage of on-edge label placement, while 21 and 11 participants mentioned unclear graphic association as a disadvantage of NED and NE, respectively. When it comes to the ease

Rank 1	Rank 2	Rank 3	Rank 4	Participants
NE	NED	OE	OED	2
NE	OE	NED	OED	0
NED	NE	OED	OE	2
NED	OED	NE	OE	0
OE	NE	OED	NED	0
OE	OED	NE	NED	4
OED	NED	OE	NE	4
OED	OE	NED	NE	6
NE	NED	OED	OE	0
NE	OE	OED	NED	1
NE	OED	NED	OE	0
NE	OED	OE	NED	2
NED	NE	OE	OED	2
NED	OE	NE	OED	0
NED	OE	OED	NE	0
NED	OED	OE	NE	1
OE	NE	NED	OED	0
OE	NED	NE	OED	0
OE	NED	OED	NE	0
OE	OED	NED	NE	0
OED	NE	NED	OE	0
OED	NE	OE	NED	3
OED	NED	NE	OE	5
OED	OE	NE	NED	10
Consistent				18
Inconsistent				24

Table 7.1: Preference assignments with distinct ranks submitted by participants. The top half of the table contains consistent rankings that follow consistent preferences regarding the underlying techniques when ranking strategies.

Rank 1	Rank 2	Rank 3
NE	OE, OED	NED
NED, OED	OE	NE
OE	NE, NED	OED
OED	NE, NED	OE
OED	NE, NED, OE	
OED	OE	NE, NED

Table 7.2: Preference assignments with shared ranks submitted by participants. Each assignment was submitted only by a single participant.

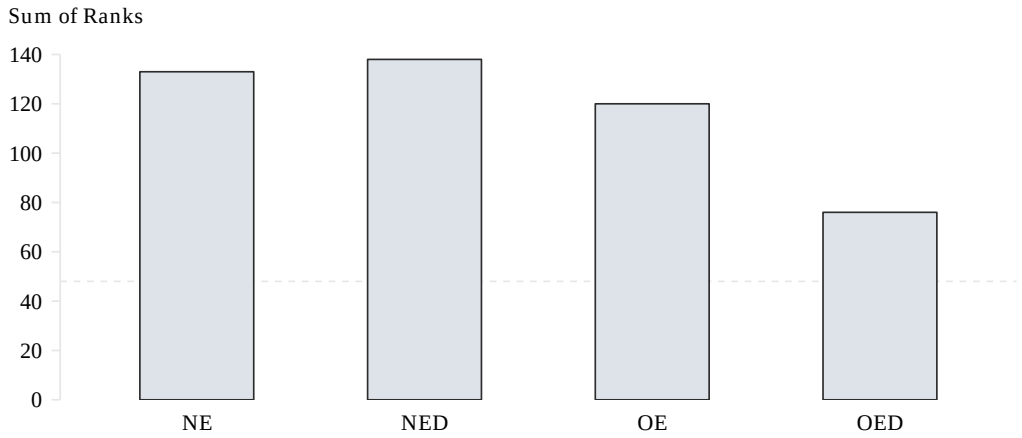


Figure 7.5: Sums of the ranks assigned by participants to each label side selection strategy from 1 (best) to 4 (worst). Lower bars indicate overall higher preference. The dashed line indicates the best score a condition could have reached (48).

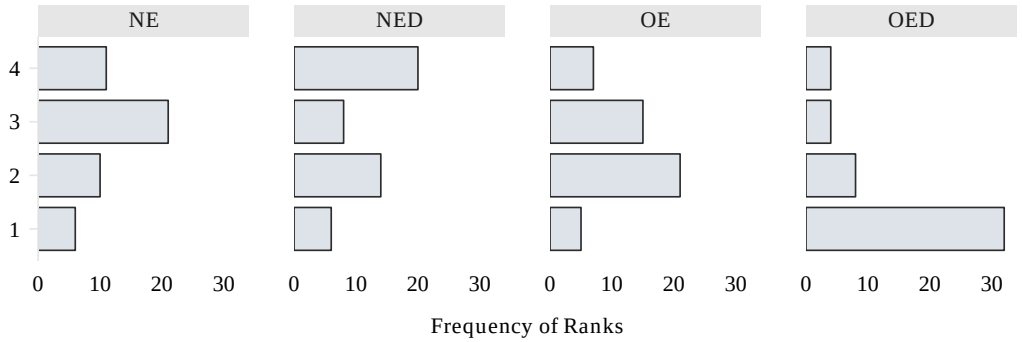


Figure 7.6: Histograms of the occurrences of each ranking for each label side selection strategy, from 1 (best) to 4 (worst).

Category	NE	NED	OE	OED	NE, NED	OE, OED
Graphic association is clear	3	1				24
Graphic association is unclear	11	21				0
Tracing edges is easy					13	5
Tracing edges is hard					0	16
Looks clear or consistent	11	2	3	7		
Looks confusing or inconsistent	1	11	0	3		
Direction easy to infer		3		33		
Direction hard to infer		27		0		
Additional information helpful		11		17		
Additional information unhelpful		4		10		

Table 7.3: Statements from interviews coded to several categories, subject to label side selection strategy. Answers for some categories are counted towards two strategies instead of just one.

of tracing edges through a diagram, the numbers are reversed: 13 mention traceability as an advantage of the next-to-edge strategies, while 16 mention it as a disadvantage of the on-edge strategies.

11 called NED confusing and inconsistent, and 27 said that it was hard to infer edge directions with. However, 10 stated that it might just be a matter of getting used to this strategy. 5 participants stated explicitly that this strategy requires explanation since encoding edge direction through placement side would not be obvious to them.

33 found OED to be helpful for inferring edge directions and 17 generally found the additional directional information to be helpful, although another 10 did not. 7 opined that the usefulness of this strategy improves as edges get longer and graphs get larger.

13 participants stated that the best strategy to use would depend on the use case.

As for feedback regarding the experiment itself, there were very few complaints. Three participants thought some of the graphs were too big for the screen, one found the task in the second part to be hard, one would have appreciated more practice tasks, and one would have liked to see their results at the end.

7.4.3 Conclusions

There is little question that, overall, participants largely preferred the OED strategy, for different reasons. First, participants found it easy to infer an edge's direction with OED, something they found hard with NED (although interestingly one participant thought he was quicker inferring directions with NED, an assessment proven wrong by the data). Second, clear graphic association was mentioned quite often. As one participant said about NED, "I would have problems associating labels with edges if there were too many edges in one place. It's a clarity thing.". However, some complained about

visual clutter and redundant information in the diagram. This might well be due to the fact that the value of additional directional markers increases as diagrams increase in size—the diagrams in the experiment were designed to be small enough to fit on a single screen to keep users from having to scroll through them. It might be valuable to make the appearance of directional decorators depend on the length of an edge. As one participant said, "For small diagrams I think I would prefer OE. If it is a large diagram with long edges, I would prefer OED. In particular in small diagrams with many edges, where everything is close by, it gets a little confusing.".

While it is understandable that many participants found graphic association to be a weakness of NED, it is interesting that 11 thought the same about NE. This might be due to a bad impression of NED carrying over to the (superficially similar) NE strategy. It might however also be due to the fact that NE becomes clear only once the viewer is aware that labels are always placed below or above their edges.

A possible threat to validity is that the first part of the experiment may have somewhat primed participants to prefer the OED condition for its obvious advantages in inferring edge directions. However, we believe that effect to be limited. First, 50% of participants indicated being aware of the fact that the value of a strategy may depend on the use case, thus acknowledging that the task imposed upon them during the first part is just one of many possible tasks. And second, 50% of participants explicitly mentioned clear graphic association as an advantage of the OE and OED strategies, while 50% also explicitly mentioned that NED suffered from unclear graphic association—an observation that the first task would probably not have primed for as much.

8 Conclusion

We have presented different placement strategies for placing edge labels in flow-based diagrams. On-edge label placement yields clearest graphical association, and usually slightly smaller diagrams. With directional decorators added it was largely preferred by participants of our experiment. Some did complain about the fact that on-edge labels interrupt their edges, but we did not find significant performance differences in a task that required participants to follow edges through a diagram.

Future Work. Some visual languages tend to produce rather long edge labels that make the assignment of labels to layers a crucial influence on the width of drawings. As we saw, the widest layer strategy did not succeed in producing smaller diagrams than the other strategies. Finding a heuristic that does seems necessary.

Clear graphic association of on-edge labels may be impaired if labels span multiple lines of text. It should be investigated whether this is indeed the case and, if so, how it can be solved.

Acknowledgements. The authors would like to thank Nadine Yarar, Andreas Mühling, and Emmanuel Manalo for helpful suggestions while designing the experiment, and Helen Purchase for her excellent book on the subject which was a great help.

Bibliography

[1] R. Castelló, R. Mili, and I. G. Tollis. An algorithmic framework for visualizing Statecharts. In *GD 2000: Proceedings of the 8th International Symposium on Graph Drawing*, volume 1984 of *LNCS*, pages 43–44. Springer-Verlag, 2001.

[2] P. Eades, X. Lin, and W. F. Smyth. A fast and effective heuristic for the feedback arc set problem. *Information Processing Letters*, 47(6):319–323, 1993.

[3] P. Eades, B. D. McKay, and N. C. Wormald. On an edge crossing problem. In *9th Australian Computer Science Conference*, pages 327–334. ACA, 1986.

[4] E. R. Gansner, E. Koutsofios, S. C. North, and K.-P. Vo. A technique for drawing directed graphs. *Software Engineering*, 19(3):214–230, 1993.

[5] M. R. Garey and D. S. Johnson. Crossing number is NP-complete. *SIAM Journal on Algebraic and Discrete Methods*, 4(3):312–316, 1983.

[6] D. Holten, P. Isenberg, J. J. van Wijk, and J.-D. Fekete. An extended evaluation of the readability of tapered, animated, and textured directed-edge representations in node-link graphs. In *2011 IEEE Pacific Visualization Symposium*, pages 195–202, mar 2011.

[7] D. Holten and J. J. van Wijk. A user study on visualizing directed edges in graphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2299–2308, New York, NY, USA, Apr. 2009. ACM.

[8] E. Imhof. Positioning names on maps. *The American Cartographer*, 2(2):128–144, 1975.

[9] M. Jünger, E. K. Lee, P. Mutzel, and T. Odenthal. A polyhedral approach to the multi-layer crossing minimization problem. In *Proceedings of the 5th International Symposium on Graph Drawing (GD'97)*, volume 1353 of *LNCS*, pages 13–24. Springer, 1997.

[10] K. G. Kakoulis and I. G. Tollis. An algorithm for labelling edges of hierachical drawings. In G. Di Battista, editor, *Graph Drawing (Proceedings GD '97)*, LNCS, pages 169–180. Springer-Verlag, 1997.

[11] G. W. Klau and P. Mutzel. Combining graph labeling and compaction. In *Proceedings of the 7th International Symposium on Graph Drawing (GD'99)*, volume 1731 of *LNCS*, pages 27–37. Springer, 1999.

[12] L. K. Klauske and C. Dziobek. Improving modeling usability: Automated layout generation for Simulink. In *Proceedings of the MathWorks Automotive Conference (MAC'10)*, 2010.

[13] D. A. Norman. *The Design of Everyday Things.* Basic Books, 1988.

[14] M. Petre. Why looking isn't always seeing: Readership skills and graphical programming. *Communications of the ACM*, 38(6):33–44, June 1995.

[15] G. Sander. Graph layout through the VCG tool. Technical Report A03/94, Universität des Saarlandes, FB 14 Informatik, 66041 Saarbrücken, Oct. 1994.

[16] A. Schelten. Hierarchy-aware layer sweep. Master's thesis, Kiel University, Department of Computer Science, Sept. 2016.

[17] C. D. Schulze, M. Spönemann, and R. von Hanxleden. Drawing layered graphs with port constraints. *Journal of Visual Languages and Computing, Special Issue on Diagram Aesthetics and Layout*, 25(2):89–106, 2014.

[18] M. Spönemann, H. Fuhrmann, R. von Hanxleden, and P. Mutzel. Port constraints in hierarchical layout of data flow diagrams. In *Proceedings of the 17th International Symposium on Graph Drawing (GD '09)*, volume 5849 of *LNCS*, pages 135–146. Springer, 2010.

[19] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, Feb. 1981.

[20] R. Tamassia, editor. *Handbook of Graph Drawing and Visualization.* CRC Press, 2013.

[21] R. von Hanxleden, M. Mendler, J. Aguado, B. Duderstadt, I. Fuhrmann, C. Motika, S. Mercer, O. O'Brien, and P. Roop. Sequentially Constructive Concurrency—A conservative extension of the synchronous model of computation. *ACM Transactions on Embedded Computing Systems, Special Issue on Applications of Concurrency to System Design*, 13(4s):144:1–144:26, July 2014.

[22] J. Wagemans, J. H. Elder, M. Kubovy, S. E. Palmer, M. A. Peterson, M. Singh, and R. von der Heydt. A century of Gestalt psychology in visual perception: I. Perceptual grouping and figure–ground organization. *Psychological Bulletin*, 138(6):1172–1217, Nov. 2012.

[23] P. C. Wong, P. Mackey, K. Perrine, J. Eagan, H. Foote, and J. Thomas. Dynamic visualization of graphs with extended labels. In *INFOVIS '05: Proceedings of the Proceedings of the 2005 IEEE Symposium on Information Visualization*, page 10, Washington, DC, USA, 2005. IEEE Computer Society.

[24] K. Xu, C. Rooney, P. Passmore, D.-H. Ham, and P. H. Nguyen. A user study on curved edges in graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 18(12):2449–2456, Dec 2012.