Automating the layout of network diagrams with specified visual organization.

Automating the Layout of Network Diagrams with Specified Visual Organization

Corey Kosak, Joe Marks, and Stuart M. Shieber

TR-22-91

Revision of April 22, 1993. An earlier version of this technical report was previously available under the title "New Approaches to Automating Network-Diagram Layout".

Automating the Layout of Network Diagrams with Specified Visual Organization

Corey Kosak Joe Marks Stuart Shieber

April 22, 1993

Abstract

Network diagrams are a familiar graphic form that can express many different kinds of information. The problem of automating network-diagram layout has therefore received much attention. Previous research on network-diagram layout has focused on the problem of aesthetically optimal layout, using such criteria as the number of link crossings, the sum of all link lengths, and total diagram area. In this paper we propose a restatement of the network-diagram layout problem in which layout-aesthetic concerns are subordinated to perceptual-organization concerns. We present a notation for describing the visual organization of a network diagram. This notation is used in reformulating the layout task as a constrained-optimization problem in which constraints are derived from a visual-organization specification and optimality criteria are derived from layout-aesthetic considerations. Two new heuristic algorithms are presented for this version of the layout problem: one algorithm uses a rule-based strategy for computing a layout; the other is a massively parallel genetic algorithm. We demonstrate the capabilities of the two algorithms by testing them on a variety of network-diagram layout problems.

To appear in IEEE Transactions on Systems, Man, and Cybernetics.

An earlier version of this technical report was previously available under the title "New Approaches to Automating Network-Diagram Layout".

1 Introduction

Network diagrams are a familiar conventional graphic form [1]; an instance is given as a placement of nodes and links, perhaps with enclosing boxes or other diacritical symbols, in a two-dimensional arrangement. They are perhaps most closely associated with conveying systems analysis and design information [22], but are used for myriad other purposes. A major task in designing a network diagram is determining its two-dimensional layout. The problem of automating network-diagram layout has consequently received much attention. One bibliography [5] cites more than 180 references for this problem.

Every approach to diagram layout requires that the diagram remain *syntactically valid*; for example, overlapping nodes and intersections between nodes and links must not be permitted. Secondary to these considerations are those of *layout aesthetics*, and it is on these considerations that previous research has focused. Eades and Tamassia [5] summarize this approach as follows:

> ...[I]n almost all data presentation applications, the usefulness of a graph depends on its readability, i.e., the capability of conveying the meaning of a diagram quickly and clearly. Readability issues are expressed by means of aesthetics, which can be formulated as optimization goals for the drawing algorithms. ...A fundamental and classic aesthetic is the minimization of crossings between edges.

(The diagram-layout literature suffers from inconsistent terminology. In this paper we prefer the terms *network diagram*, *node*, and *link* over the corresponding terms *graph*, *vertex*, and *edge*, respectively.) Besides the number of link crossings, other common layout-aesthetic criteria are diagram area, diagram aspect ratio, the number of bends in polyline links, the sum of link lengths, the length of the longest link, link-length equality, and node-density distribution [30].

Syntactic validity and layout aesthetics do not, however, account for all the important aspects of network-diagram layout. For example, human graphic designers rely routinely on grouping principles derived from the classical Gestalt Laws of perceptual psychology [12] to organize diagrams visually. Furthermore, inappropriate perceptual organization has been identified as a major cause of design flaws in informational graphics: Kosslyn [15, 16] has performed psychological experiments to support his claim that "when a visual display is difficult to interpret, violations of [the Gestalt] grouping laws are often the root of the problem," and Marks and Reiter [20] have described the effects of misleading perceptual organization on the semantic interpretation of network diagrams. We therefore generalize the problem of network-diagram layout by introducing layout considerations that concern *perceptual organization.* The resulting three categories of layout consideration—syntactic validity, perceptual organization, and aesthetic optimality—are illustrated in Figure 1, which shows a representative selection of appropriately classified layout considerations.

One contribution of this paper is a notation for describing the *visual organization* of a network diagram; our concept of visual organization subsumes both network topology

This work was supported in part by contracts from U S WEST Advanced Technologies and the Lockheed Palo Alto Research Laboratory. We are happy to acknowledge the support and advice of Tony Cox, Mark Friedell, and Barbara Grosz, and the additional help of Dave Davis, Janusz Juda, Sandeep Kochhar, and Steve Sistare.

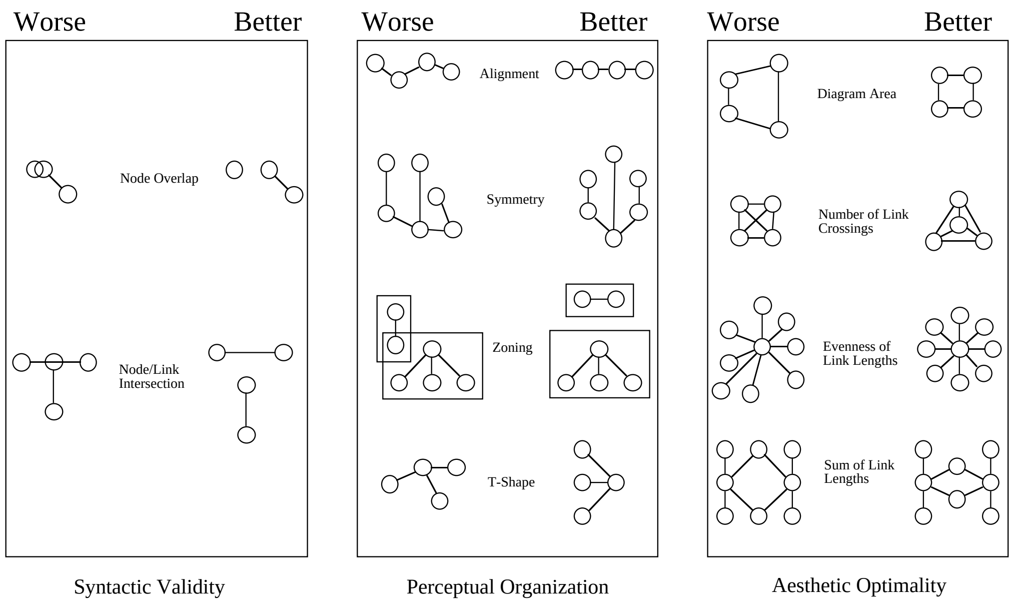


Figure 1: Layout considerations

and the perceptual organization of symbols. Using this notation, a desired visual organization can be specified that is independent of any particular layout. A visual organization for a network diagram might be specified by a human user, or it might be specified automatically.[1] The layout task can then be formulated as a constrained-optimization problem in which constraints are derived from syntactic-validity and visual-organization requirements, and optimality criteria are derived from layout aesthetics. Although the idea of incorporating constraint satisfaction into layout algorithms as a mechanism for achieving some degree of perceptual organization in network diagrams has been tried before [30, 2, 9], the visual-organization features discussed elsewhere are very different in scope and nature from those considered here.

It should come as no surprise that this constrained-optimization problem is computationally intractable, because less general formulations of the diagram-layout problem are themselves problematic. For instance, the simpler problem of laying out a graph with a minimal number of edge crossings is NP-complete [6]. Likewise the simpler task of computing a layout that merely exhibits specified perceptual groupings is NP-complete, given some reasonable assumptions concerning the size and nature of the display [17]. These results effectively rule out the existence of an efficient algorithm for finding layouts that satisfy given constraints and that are guaranteed to be aesthetically optimal.

The apparent intractability of the constrained-optimization formulation of the layout problem suggests the use of heuristics. A second contribution of this paper is the description of two new heuristic algorithms for our version of the layout problem. One

[1] The visual-organization specifications for all network diagrams that appear in this paper were generated automatically using a rule-based technique that determines an appropriate perceptual organization for the information to be communicated in a diagram [17, 19]. However, we believe that there are many application contexts in which visual-organization specifications would be best supplied by the user, perhaps through a direct-manipulation interface.

algorithm uses a rule-based strategy for computing layouts; the other is a genetic algorithm that is suitable for massively parallel computers. We demonstrate the different capabilities of the two algorithms by testing them on a selection of layout problems.

This paper is structured as follows. First, we define our notation for describing the visual organization of network diagrams. Second, we describe our rule-based approach to diagram layout. Finally, we describe our parallel genetic algorithm for the same problem. In concluding remarks, we suggest directions for future work.

2 Specifying the Visual Organization of Network Diagrams

A layout-problem statement is given as a symbolic description of the diagram's desired topology and perceptual organization. Such a *visual-organization specification* (VOS) is given as a specification of a network topology along with a list of *visual-organization features* (VOF).

The topology of a network diagram is described as a set of nodes, links, and enclosure boxes, along with a specification of the nodes that the links connect and the boxes enclose. More formally, the following components must be specified:

- N, L, and ENC, the set of *nodes*, *links*, and *enclosures* for the network.
- $TC : L \rightarrow N \times N \times \{unidirectional, bidirectional\}$, the *topological connectivity* of the network. The specification is such that $TC(l) = (n_i, n_j, d)$ just in case link l connects node n_i to node n_j with directionality d.
- $ENCLOSE : ENC \rightarrow 2^N$, the *enclosure structure* of the network. The specification is such that $ENCLOSE(e) = s$ just in case enclosure e encloses exactly the nodes in the set s.

The VOFs codify the desired visual organization of the diagram. The VOFs in our formulation concern nodes only. They specify perceptual groupings due to various kinds of proximity relations, sequentially ordered layout, alignment, axial and radial symmetry, and special, easily recognized layout patterns, such as the "T-shape" pattern that describes a conventional layout for nodes that are related hierarchically. A notation and description for our complete set of VOFs is described in the appendix. In the next paragraph we shall describe a selected few in detail. Though we do not believe that our set of VOFs is inherently exhaustive, we have noted over a period of time and through informal taxonomic research that our set of features provides excellent coverage in practice for the actual organizational primitives conventionally used in network diagrams by graphic designers.

To illustrate the general form and content of VOF predicates, and to establish enough context for a simple but complete visual-organization specification, we consider zones and clusters (two kinds of proximity relations), and axial symmetries. A zone, as codified in the *ZONES* predicate over node sets, specifies a set of nodes to be laid out such that a rectangular region of the display is reserved for these nodes only; no other nodes may intrude into the allotted region. A cluster, specified using the *CLUSTERS* predicate over sets of nodes, is related more directly to the concept of perceptual grouping by proximity: the specified nodes must be positioned close enough in the display to be perceived as a distinct gestalt. Symmetry about an axis (either horizontal or vertical, though which of the two may be unspecified) is expressed using the *SYM* predicate over sets of nodes. The nodes in a *SYM* group must be laid out so as to exhibit the specified symmetries.

A sample visual-organization specification is shown in Figure 2. The layout task is to compute an aesthetically optimal or near-optimal layout that exhibits the VOFs in the specification. A layout that exhibits the required VOFs is shown in Figure 3. The layout problem is thus cast as a constrained-optimization problem: the constraints come from the VOFs and the syntactic validity requirements, and the optimality criteria come from the layout aesthetics. We turn now to the problem of generating heuristic solutions to these inherently intractable constrained-optimization problems.

3 A Rule-Based Approach to Layout

Our motivation in using heuristic rules for layout is to emulate how human graphic designers appear to lay out diagrams. Similar sublayout patterns tend to recur repeatedly in human-designed network diagrams, suggesting that human designers utilize a small set of patterns when generating diagram layouts. This pattern-generation expertise can be captured to some degree in heuristic rules.[2] A layout is computed incrementally by augmenting a nascent layout until each node has been positioned; each augmentation of the layout is achieved by applying a layout rule. (In this paper we assume straight-line links, so node placement essentially subsumes link routing.) The network diagram in Figure 4 was laid out using the rule-based approach. The diagram depicts a local-area computer network [13]; its visual-organization specification is given in Figure 5. The numbering of the nodes indicates the order in which they were positioned by successive rule applications.

The left-hand side of a layout rule consists of a sublayout and a set of VOFs. A rule can be applied if certain *applicability criteria* are met, namely:

- if the required sublayout is found in the nascent layout
- if the required VOFs have been specified in the visual-organization specification
- if the application of the rule does not compromise syntactic validity

The successful application of a rule leads to an augmented nascent layout in which one or more additional nodes have been positioned, as indicated in the figure. Figure 6 illustrates a simple layout rule (this rule was used extensively to compute the layout shown in Figure 10). The required sublayout for the rule in Figure 6 comprises two vertically aligned nodes. These nodes participate in two VOFs, one for evenly spaced, sequentially ordered layout (*SSEQLAY*), and one for evenly spaced, horizontally or vertically aligned layout (*SALI*). Given the existing locations of nodes $n1$ and $n2$, a candidate position for $n3$ is generated that does not contravene the specified VOFs. If placing node $n3$ at this position does not invalidate the diagram on syntactic grounds (by causing symbol

[2]Sugihara et al. [29] describe a variant of rule-based layout in which layout rules are derived from examples ("layout stereotypes") stated using fuzzy logic.

Topology

$$N = \{n_1, \ldots, n_{12}\}$$
$$L = \{l_1, \ldots, l_{12}\}$$
$$ENC = \{e_1, \ldots, e_4\}$$

$$\begin{aligned}
TC = \{\ & l_1 \mapsto (n_1, n_3, unidirectional), && l_2 \mapsto (n_2, n_4, unidirectional),\\
& l_3 \mapsto (n_3, n_5, unidirectional), && l_4 \mapsto (n_4, n_7, unidirectional),\\
& l_5 \mapsto (n_5, n_8, unidirectional), && l_6 \mapsto (n_8, n_6, unidirectional),\\
& l_7 \mapsto (n_6, n_{10}, unidirectional), && l_8 \mapsto (n_7, n_{10}, unidirectional),\\
& l_9 \mapsto (n_9, n_8, unidirectional), && l_{10} \mapsto (n_{10}, n_9, unidirectional),\\
& l_{11} \mapsto (n_8, n_{11}, unidirectional), && l_{12} \mapsto (n_{10}, n_{12}, unidirectional)\}
\end{aligned}$$

$$\begin{aligned}
ENCLOSE = \{\ & e_1 \mapsto \{n_5, n_6, n_7\}, && e_2 \mapsto \{n_3, n_4, n_8, n_{10}\},\\
& e_3 \mapsto \{n_{11}, n_{12}\}, && e_4 \mapsto \{n_1, n_2, n_9\}\}
\end{aligned}$$

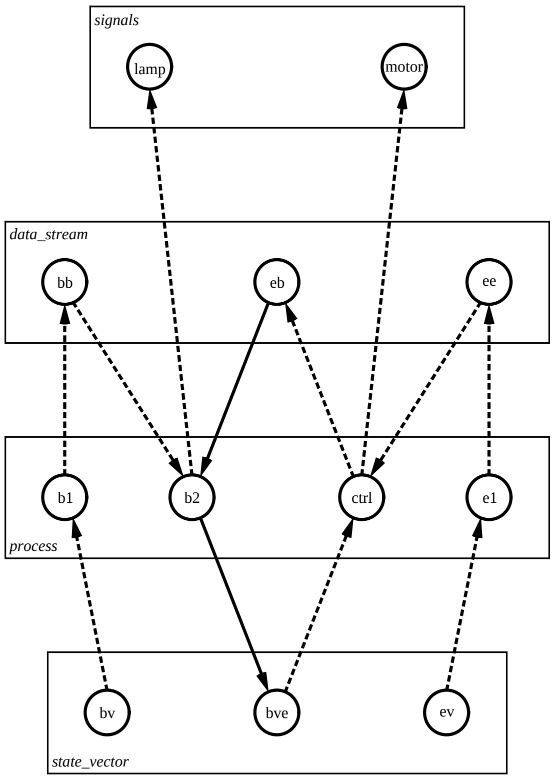
Visual Organization Features

$$\begin{aligned}
ZONES = \{\ & \{n_5, n_6, n_7\}, && \{n_3, n_4, n_8, n_{10}\},\\
& \{n_{11}, n_{12}\}, && \{n_1, n_2, n_9\}\}\\
CLUSTERS = \{\ & \{n_5, n_6, n_7\}, && \{n_3, n_4, n_8, n_{10}\},\\
& \{n_{11}, n_{12}\}, && \{n_1, n_2, n_9\}\}\\
SYM = \{\ & (\{n_5, n_6, n_7\}, vertical), && (\{n_5, n_6, n_7\}, horizontal),\\
& (\{n_3, n_4, n_8, n_{10}\}, vertical), && (\{n_3, n_4, n_8, n_{10}\}, horizontal),\\
& (\{n_{11}, n_{12}\}, vertical), && (\{n_{11}, n_{12}\}, horizontal),\\
& (\{n_1, n_2, n_9\}, vertical), && (\{n_1, n_2, n_9\}, horizontal)\}
\end{aligned}$$

Labeling Information

$$\begin{aligned}
NODE\ LABELS = \{\ & n_1 \mapsto \text{“}ev\text{”}, && n_2 \mapsto \text{“}bv\text{”},\\
& n_3 \mapsto \text{“}e1\text{”}, && n_4 \mapsto \text{“}b1\text{”},\\
& n_5 \mapsto \text{“}ee\text{”}, && n_6 \mapsto \text{“}eb\text{”},\\
& n_7 \mapsto \text{“}bb\text{”}, && n_8 \mapsto \text{“}ctrl\text{”},\\
& n_9 \mapsto \text{“}bve\text{”}, && n_{10} \mapsto \text{“}b2\text{”},\\
& n_{11} \mapsto \text{“}motor\text{”}, && n_{12} \mapsto \text{“}lamp\text{”}\}\\
ENCLOSURE\ LABELS = \{\ & e_1 \mapsto \text{“}data\_stream\text{”}, && e_2 \mapsto \text{“}process\text{”},\\
& e_3 \mapsto \text{“}signals\text{”}, && e_4 \mapsto \text{“}state\_vector\text{”}\}
\end{aligned}$$

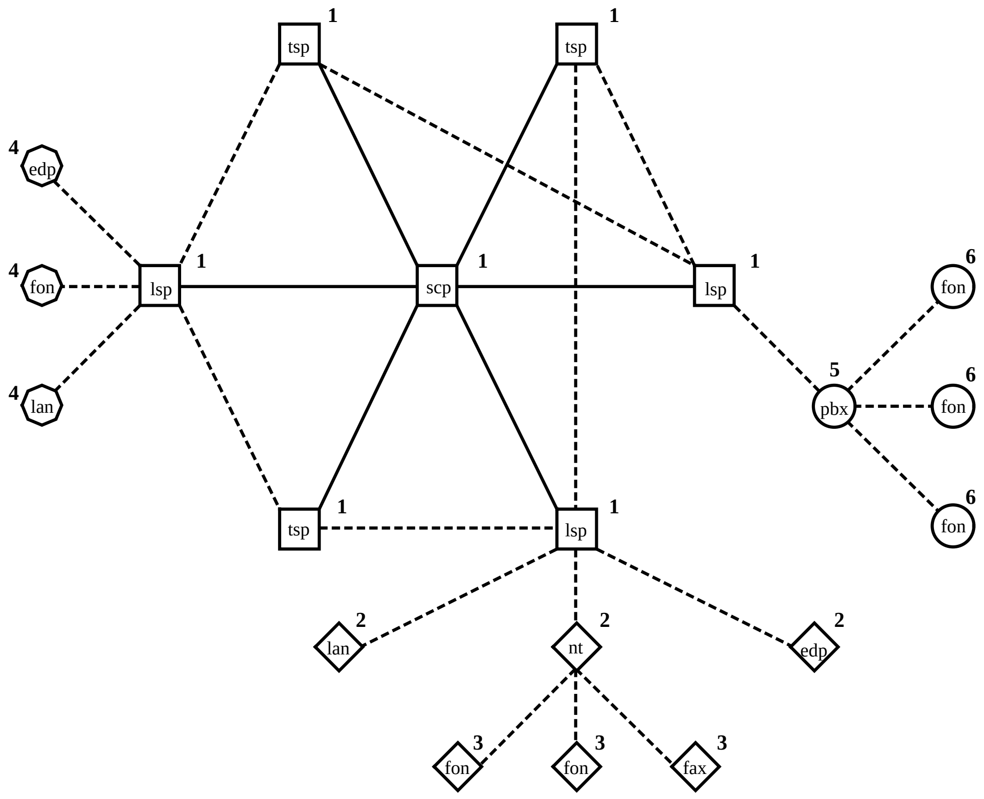
Figure 2: A sample visual-organization specification



Legend

- Text label ⇒ Vertex name
- Enclosure boxes ⇒ Vertex type
- Link pen type ⇒ Edge type
  - - - → single
  ——→ multiple

Figure 3: A network diagram with the appropriate VOFs



Legend

- Text label ⇒ Vertex type
- Node shape ⇒ Vertex affiliation

□	public network
○	warehouse
⬡	headquarters
◇	sales office

- Link pen type ⇒ Edge type

- - - -	No. 7 signaling
——	Network voice/data paths

Figure 4: A layout computed using the rule-based approach

Topology

$$N = \{n_1, \ldots, n_{20}\}$$
$$L = \{l_1, \ldots, l_{25}\}$$

$$\begin{aligned}
TC = \{ \; & l_1 \mapsto (n_1, n_3, bidirectional), & l_2 \mapsto (n_1, n_4, bidirectional),\\
& l_3 \mapsto (n_1, n_7, bidirectional), & l_4 \mapsto (n_1, n_6, bidirectional),\\
& l_5 \mapsto (n_1, n_5, bidirectional), & l_6 \mapsto (n_1, n_2, bidirectional),\\
& l_7 \mapsto (n_2, n_3, bidirectional), & l_8 \mapsto (n_3, n_4, bidirectional),\\
& l_9 \mapsto (n_4, n_7, bidirectional), & l_{10} \mapsto (n_6, n_7, bidirectional),\\
& l_{11} \mapsto (n_5, n_6, bidirectional), & l_{12} \mapsto (n_2, n_5, bidirectional),\\
& l_{13} \mapsto (n_3, n_8, bidirectional), & l_{14} \mapsto (n_8, n_9, bidirectional),\\
& l_{15} \mapsto (n_8, n_{10}, bidirectional), & l_{16} \mapsto (n_8, n_{11}, bidirectional),\\
& l_{17} \mapsto (n_5, n_{12}, bidirectional), & l_{18} \mapsto (n_5, n_{13}, bidirectional),\\
& l_{19} \mapsto (n_5, n_{14}, bidirectional), & l_{20} \mapsto (n_7, n_{15}, bidirectional),\\
& l_{21} \mapsto (n_7, n_{16}, bidirectional), & l_{22} \mapsto (n_7, n_{20}, bidirectional),\\
& l_{23} \mapsto (n_{16}, n_{17}, bidirectional), & l_{24} \mapsto (n_{16}, n_{18}, bidirectional)\\
& l_{25} \mapsto (n_{16}, n_{19}, bidirectional)\}
\end{aligned}$$

Visual Organization Features

$$\begin{aligned}
ZONES = \; & \{\{n_1, n_2, n_3, n_4, n_5, n_6, n_7\}\}\\
HUB = \; & \{(n_1, \{n_2, n_3, n_4, n_5, n_6, n_7\})\}\\
T\text{-}SHAPE = \; & \{ \; (n_8, \{n_9, n_{10}, n_{11}\}), \quad (n_5, \{n_{12}, n_{13}, n_{14}\}),\\
& (n_7, \{n_{15}, n_{16}, n_{20}\}), \quad (n_{16}, \{n_{17}, n_{18}, n_{19}\})\}
\end{aligned}$$

Labeling Information

$$\begin{aligned}
NODE\ LABELS = \{ \; & n_1 \mapsto \text{“}scp\text{”}, & n_2 \mapsto \text{“}tessp\text{”},\\
& n_3 \mapsto \text{“}lessp\text{”}, & n_4 \mapsto \text{“}tessp\text{”},\\
& n_5 \mapsto \text{“}lessp\text{”}, & n_6 \mapsto \text{“}tessp\text{”},\\
& n_7 \mapsto \text{“}lessp\text{”}, & n_8 \mapsto \text{“}pabx\text{”},\\
& n_9 \mapsto \text{“}phone\text{”}, & n_{10} \mapsto \text{“}phone\text{”},\\
& n_{11} \mapsto \text{“}phone\text{”}, & n_{12} \mapsto \text{“}lan\text{”},\\
& n_{13} \mapsto \text{“}phone\text{”}, & n_{14} \mapsto \text{“}edp\text{”},\\
& n_{15} \mapsto \text{“}lan\text{”}, & n_{16} \mapsto \text{“}nt\text{”},\\
& n_{17} \mapsto \text{“}phone\text{”}, & n_{18} \mapsto \text{“}phone\text{”},\\
& n_{19} \mapsto \text{“}fax\text{”}, & n_{20} \mapsto \text{“}edp\text{”}\}
\end{aligned}$$

Figure 5: Another sample visual-organization specification

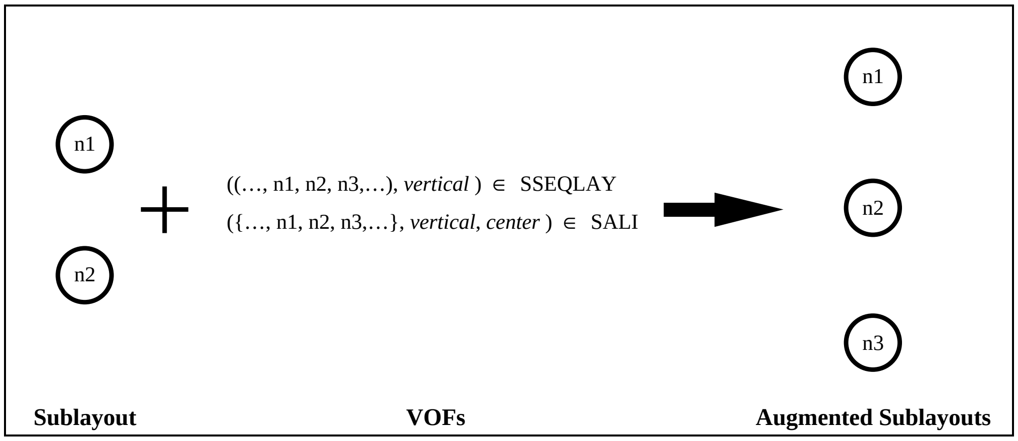

n1
n2
((…, n1, n2, n3,…), *vertical*) ∈ SSEQLAY
({…, n1, n2, n3,…}, *vertical, center*) ∈ SALI
n1
n2
n3
Sublayout
VOFs
Augmented Sublayouts

Figure 6: A simple layout rule

overlaps, for example), the nascent layout is augmented with the new position for node $n3$.

Many layout rules in our current rule base are as simple as the one in Figure 6. Sometimes, however, more powerful rules are useful. One way a rule can be made more powerful is by the use of more general applicability criteria. These criteria may require computations that potentially involve all parts of the current nascent layout and all VOFs in the specification. For example, an alternate version of the rule in Figure 6 might utilize a predicate that inhibits application of the rule if the layout augmentation would result in too many additional link crossings. General applicability criteria can thus be used to include important layout heuristics that cannot otherwise be expressed. Another way a layout rule can be made more powerful is by parameterizing its layout augmentation. For example, the width, height, and orientation of a layout augmentation are parameters that might be modified to generate more candidate layout augmentations. The rules in Figures 7 and 8 (the former rule was used to position most of the nodes in Figure 4) illustrate various different layout augmentations that result from parameter changes.

Our implementation of the rule-based algorithm is written in Prolog [28, 25]. Prolog's resolution-based search strategy is used to find sublayout patterns in the nascent layout and VOFs in the visual-organization specification. Verifying the validity of a candidate sublayout requires only simple checks for overlapping symbols and intersecting enclosures. The current rule-based layout system has fewer than 50 rules, though each rule may have several possible layout augmentations at its disposal. The actual layout rules used are a little more complex than those depicted in the figures. For example, additional candidate layout augmentations are generated by permuting the positions of nodes in the augmented sublayouts. Furthermore, although the rules in the figures appear to apply to node sets of fixed cardinality, the corresponding Prolog versions of these rules can accommodate nodes sets of variable cardinality: thus there is only one rule like the one in Figure 7, not multiple rules that differ only in the number of nodes involved.

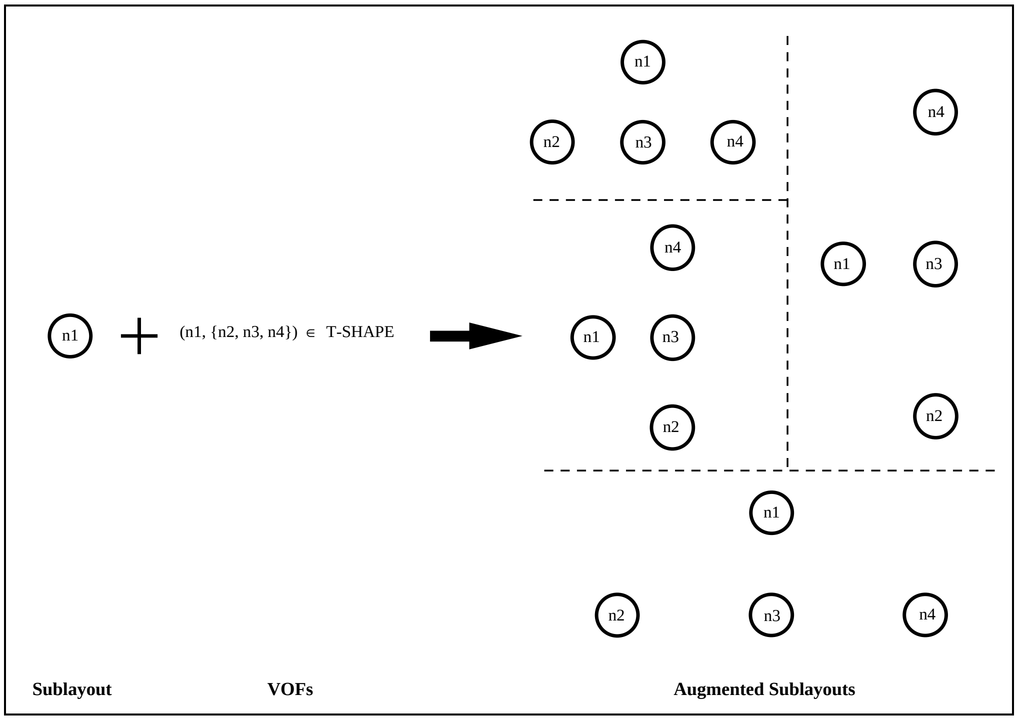


Figure 7: A layout rule with multiple augmentation options

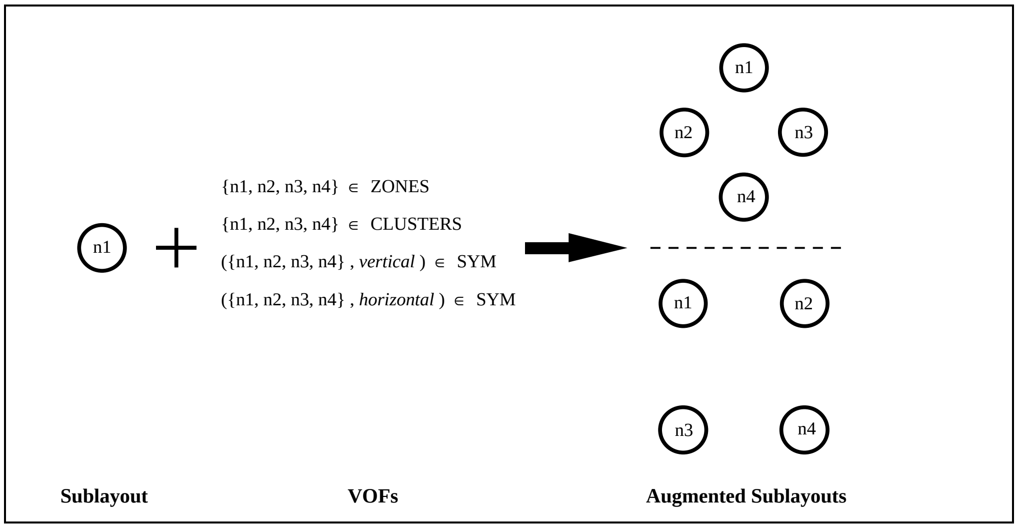

n1
{n1, n2, n3, n4} ∈ ZONES
{n1, n2, n3, n4} ∈ CLUSTERS
({n1, n2, n3, n4} , vertical) ∈ SYM
({n1, n2, n3, n4} , horizontal) ∈ SYM
n1
n2
n3
n4
Sublayout
VOFs
Augmented Sublayouts

Figure 8: Another layout rule with multiple augmentation options

These added complexities are represented and processed easily in Prolog. Marks [17] provides a more detailed discussion of the rule base.

Even with a small rule base containing relatively primitive rules (for instance, none of the standard layout aesthetics figure in the applicability criteria of our current set of rules) our rule-based layout system often computes excellent layouts. These layouts are also computed very quickly, i.e., usually in under 10 seconds on a Sun-4 for diagrams with fewer than 20 nodes. Figures 9 and 10 illustrate two of the more than 30 different layout problems on which we have tested our algorithms. Both network diagrams convey the same information as the diagram in Figure 3 (the diagrams depict an elevator control system [26]). The diagram in Figure 9 has the visual-organization specification shown in Figure 2. The diagram in Figure 10 has the same topology, but a different visual-organization specification that includes VOFs for evenly-spaced alignment ($SALI$) and sequential layout ($SSEQLAY$).[3]

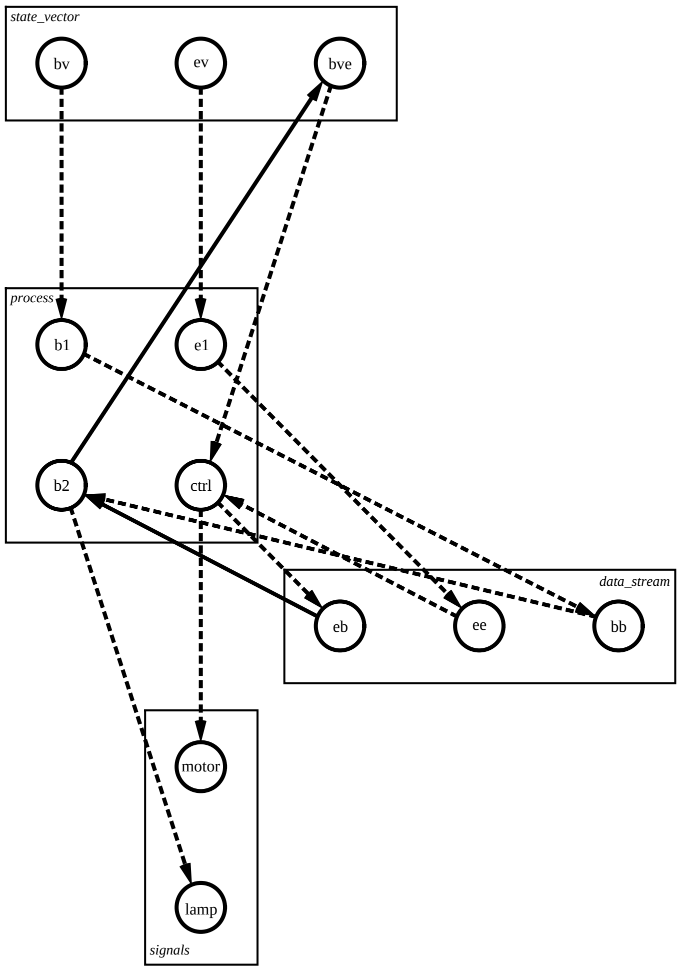
The layouts generated using the rule-based approach are qualitatively very different from those generated using any previously reported layout algorithm [5]. The unique qualities of the rule-based approach stem from the efficient generation of sublayouts that have a desired visual organization. Another advantage of layout rules is that graphic-design knowledge is represented in a form that is easily comprehended, thus facilitating the modification and extension of a rule base.[4] However, the rule-based approach has significant weaknesses that include:

- *No guarantee of success:* There is no guarantee that a valid diagram layout will always be found if one exists. In fact, this is a problem for any efficient solution to the layout problem, because merely determining if a layout with the desired VOFs exists is an NP-complete problem [17].

- *Occasionally unacceptable performance:* Prolog's resolution-based search strategy can be used to automatically backtrack when no layout rules are applicable for a given nascent layout, but excessive backtracking can lead to unacceptable performance. There is no obvious way to formulate an intelligent backtracking strategy that could quickly identify and undo infelicitous layout decisions. The rule base has therefore been designed in a way that reduces the likelihood of having to undo layout decisions. This is achieved by including in the rule base some weak, but very generally applicable rules, thus ensuring that some layout rule is applicable for almost every nascent layout. The more complex (and potentially useful) rules in the rule base are always tried first, but if none are applicable some weak rule can usually be applied instead of backtracking. This strategy can have a negative impact on diagram quality (in particular, some desired VOFs may not appear in the diagram), but will avoid expensive backtracking most of the time.

[3]All graphical aspects of these diagrams were designed by the ANDD system [17, 19]. The ANDD system first generates a visual-organization specification for a network diagram. The specifications generated by ANDD are more general than the ones discussed here. They include additional VOFs that describe perceptual grouping by similarity, perceptual ordering, and perceived magnitude. Having generated a visual-organization specification, ANDD then instantiates a network diagram that is consistent with the specification. A comparison of human-designed and ANDD-designed network diagrams is given in [18].

[4]The current rule base used by ANDD is only one of probably many useful rule bases. Different rule bases might reflect different layout styles, and may be better for some kinds of network diagrams than others. The development and comparison of different rule bases is a possible goal for future research.



Legend

- Text label ⇒ Vertex name
- Enclosure boxes ⇒ Vertex type
- Link pen type ⇒ Edge type
  - - - → single
  ——→ multiple

Figure 9: A second layout computed using the rule-based approach

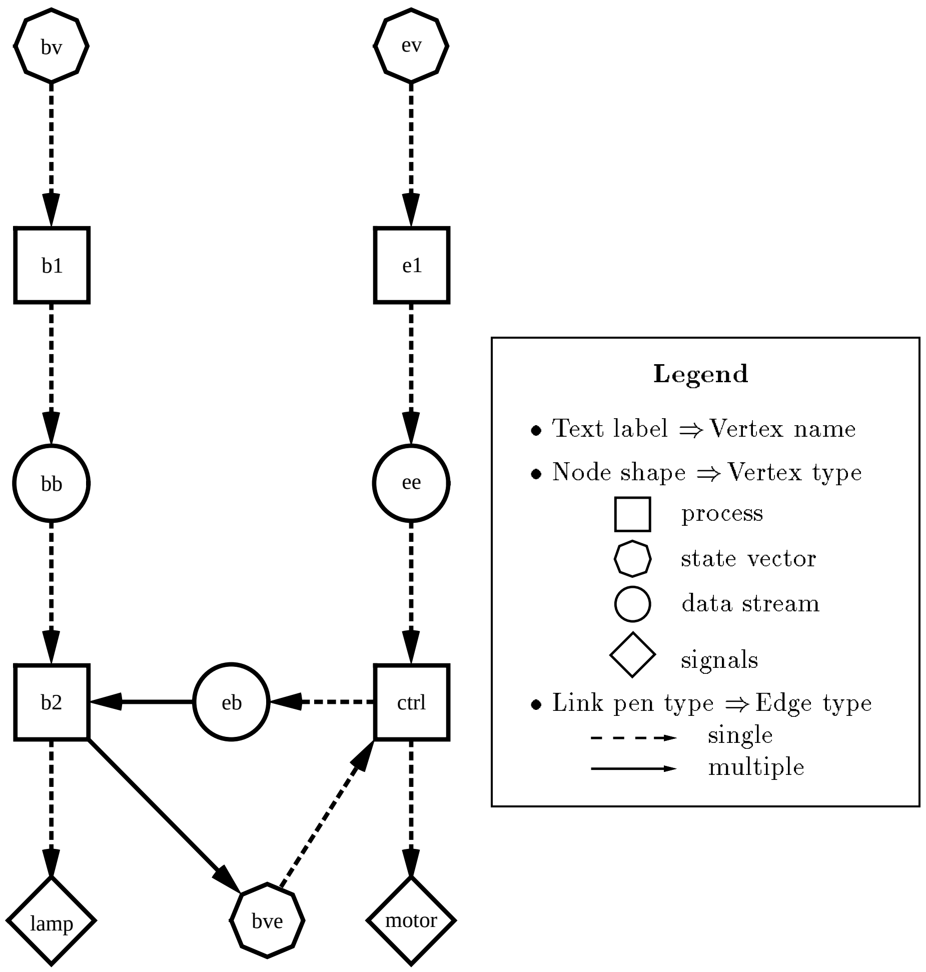


Figure 10: A third layout computed using the rule-based approach

- *Introduction of unwanted VOFs:* There is no guarantee that the algorithm will not introduce unwanted VOFs inadvertently, possibly resulting in a diagram that carries unwanted implicatures [20]. It is hard to imagine a heuristic layout strategy that could efficiently guarantee not to introduce unwanted VOFs. At the very least such a strategy would have to incorporate some computational methods for detecting VOFs in a given diagram, which is essentially a computer-vision problem. Fortunately, the more VOFs in a diagram, the harder it usually is to introduce additional unwanted ones. By concentrating on including desired VOFs, the likelihood of inadvertently introducing unwanted ones is reduced. Both our layout algorithms rely tacitly on this empirical observation.

- *Difficulties with certain layout aesthetics:* Most layout aesthetics, e.g., the number of link crossings in a diagram, are global in nature. Layout rules, on the other hand, are essentially local in nature. The effects of local layout decisions on any global aesthetic will be difficult to gauge, making it very hard to guarantee even near-optimal layouts when layout decisions are made on a local basis. This problem is apparent, for instance, in the layout of Figure 9 with its large number of link crossings.

- *Problems posed by interacting VOFs:* Interacting VOFs, i.e., different VOFs that apply to the same node(s), pose difficult problems for a rule-based approach to layout, particularly when the interacting VOFs are mutually inconsistent. Interactions—especially those that result in inconsistencies—can often be resolved by compromise layouts that exhibit aspects of all interacting VOFs, but formulating rules that can forge compromise layouts is not easy.

So although a rule-based approach may be useful in some contexts (e.g., for applications that require near real-time design of network diagrams, or in situations where an automatically designed diagram may be improved by a human designer), the inherent shortcomings of layout rules have led us to investigate another approach to the layout problem based on the idea of layout as constrained optimization.

4 A Stochastic-Optimization Approach to Layout

The more expensive stochastic-optimization approach to diagram layout described in this section is superior to the rule-based approach in that it provides simple mechanisms for taking layout aesthetics and interacting VOFs into account, and it uses a powerful search strategy that is a better and more efficient method for exploring alternative layouts than the backtracking strategy implicit in the rule-based approach. The algorithm is a parallel genetic algorithm [7, 3] adapted to both the diagram-layout problem and to the massively parallel SIMD architecture of the Connection Machine [8]. A high-level description of the algorithm is presented in Figure 11. A more detailed description of the algorithm is provided in [14]. The layouts computed by this algorithm are node embeddings on a fixed-size integer grid. Each (virtual) processor in the Connection Machine stores one layout in its local memory. The initial *generation* of layouts is chosen randomly, with each processor computing its own random layout. Subsequent generations are computed by *mating* and *mutating*, two "genetic" operations that can alter existing

Definitions:

Each processor has eight neighboring processors.
A processor and its neighboring processors constitute a neighborhood.

Procedures and Functions:

initialize—compute a random node layout.
evaluate—compute layout fitness.
mutate—randomly translate one or more nodes.
crossover—copy node locations from a neighboring processor.
reproduce—copy a layout from a processor in the neighborhood.
scatter—send a layout to a randomly chosen processor.
find-best—find the best layout of the current generation.
A '?' suffix indicates stochastic application of a procedure.

Algorithm:

```
parallel {initialize}
repeat
    parallel
    {
        evaluate
        reproduce
        scatter?
        crossover?
        mutate?
        find-best
    }
until no improved layout has been found for $N$ generations
return the best layout found
```

Figure 11: Outline of a parallel genetic algorithm for diagram layout

layouts. Prior to mating, each layout is evaluated and assigned a single numeric score to indicate its *layout fitness*. This requires approximating the constrained-optimization problem described in Section 2 as a pure optimization problem, so that the quality of a layout can be expressed with a single value. A simple method for performing this approximation is to develop scoring methods for the constraints as well as the layout aesthetics, combining these scores according to a weighted sum. The magnitude of the weights corresponding to the syntactic validity subterms are largest; a syntactically invalid diagram ought to have a very poor score. These weights dominate those corresponding to the degree of conformity with the visual-organization specification, which in turn dominate those corresponding to layout-aesthetic quality.

Fitness is therefore determined by the following layout formula:

$$\mathit{fitness} = \left(\begin{array}{ll} & (\mathit{Degree\ of\ Syntactic\ Validity}) \\ + & (\mathit{Degree\ of\ Conformity\ to\ the\ Visual\text{-}Organization\ Specification}) \\ + & (\mathit{Layout\text{-}Aesthetic\ Quality}) \end{array} \right)$$

Each term of this formula is composed of several subterms, each of which is assigned an empirically determined weight that corresponds to its relative importance [14]. We discuss the three terms and their subterms below.

- *Degree of syntactic validity:* This term has the following form:

  $$\mathit{Degree\ of\ Syntactic\ Validity} = s_1 \times |\mathit{Node\_Overlaps}| + s_2 \times |\mathit{Node\_Invasions}|$$

  Node_Overlaps is the set of all pairs of overlapping nodes. *Node_Invasions* is the set of all pairs of intersecting links and nodes. The factors s_1 and s_2 are the empirically determined weighting factors.

- *Degree of conformity to the visual-organization specification:* In the following formula, we write $c(V)$ as an abbreviation for "degree of conformity to visual-organization specification V". For example, the term $c(\mathit{NP\text{-}PROX})$ is the candidate diagram's degree of conformity to the *NP-PROX* specification.

  $$\mathit{Degree\ of\ VOS\ Conformity}$$
  $$= \left(\begin{array}{llll} & v_1 \times c(\mathit{NP\text{-}PROX}) & + & v_2 \times c(\mathit{NN\text{-}PROX}) \\ + & v_3 \times c(\mathit{ZONES}) & + & v_4 \times c(\mathit{CLUSTERS}) \\ + & v_5 \times c(\mathit{SEQLAY}) & + & v_6 \times c(\mathit{SSEQLAY}) \\ + & v_7 \times c(\mathit{ALI}) & + & v_8 \times c(\mathit{SALI}) \\ + & v_9 \times c(\mathit{SYM}) & + & v_{10} \times c(\mathit{T\text{-}SHAPE}) \\ + & v_{11} \times c(\mathit{PERIMETER}) & + & v_{12} \times c(\mathit{PERIMETER\text{-}CYCLE}) \\ + & v_{13} \times c(\mathit{HUB}) & + & v_{14} \times c(\mathit{HUB\text{-}CYCLE}) \end{array} \right)$$

  Like the s_i, the v_i are empirically determined weighting factors. The complete list of visual-organization specifications, along with a description of the computation of $c(V)$, is given in the appendix.

- *Layout-aesthetic quality:* This term has the following form:

$$\textit{Layout-Aesthetic Quality} = \left(\begin{array}{rl} & a_1 \times (\textit{Total Diagram Area}) \\ + & a_2 \times (\textit{Number of Link Intersections}) \\ + & a_3 \times (\textit{Sum of Link Lengths}) \\ + & a_4 \times (\textit{Degree of Link Length Inequality}) \\ + & a_5 \times (\textit{Degree of Global Symmetry}) \end{array} \right)$$

  As above, the a_i are empirically determined weighting factors. For certain diagrams, it may be appropriate to disable one or more of these aesthetics (by zeroing the corresponding weighting factor). This can be done either under program control or by the user.

  The terms in this formula are intended to: encourage smaller diagrams, discourage link intersections, minimize both link lengths and the differences between link lengths, and promote a global symmetry in the diagram. The computation of the first three terms is trivial. The fourth is computed as the summed absolute difference of the link lengths and the average link length:

$$\sum_{i \in L} abs(\textit{link-length}_i - mean(\textit{link-length}))$$

  where $\textit{link-length}_i$ is the length of link i, and $mean(\textit{link-length})$ is the average length of all the links. The final symmetry term is computed using the same methodology as for the *SYM* VOF described in the appendix.

Once layout fitnesses have been computed, mating can occur. There are two kinds of mating, *reproduction* and *crossover*. Reproduction is primarily a local operation. Each processor overwrites its own layout with a layout from one of the processors in its neighborhood. It seems intuitively obvious that we would want good layouts to be favored over bad in the reproduction step—indeed, the directional component of the algorithm's search strategy derives solely from reproduction. However, the simple heuristic of always copying the best layout in the neighborhood is rarely the most effective reproduction strategy. Instead, reproduction is governed by a *reproduction schedule*, which sets a probability for copying the n-th best layout in a neighborhood. For example, a good reproduction schedule may heavily favor the best and second-best layouts in a neighborhood, but may still assign non-zero probabilities to the copying of all neighborhood layouts, even the worst one.

The reproduction operation ensures that good layouts will migrate from processor to processor; the crossover operation ensures that good *sublayouts* migrate between processors. A processor performs crossover by copying the locations of one or more randomly chosen nodes from a layout in a neighboring processor. The crossover operation may be applied to individual nodes or to groups of nodes that are perceptually related, as indicated in the visual-organization specification.[5]

[5]Repeated applications of the reproduction operator tend to create homogeneous neighborhoods or "colonies", in which neighboring processors have substantially similar layouts. Because it would seem that crossover should work poorly in these kinds of neighborhoods, we developed the *scatter* operator, which copies layouts between randomly chosen pairs of processors—in effect creating "immigration" into the colonies. However, our experiments showed that although the scatter operator was able to decrease homogeneity, it had a negligible effect on the quality of the final layouts [14].

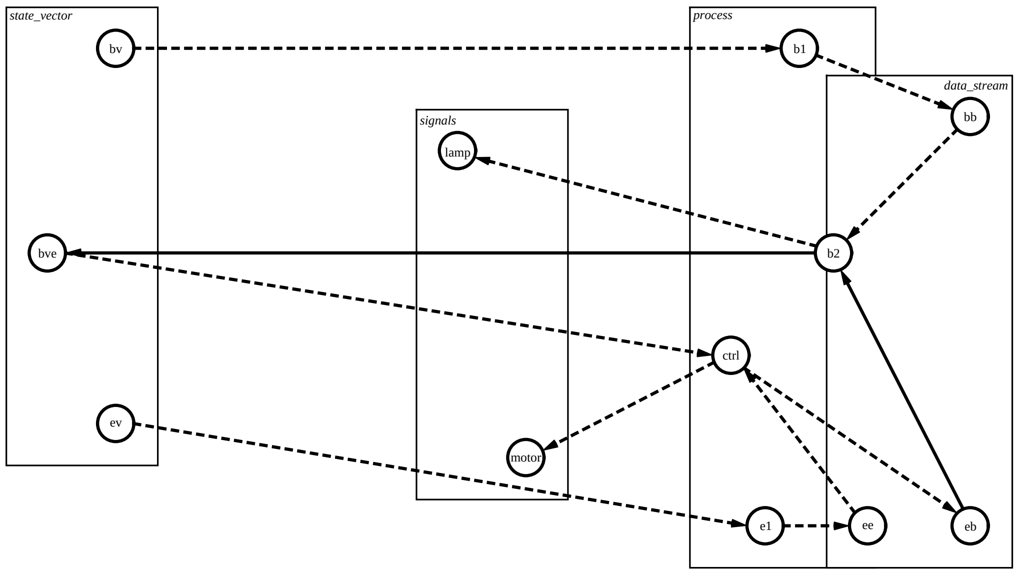
The mating operations—reproduction and crossover—are of limited utility by themselves because they cannot generate new sublayouts, i.e., sublayouts that were not in some layout of the initial generation. New sublayouts are generated by the *mutation* operation. A processor performs mutation by randomly moving one or more nodes in a layout. Like crossover, mutation may be applied to individual nodes or to perceptually related groups of nodes.

After each new generation of layouts has been computed, all processors are examined to determine which one has the best layout, which is then recorded. The program terminates if the average improvement in the best layout over a set number of generations falls below a certain threshold. Then the overall best layout from all generations is returned. Figure 3 contains a network diagram designed by the ANDD+ system that incorporates this layout algorithm. The network diagram in Figure 3 has the visual-organization specification given in Figure 2. Global symmetry, minimization of link crossings, and area minimization were the layout aesthetics used. This network diagram can therefore be compared directly to the one in Figure 9 that was laid out using the rule-based approach. The number of link crossings is the most salient difference between the diagrams.

The layout in Figure 3 evolved over 128 generations. To illustrate the way in which a layout evolves, we have included two layouts that were generated at intermediate points in the algorithm's progress. The best layout that had evolved after 17 generations is shown in Figure 12. The diagram is not even syntactically well-formed; for instance, enclosure boxes overlap. By the 69th generation (Figure 13), a syntactically valid diagram has evolved. This diagram exhibits approximately the right perceptual organization, has no link crossings, and is almost symmetric about a horizontal axis. By the final generation, the syntactic and VOF constraints are satisfied completely by a totally symmetric layout (symmetric about a vertical axis) that contains no link crossings. It is interesting to note that the final dominant layout is oriented differently from the earlier dominant layouts. It is therefore likely that the final layout did not evolve from these earlier layouts, but from a line of layouts that only achieved dominance late in the game.

Another diagram laid out by ANDD+ is shown in Figure 14. It was designed to conform to the visual-organization specification that led to the network diagram shown in Figure 10. For this particular diagram the rule-based and the genetic layout algorithms both appear to work well. However, the global symmetry of the layout computed by the genetic algorithm was achieved by design, whereas the global symmetry achieved with the rule-based approach was somewhat serendipitous. A bigger layout computed using the genetic algorithm—the diagram depicts a causal financial model [27]—is shown in Figure 15. This diagram exhibits the same kinds of VOFs that are evident in Figure 3.

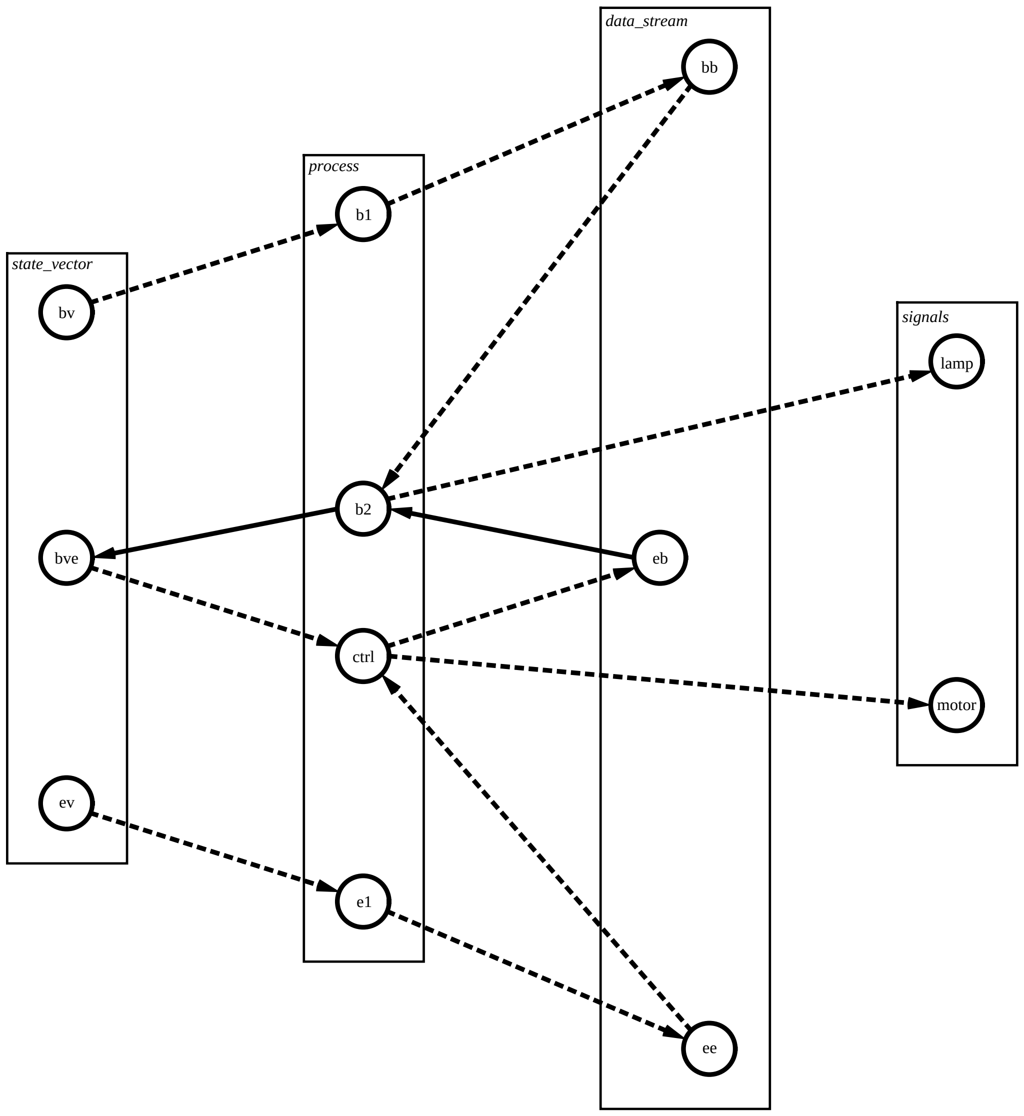
The genetic-algorithm approach to network-diagram layout shares some of the drawbacks of the rule-based approach: there is still no guarantee that a valid (or optimal) layout will be found if one exists, and there is also no guarantee that unwanted VOFs will not be introduced inadvertently. The algorithm is also quite expensive. Most small (1–20 nodes) or medium-sized (20–50 nodes) diagrams require 100–300 generations for a good layout to evolve, where each generation comprises 4,096 individual layouts. For small diagrams a new generation usually evolves in under a second on the Connection Machine (a CM-2), so the layout in Figure 3 took about four minutes to evolve. This slow performance is probably due to the fairly severe limitations of the individual processors in the CM-2.



Legend

- Text label ⇒ Vertex name
- Enclosure boxes ⇒ Vertex type
- Link pen type ⇒ Edge type
  - ⇢ single
  - → multiple

Figure 12: The best layout after 17 generations



Legend

- Text label ⇒ Vertex name
- Enclosure boxes ⇒ Vertex type
- Link pen type ⇒ Edge type
  - - - → single
  ——→ multiple

Figure 13: The best layout after 69 generations

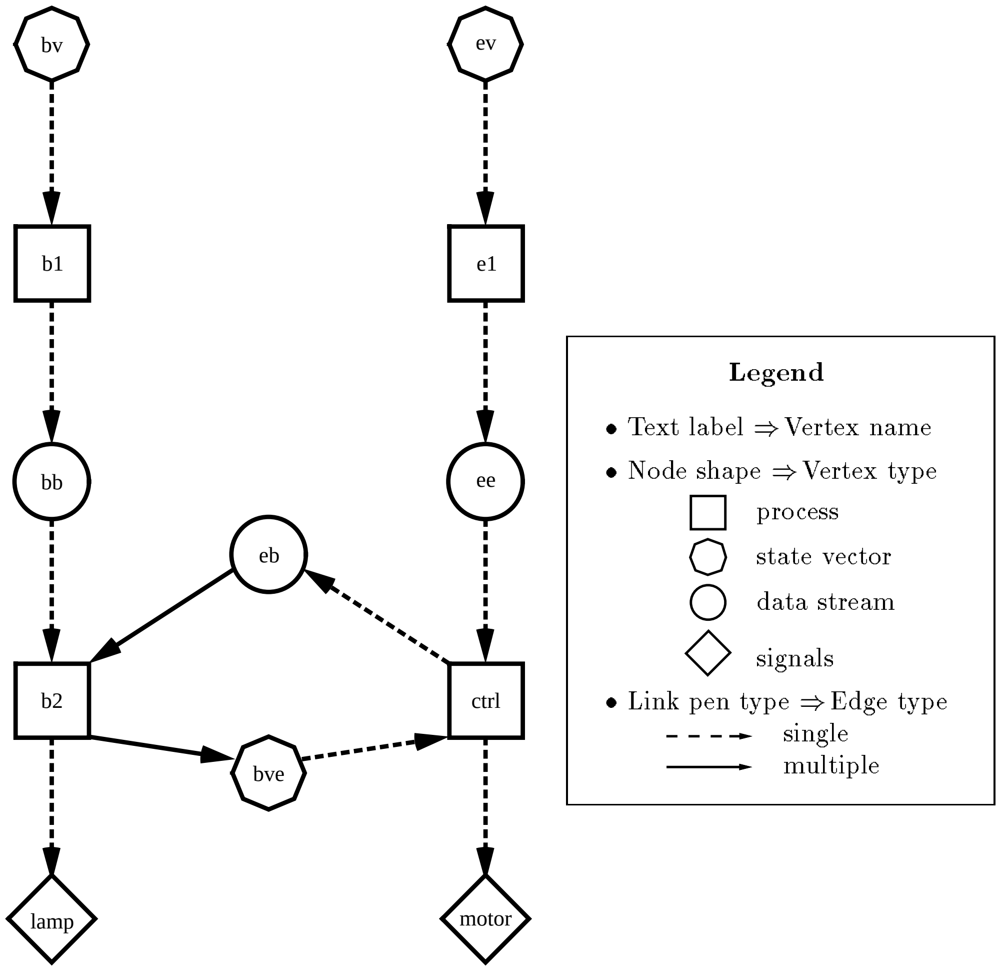


Figure 14: Another layout computed using the genetic algorithm

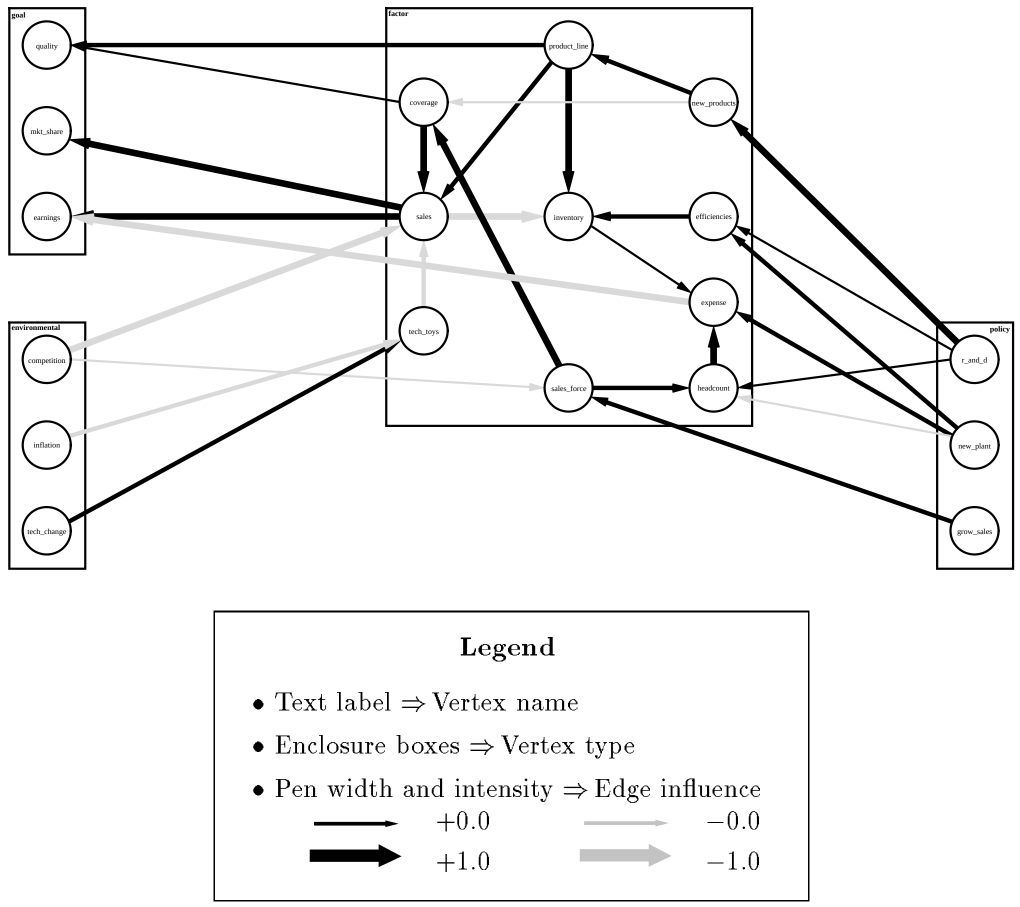


Figure 15: A larger layout due to the genetic algorithm

The advantages of the genetic-algorithm approach, however, are considerable: the notion of layout fitness provides a robust mechanism for taking multifarious layout aesthetics into account and for creatively handling interacting VOFs; the search strategy supports the directed exploration of many different solutions to the layout problem; and, most important of all, the layouts computed by this algorithm are usually of excellent quality. Furthermore, newer SIMD parallel computers with more powerful processors should be able to run the algorithm much more quickly; serial implementations of the genetic-algorithm approach may also give acceptable performance [21, 23, 24].

5 Conclusions and Future Work

We have described a novel version of the network-diagram-layout problem in which perceptual organization is given priority over purely syntactic aesthetic considerations. We discussed two new layout algorithms for this problem: one uses heuristic layout rules to incrementally compute a layout; the other uses a massively parallel genetic algorithm to explore the solution space of possible layouts. Both algorithms compute layouts that are qualitatively different from those computed using existing automatic methods. Another major advantage shared by the algorithms is flexibility: they are both easily adapted to take new perceptual gestalts and layout aesthetics into account. For example, new layout rules can be incorporated easily into an existing rule base. Likewise, terms corresponding to new gestalts and aesthetics can be introduced into the layout-fitness formula used by the genetic algorithm. Our algorithms are incorporated into the ANDD and ANDD+ systems that automate fully the design of network diagrams with a given semantic content, but we believe that they will also prove useful in traditional CAD environments for the semi-automated development of network diagrams.

In future work we hope to improve the performance and efficiency of our algorithms. One obvious problem is the slow rate of convergence of the genetic algorithm. Manifesting a behavior that is typical of genetic algorithms [7, 3], our algorithm initially makes rapid progress towards a solution, but then converges very slowly to a global optimum (or at least to a good local one). We have investigated the use of a gradient-descent methodology for rapidly converging on a locally optimal solution once an almost locally optimal solution has been generated by the genetic algorithm. The goal of this work is to develop an algorithm that will compute layouts for medium-sized diagrams in a few seconds on a regular workstation. Some preliminary results are reported in [11]; more recent developments of this theme are presented in [4].[6] Another promising direction is the development of a hybrid algorithm that uses heuristic layout rules to generate layouts with which to seed the genetic algorithm (the genetic algorithm is currently seeded with random layouts). We also hope to incorporate polyline-link routing into the algorithms described here: initial experiments have shown that the routing of polyline links can be subsumed into our genetic algorithm by including a single mutation operator that introduces link bends by creating and positioning dummy nodes.

[6] This is joint work with Ed Dengler, Mark Friedell, Peter McMurry, and Steve Sistare.

Appendix: Specification and Implementation of Visual Organization

A visual-organization specification can be stated using a set of visual organization features codified as predicates. We make use of the following notations. The set of nodes is N, and its powerset 2^N; the set of nonrepeating sequences of nodes is notated $seq(N)$. Each node is given a position in the real plane $\mathcal{R} \times \mathcal{R}$ where $\mathcal{R}$ specifies the real numbers. The set of axes is given by

$$AXIS = \{horizontal, vertical, any\} \qquad .$$

(The value *any* indicates that a particular axis is not mandated, i.e., that either the horizontal or vertical axis may be used.) The set of possible alignment positions is given by

$$POS = \{center, top, bottom, left, right, any\}$$

and the possible perimeter shapes by

$$SHAPE = \{circle, rectangle, any\} \qquad .$$

The visual-organization predicates are then defined as follows:

Proximity

$$\begin{aligned} \textit{NN-PROX} &= \{(n_i, n_j) \in N \times N \mid n_i \text{ is positioned near } n_j\} \\ \textit{NP-PROX} &= \{(n, x, y) \in N \times \mathcal{R} \times \mathcal{R} \mid \text{ the position of } n \text{ in the display reflects} \\ &\qquad \text{its location } (x, y) \text{ in the real plane}\} \\ \textit{ZONES} &= \{s \in 2^N \mid \text{ the nodes in } s \text{ are the only nodes positioned within a} \\ &\qquad \text{rectangular region of the display}\} \\ \textit{CLUSTERS} &= \{s \in 2^N \mid \text{ the nodes in } s \text{ are perceptually grouped by proximity}\} \end{aligned}$$

Sequential Layout

$$\begin{aligned} \textit{SEQLAY} &= \{(q, a) \in seq(N) \times AXIS \mid \text{the nodes in the tuple } q \text{ are positioned} \\ &\qquad \text{left-to-right or top-to-bottom along the axis specified by } a\} \\ \textit{SSEQLAY} &= \{(q, a) \in seq(N) \times AXIS \mid \text{the nodes in the tuple } q \text{ are positioned} \\ &\qquad \text{evenly spaced, left-to-right or top-to-bottom along the axis} \\ &\qquad \text{specified by } a\} \end{aligned}$$

Alignment

$$\begin{aligned} \textit{ALI} &= \{(s, a, p) \in 2^N \times AXIS \times POS \mid \text{the nodes in the set } s \text{ are aligned} \\ &\qquad \text{along the axis } a \text{ and relative to the position given by } p\} \\ \textit{SALI} &= \{(s, a, p) \in 2^N \times AXIS \times POS \mid \text{the nodes in the set } s \text{ are aligned} \\ &\qquad \text{evenly spaced along the axis } a \text{ and relative to the} \\ &\qquad \text{position given by } p\} \end{aligned}$$

Axial Symmetry

$$\begin{aligned} \textit{SYM} &= \{(s, a) \in 2^N \times AXIS \mid \text{the nodes in the set } s \text{ are symmetric about} \\ &\qquad \text{the axis } a\} \end{aligned}$$

T-Shape

$$\textit{T-SHAPE} = \{(t,s) \in N \times 2^N \mid \text{there are axes } a_1, a_2 \in \{\mathit{horizontal}, \mathit{vertical}\} \text{ such that } a_1 \neq a_2 \text{ and the nodes } \{t\} \cup s \text{ are laid out satisfying } (\{t\} \cup s, a_1) \in \mathit{SYM} \text{ and } (s, a_2, \mathit{center}) \in \mathit{SALI}\}$$

Circumferential Layout

$$\begin{aligned}
\textit{PERIMETER} &= \{(s,p) \in 2^N \times \mathit{SHAPE} \mid \text{all nodes in } s \text{ are positioned on the perimeter of the shape specified by } p\} \\
\textit{PERIMETER-CYCLE} &= \{(q,p) \in \mathit{seq}(N) \times \mathit{SHAPE} \mid \text{all nodes in } q \text{ are positioned sequentially in clockwise order on the perimeter of the shape specified by } p\} \\
\textit{HUB} &= \{(c,s,p) \in N \times 2^N \times \mathit{SHAPE} \mid \text{all nodes in } s \text{ are positioned on the perimeter of the shape specified by } p \text{, with node } c \text{ at its center}\} \\
\textit{HUB-CYCLE} &= \{(c,q,p) \in N \times \mathit{seq}(N) \times \mathit{SHAPE} \mid \text{all nodes in } q \text{ are positioned sequentially in clockwise order on the perimeter of the shape specified by } p \text{, with node } c \text{ at its center}\}
\end{aligned}$$

Because perceptual organization is not yet perfectly understood, terms like "near" and "perceptually grouped by proximity" that are used in the predicates above cannot be defined rigorously [10]. Nevertheless, these perceptual-organization concepts can be modeled approximately: for example, nearness can be modeled adequately using simple distance measures; perceptual grouping by proximity can be modeled as a function of the area and aspect ratio of the region occupied by the grouped nodes. The methods that we used for approximating degree of conformity to a visual-organization specification V, notated $c(V)$, are as follows:

- $c(\textit{NN-PROX})$
  The nodes in each pair in the visual organization specification $\{(n_{1i}, n_{2i})\}$ are to be placed near to each other. The score can therefore be computed as the sum of the distances between the paired nodes, that is,

  $$c(\textit{NN-PROX}) = \sum_i [(x_{1i} - x_{2i})^2 + (y_{1i} - y_{2i})^2]^{\frac{1}{2}}$$

  where node n_{pi} is at position (x_{pi}, y_{pi}).[7]

- $c(\textit{NP-PROX})$
  Each node n in the *NP-PROX* specification has a "desired position" (x_i, y_i), given in the visual-organization specification, and an "actual position" (a_i, b_i), which represents the node's position in the candidate diagram. First, a normalization step is performed so that the (x_i, y_i) and (a_i, b_i) have values in the range $[(0,0) \ldots (1,1)]$. The purpose of the normalization step is to avoid the inappropriate penalization of candidate diagrams that are merely scaled or translated versions of the desired diagram. Then the score is computed as the Cartesian distance of the actual position from the desired position:

  $$c(\textit{NP-PROX}) = \sum_i [(x_i - a_i)^2 + (y_i - b_i)^2]^{\frac{1}{2}}$$

[7]The current ANDD system does not implement the *NN-PROX* VOF because of its limited utility.

- $c(\mathit{ZONES})$
  First, the geometry of each zone is computed by finding the smallest rectangle that encloses all the nodes in that zone. Let *Zone_Overlaps* be the set of all pairs of intersecting rectangles and let *Zone_Invasions* be the set of all zone invasions. The pair (n, Z) is a zone invasion if node n lies within the rectangle defined by zone Z, but $n \notin Z$. Then

  $$c(\mathit{ZONES}) = |\mathit{Zone\_Overlaps}| + |\mathit{Zone\_Invasions}|$$

  (The two terms in the above formula, as well as the terms of all of the following formulas, were assigned empirically determined relative weights.)

- $c(\mathit{CLUSTERS})$
  The geometry of each cluster is computed by finding the smallest rectangle that encloses all the nodes in that cluster. Then

  $$c(\mathit{CLUSTERS}) = \sum_i \mathit{area}(\mathit{cluster}_i) + \sum_i \mathit{abs}(\mathit{width}(\mathit{cluster}_i) - \mathit{height}(\mathit{cluster}_i))$$

- $c(\mathit{SEQLAY})$
  $c(\mathit{SSEQLAY})$
  The *SEQLAY* VOF indicates that that nodes in the specification are to appear in a left-to-right or top-to-bottom order. The *SSEQLAY* VOF specifies that in addition to this, the nodes should be evenly spaced.

  The sequential-ordering term is computed by accruing a penalty if a node in the sequence is positioned to the left of (above) a node earlier in the sequence. The accrued penalty is proportional to the horizontal (vertical) distance between the offending nodes.

  To compute the even-spacing term, we first compute the spacings between each node along the horizontal (vertical) axis. Then the mode of the spacings is found. The even-spacing term is the sum of the absolute differences between the individual spacings and their mode.

- $c(\mathit{ALI})$
  $c(\mathit{SALI})$
  The *ALI* VOF indicates that the nodes in the specification are to be aligned horizontally or vertically. The *SALI* VOF specifies that in addition to this, the nodes should be evenly spaced.

  The alignment term is computed in the following way. If the VOF indicates that the nodes should be aligned along one axis, then for each pair of nodes in the VOF, a penalty accrues that is proportional to the distance between them along the opposite axis. When the nodes are aligned, all such distances will be zero.

  The even-spacing term is computed in the same manner as for *SSEQLAY*.

- $c(\mathit{SYM})$ This term measures the degree to which the nodes are symmetric. For each node in the VOF, its reflected position is computed by reflecting the node around the desired axis of symmetry. The penalty term that accrues is proportional to the Cartesian distance from that reflected position to the nearest (unreflected) node. In a symmetric diagram, all such distances will be zero.

- c(*T-SHAPE*) The *T-SHAPE* VOF is derived from *SYM* and *SALI*, as described above. The degree of conformity to *T-SHAPE* is computed by performing the corresponding computations for *SYM* and *SALI*, weighting the results appropriately, and computing their sum.

- c(*PERIMETER*)
  c(*PERIMETER-CYCLE*)
  c(*HUB*)
  c(*HUB-CYCLE*)
  The description of these terms assumes that the specified shape is a circle. The computation for the square shape is similar.

  HUB (*-CYCLE*) differs from *PERIMETER* (*-CYCLE*) in that the the center of the shape is specified to be at a given node. For *PERIMETER* (*-CYCLE*), the center is computed to be the centroid of the positions of all the nodes on the perimeter.

  For each node n_i on the perimeter, its position in polar coordinates (r_i, θ_i) relative to the center point is computed. The nodes are sorted in order of increasing θ, and the difference Δ_i between successive values of θ is computed as well.

  Penalties accrue in three ways:

  – a penalty accrues proportional to $abs(r_i - mean(r))$. Intuitively this penalizes nodes that do not lie on the circumference of the circle.
  – a penalty accrues proportional to $abs(\Delta_i - mode(\Delta))$. Intuitively this penalizes nodes that are not spaced evenly along the circumference.
  – for the *-CYCLE* forms of the specifications, a penalty accrues for nodes that appear on the perimeter that are out of sequential clockwise order.

  These penalties are weighted appropriately and summed.

References

[1] J. Bertin. *Semiology of Graphics: Diagrams, Networks, Maps.* University of Wisconsin Press, 1983. Translated by W. J. Berg.

[2] K.-F. Böhringer and F. N. Paulisch. Using constraints to achieve stability in automatic graph layout algorithms. In *CHI '90 Conference Proceedings*, pages 43–51, Seattle, Washington, 1990.

[3] L. Davis. *Handbook of Genetic Algorithms.* Van Nostrand Reinhold, New York, 1991.

[4] E. Dengler, M. Friedell, and J. Marks. Constraint-driven diagram layout. Technical Report TR-10-93, Division of Applied Sciences, Harvard University, Cambridge, Massachusetts, 1993.

[5] P. Eades and R. Tamassia. Algorithms for drawing graphs: An annotated bibliography. Technical report CS-89-09, Department of Computer Science, Brown University, 1989. Revised version.

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

[7] D. E. Goldberg. *Genetic Algorithms in Search, Optimiziation, and Machine Learning.* Addison-Wesley, Reading, Massachusetts, 1988.

[8] W. D. Hillis. *The Connection Machine.* MIT Press, Cambridge, Massachusetts, 1985.

[9] T. Kamada and S. Kawai. A general framework for visualizing abstract objects and relations. *ACM Transactions on Graphics*, 10(1):1–39, January 1991.

[10] L. Kaufman. *Sight and Mind: An Introduction to Visual Perception.* Oxford University Press, New York, 1974.

[11] S. Kochhar, M. Friedell, S. Sistare, J. Juda, M. LaPolla, J. Marks, P. McMurry, C. Kosak, and S. Shieber. A design environment for scientific and program visualizations. In *Proceedings of Compugraphics '91*, pages 321–332, Sesimbra, Portugal, September 1991. Springer-Verlag.

[12] W. Köhler. *Gestalt Psychology.* The New American Library, New York and Toronto, 1947.

[13] R. Kopeikin. ISDN professional service. *Electrical Communication*, 63(4):366–373, 1989.

[14] C. Kosak, J. Marks, and S. Shieber. A parallel genetic algorithm for network-diagram layout. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 458–465, University of California, San Diego, California, July 1991.

[15] S. M. Kosslyn. Graphics and human information processing. *Journal of the American Statistical Association*, 80(391):499–512, 1985.

[16] S. M. Kosslyn. Understanding charts and graphs. *Applied Cognitive Psychology*, 3:185–226, 1989.

[17] J. Marks. *Automating the Design of Network Diagrams.* PhD thesis, Division of Applied Sciences, Harvard University, Cambridge, Massachusetts, 1991.

[18] J. Marks. The competence of an automated graphic designer. In *Proceedings of the 1991 Long Island Conference on Artificial Intelligence and Computer Graphics*, pages 53–61, New York Institute of Technology, New York, March 1991.

[19] J. Marks. A formal specification scheme for network diagrams that faciliates automated design. *Journal of Visual Languages and Computing*, 2(4):395–414, December 1991.

[20] J. Marks and E. Reiter. Avoiding unwanted conversational implicatures in text and graphics. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI '90)*, pages 450–456, Boston, Massachusetts, 1990.

[21] A. Márkus. Experiments with genetic algorithms for displaying graphs. In *Proceedings of the 1991 IEEE Workshop on Visual Languages*, pages 62–67, Kobe, Japan, October 1991.

[22] J. Martin and C. McClure. *Diagramming Techniques for Analysts and Programmers*. Prentice-Hall, New Jersey, 1985.

[23] T. Masui. Graphic object layout with interactive genetic algorithms. In *Proceedings of the 1992 Workshop on Visual Languages*, pages 74–80, Seattle, Washington, September 1992. IEEE Computer Society.

[24] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, Berlin, 1992.

[25] R. A. O'Keefe. *The Craft of Prolog*. MIT Press, Cambridge, Massachusetts, 1990.

[26] B. Sanden. An entity-life modeling approach to the design of concurrent software. *Communications of the ACM*, 32(3):336, 1989.

[27] M. Smith, P. Briggs, and E. H. Freeman. Understanding causal feedback using the strategic planning system (SPS). In *Proceedings of the Seventh Conference on AI Applications*, Miami Beach, Florida, February 1991. IEEE Computer Sciences Press.

[28] L. Sterling and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques*. MIT Press, Cambridge, Massachusetts, 1986.

[29] K. Sugihara, K. Yamamoto, K. Takeda, and M. Inaba. Layout-by-example: A fuzzy visual language for specifying stereotypes of diagram layout. In *Proceedings of the 1992 IEEE Workshop on Visual Languages*, pages 88–94, Seattle, Washington, September 1992.

[30] R. Tamassia, G. D. Battista, and C. Batini. Automatic graph drawing and readability of diagrams. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):61–79, 1988.