# Cooperation between Interactive Actions and Automatic Drawing in a Schematic Editor

Gilles Paris
IREQ (Institut de recherche d Hydro-QuØbec)
1800, Boul. Lionel-Boulet, Varennes (QuØbec) Canada J3X 1S1
`paris@ireq.ca`

**Abstract.** We demonstrate the use of automatic drawing facilities in an interactive single-line diagram editor which is the user interface to a real-time power system simulator. As well as being used as a filter to generate ready-to-simulate schematics, the automatic drawing algorithm can cooperate with interactive editing actions to work towards obtaining a final drawing satisfying user constraints.

## 1 Introduction

We implemented in a schematic editor[1] an automatic drawing facility based on extensions to the Sugiyama-Misue (S-M) [4] algorithm for compound digraphs applied to the drawing of single-line schematics used in electrical power system applications [1, 2, 3].

Such diagrams mainly use a visibility representation which is produced by our algorithm even for non-planar graphs in the presence of edge crossings and vertex clustering.

The primary use of this facility is for generating a single-line diagram from existing EMTP[2] files. The resulting drawing is ready to be simulated as all supported electrical values in the EMTP file have been translated into the property sheets of the generated diagram elements.

The automatically obtained drawing has at first only horizontal or vertical edge direction. A bidirectional drawing may be obtained using either interactive manual editing actions combined with an automatic reconstruction of the layout, or by incrementally rotating large or small portions of the drawing automatically.

Already satisfactory portions of the drawing can also be shielded from further modification in a natural way.

We will start by describing briefly the principles underlying our extended algorithm.

## 2 Quasi-Visibility Drawing of Compound Digraphs

We call *quasi-visibility* the type of drawing obtained in our implementation because our graphs are not always *hierarchical planar* or *compound*

1. The schematic editor runs on Sun workstations under Solaris 2.5 (SunOS 5.5.1).
2. <u>E</u>lectro<u>M</u>agnetic <u>T</u>ransients <u>P</u>rogram. Standard batch simulation program.

*planar* [6] and the heuristic has to maintain the visibility representation as much as possible in the presence of bends, edge crossings and clustering.

Our basic algorithm is limited to one orientation only for edges, increasing either in the *x* or *-y* direction. Interesting new approaches to drawing in both directions have been published recently [7], but cannot be directly applied to our graphs because of their non-planarity and restrictions in the connection of edges to vertices. Our vertices must be stretched in one direction only and edges must be connected orthogonally to the stretched side.

## 3 Extensions to the S-M Algorithm

The original S-M algorithm is divided into four steps:

| | | |
|---|---|---|
| *Step I* | *Hierarchization:* | Cycle removing and compound level assignement. |
| *Step II* | *Normalization:* | Obtaining proper adjacency edges. |
| *Step III* | *Vertex ordering:* | Edge crossing reduction using the barycenter heuristic to reorder vertices. |
| *Step IV* | *Metrical layout:* | Improving vertex positioning with a variant of the barycenter heuristic. |

Most of our additions were concentrated in steps II and IV.

Our extensions may be thought as adding new attributes to the layout produced without losing previous capabilities. These are, in increasing order of complexity:

Multiple connection of edges with vertices.
Orthogonal edge drawing.
Local quasi-visibility representation.
Global quasi-visibility representation.

Multiple connection of edges (done in step IV) is easy and only implies using the crossing reduction obtained in step III when assigning a connection ordering of edges to vertex. It causes a proportional stretching of the vertex only when a visibility representation is chosen.

Orthogonal edge drawing is done with a variation of the method described in [8]. A proper algorithm for orthogonal edge drawing is necessary for a correct treatment of the bends introduced by the non-planarity of our graphs. In the planar case, only straight vertical or horizontal lines need be used.

We call *local quasi-visibility representation* the situation where vertices are stretched and edges straightened only inside a given compound level (or cluster). In this case, edges connecting vertices in different clusters are joined as in the original S-M algorithm, either orthogonally or with straight-line paths.

In the *global quasi-visibility representation*, recursive application of the same algorithm to each cluster has to be forgone for a global treatment of edges and vertices at all compound levels, in order to stretch vertices as if enclosing levels of structure were transparent.

For these representations, the definition of proper adjacency edge is modified (in step II) so as to reduce the number of compound hidden (or dummy) nodes to only these cases where vertices are far apart in some sense. When no clustering is done, local and global quasi-visibility produce the same drawing.

We can still produce with our implementation the initial layout obtained with the Sugiyama-Misue algorithm, so that a direct comparison of the aesthetic choices made in each case is possible (see Fig. 6)

The main idea behind the method used to obtain quasi-visibility is to allow multiple connections of a vertex to be considered as freely moving *connection nodes* and to apply (again) a barycenter method to let them align themselves as much as possible, even in the presence of crossings.

The alignement is done by limiting barycentric iterations to only movements toward the right. When a crossing is encountered, connection nodes further to the left stop moving while those to the right of the crossing (on the alternate level) keep on moving to the right. The restricted movement produces a drawing which is not symmetrical compared to the initial S-M layout, but this is not perceived as a problem in our type of drawings.

## 4 A Recursive Structure of Recursive Structures

The S-M algorithm used as a basis for our method already features clustering of the graph by the use of a tree structure on top of the graph representation.

We introduce another level of tree structuring in the definition of our bidirectional drawings. Each portion of a drawing going in the opposite direction (vertical or horizontal) is extracted as a full feature compound digraph and added as a child to the parent graph. The drawing of bended bi-edges which are extracted edges going to or from a parent graph to a child graph is done by a new specialized method, while the normal extended S-M algorithm is retained to recursively draw each child graph in its orthogonal direction.

## 5 Constraints for Satisfactory Single-Line Diagrams

The desired representation for complex electrical single-line diagrams varies with individual taste, and also often embodies electrical symmetries and underlying geographical positioning which are difficult to attain with completely automatic generation, even with a constraint grammar. Refer to Fig. 1 for an example of a large final drawing.

For this reason, our current approach has been to make both interactive and automatic actions cooperate as naturally as possible to obtain the final refined drawing from an automatically generated first draft.

## 6 Editing with the Automatic Drawing Menu

One approach to editing the drawing is using automatic drawing operations on selected elements. The user makes a selection, applies the operation and the drawing is updated. The user may undo the operation if needed or go on with other actions until the desired goal is attained, or forego the automatic drawing process and continue with manual editing operations.

The automatic drawing *diagram* menu is shown in Fig. 2. Possible operations, or groups of operations, which may be also invoked by a single keystroke, are:

*Fix and reposition / Update / Undo*

Automatic redrawing with portions left untouched and automatic rotation applied if needed. The previous representation may be reloaded.

*EMTP file*

Generate a drawing from an EMTP file selected with a file chooser. The filter program is automatically invoked.

*Orthogonal branches / Oblique branches*

Use either manhattan style edges or straight lines.

*Enclose / Separate*

Clustering or unclustering of selected elements.

*Implode / Explode*

Collapse/expand selected elements into a small square keeping all edges connected.

*Fix / Free*

Freeze/unfreeze current layout of selected elements, protecting them from further automatic redrawing.

*Permute*

Select a pair of elements and permute their barycentric ordering. This action is useful for eliminating undetected crossings, or choosing from barycentric equivalent ones. The relative order of permuted elements is kept until the next Fix and reposition.

*Rotate normal / Rotate inverted / Fuse back*

Generate a rotated child graph keeping either the normal *top->bottom* or *left->right* direction of edges, or inverting it. If the parent has vertical edges, the child will have horizontal ones and vice-versa. A child graph may be fused back with its parent graph to undo this operation.

*Show model / Stretch bus / Vertical / Compact / Show structure*

These are toggles for the drawing operation. The model used internally by the algorithm may be shown as enclosing boxes for validation. Buses may be stretched (visibility representation) or not (similar to the S-M representation). A last minute bidirectional compaction step may be used or not. This step violates the normal compound level hierarchy and some undetected collision of vertices with edges may appear. The recursive structuring of parent/child graph, clustering and fixed portions may be shown with differently colored enclosing rectangles.

## 7 Editing with the Standard Graphic Operations

With this alternative approach, the user concentrates only on standard graphic editing operations from the *drawing palette*, shown in Fig 3.

The edge orientation of the topmost parent graph is important for correct results and must be toggled appropriately prior to any editing.

The user then applies any interactive action from the *drawing palette* to the drawing, such as rotating single elements, flipping or stretching them as desired, and of course any non-critical graphic action such as adding color or text attributes.

If a portion of the drawing is already satisfactory, the user then selects and groups those elements (to any recursive level if needed) in order to shield it from any further modification by the automatic repositionning.

At any time, applying the *Fix and reposition* action is followed by the following events:

The list of graphic objects in the editor is scanned and candidate child graphs are found. Those are the connected objects having opposite edge orientation from the parent.

Grouped objects are interpreted as fixed elements and not scanned.

The relative positions of connectors between child and parent elements are used to infer the desired orientation of child graph edges.

A unidirectional repositioning of the complete drawing is then generated as a topmost parent graph, and automatic rotation actions are done to generate all child graphs.

This method generates only 2-levels hierarchies of graphs while any degree of recursive embedding may be done using the *diagram* menu. Such an existing embedding would then be reduced to two levels if the *drawing palette* technique is used afterwards, giving rise to a slightly different result.

Of course, shielded fixed portions cannot always be aesthetically incorporated in the complete drawing since their content is untouched by the positionning algorithm, but connectivity is preserved and no overlapping occurs.

Further iterations of manual editing and regrouping of satisfactory portions of the drawing brings the user closer to the desired goal.

Fig 4. shows part of an editing session using the *diagram* menu. Fig 5. shows a result of using only *drawing palette* actions.

## 8 Automatic Bidirectional Drawing

We have also recently added an automatic clustering facility along the lines of [5] in order to produce a bidirectional drawing by using the ratio cut method recursively down to a specified compound level. The revealed clustering is then drawn as a tree of orthogonally oriented child graphs.

This is useful in order to produce a drawing with a more pleasing aspect ratio and also to emphasize structure already present in the graph specification. A drawback is the addition of bends to the drawing.

## 9 Conclusion

The use of the automatic drawing facility saves a lot of time at the start of power system studies when an EMTP file is available.

Using the algorithm during editing may be of help to some while others may want to do everything manually, but its presence is transparent in the editor and the desired drawing may be more easily obtained with it.

## 10 Future Work

We already have the capability to implement user constraints for relative up-down or right-left positionning for clusters or vertices. There remains to define a grammar for specifying such constraints to control automatic generation.

## References

1. N. De Guise, G. Paris, M. Rochefort, IREQ: Extending a Real-Time Power System Simulator s Graphical User Interface. *Presented at ICDS 97, second International Conference on Digital power system Simulators.*
2. G. Paris: Automatic Drawing of Compound Digraphs for a Real-Time Power System Simulator. *Proc. 4th Symposium on Graph Drawing (GD 95), LNCS 1027, Springer-Verlag.*
3. M. Rochefort, N. De Guise, L. Gingras, IREQ: Development of a graphical user interface for a real-time power system simulator. *Presented at ICDS 95, first International Conference on Digital power system Simulators.*

4. K. Sugiyama, K. Misue: Visualization of Structural Information: Automatic Drawing of Compound Digraphs. *IEEE Transactions on Systems, Man and Cybernetics, Vol 21, No. 4, July/August 1991.*
5. T. Roxborough, A. Sen: Graph Clustering Using Multiway Ratio Cut. *Proc. 5th Symposium on Graph Drawing (GD 97), LNCS 1353, Springer-Verlag.*
6. P. Eades, Q-W Feng, X Lin: Straight-Line Drawing Algorithms for Hierarchical Graphs and Clustered Graphs. *Proc. 5th Symposium on Graph Drawing (GD 96), LNCS, Springer-Verlag.*
7. U. F βmeier, G. Kant, M. Kaufmann: 2-Visibility Drawings of Planar Graphs. *Proc. 5th Symposium on Graph Drawing (GD 96), LNCS, Springer-Verlag.*
8. G. Sander: A Fast Heuristic for Hierarchical Manhattan Layout. *Proc. 4th Symposium on Graph Drawing (GD 95), LNCS 1027, Springer-Verlag.*

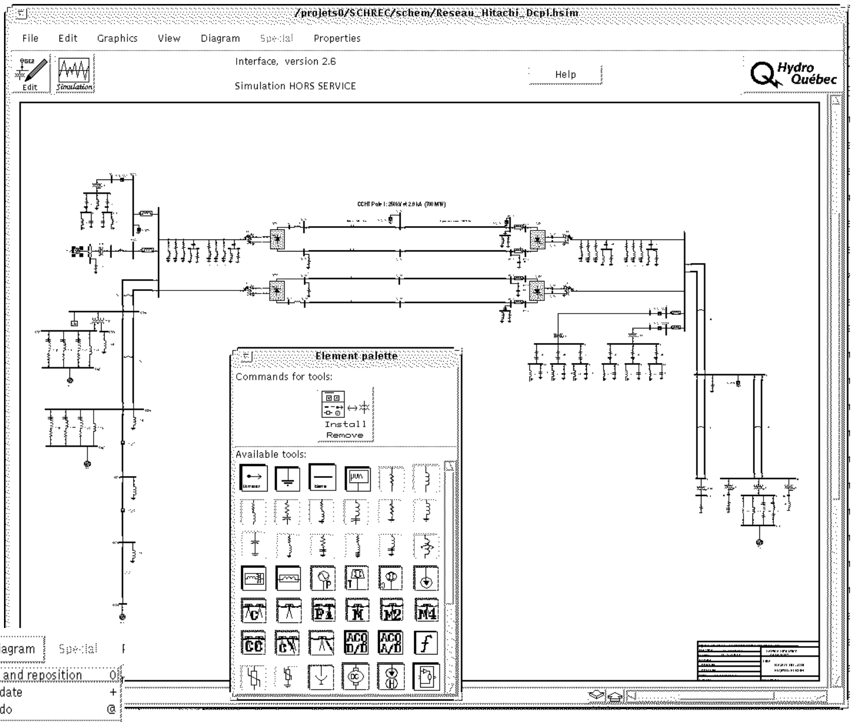


Fig. 1 - A large single-line diagram.

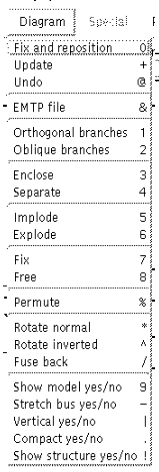


Fig. 2 - The diagram menu.

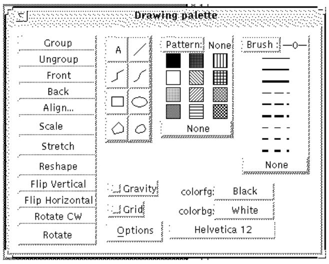
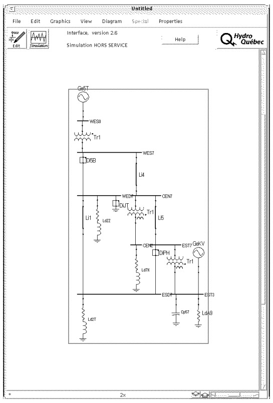
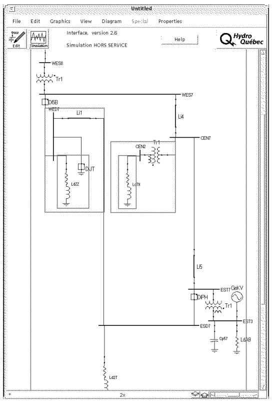


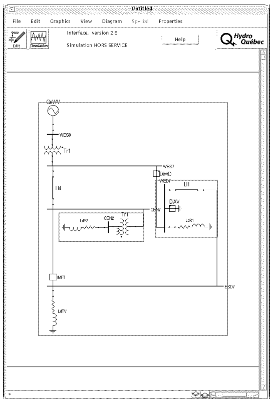
Fig. 3 - The drawing palette.



a)



b)



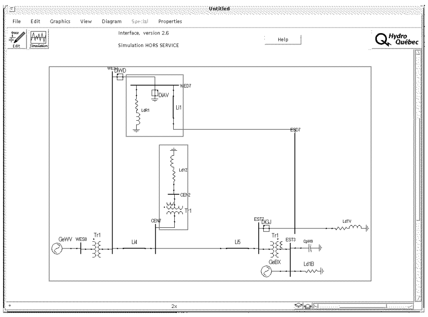
c)

Fig. 4 - Some actions using the menu.

a) Original graph generated from EMTP file.

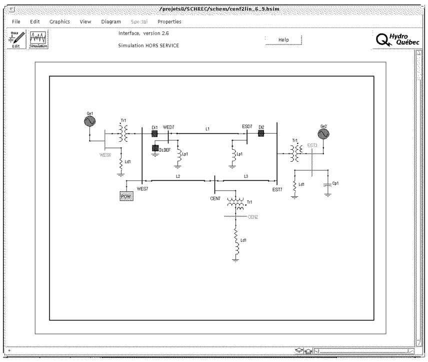
b) After extracting some child graphs.

c) A different case, after imploding part of the parent graph.

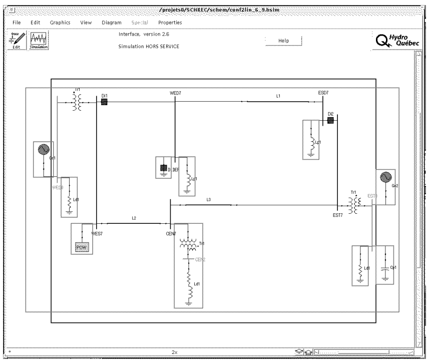


d)

d) Changing edge orientation of parent; note conservation of initial orientation of c) for children; this action is not a direct $90^{o}$ rotation.



e)



f)

Fig. 5 - Result of automatic redrawing of manually constructed diagram.

e) Original hand-created drawing.
f) Automatic redrawing complemented by some permutations; the original fixed size border has been kept to show the small increase in area.

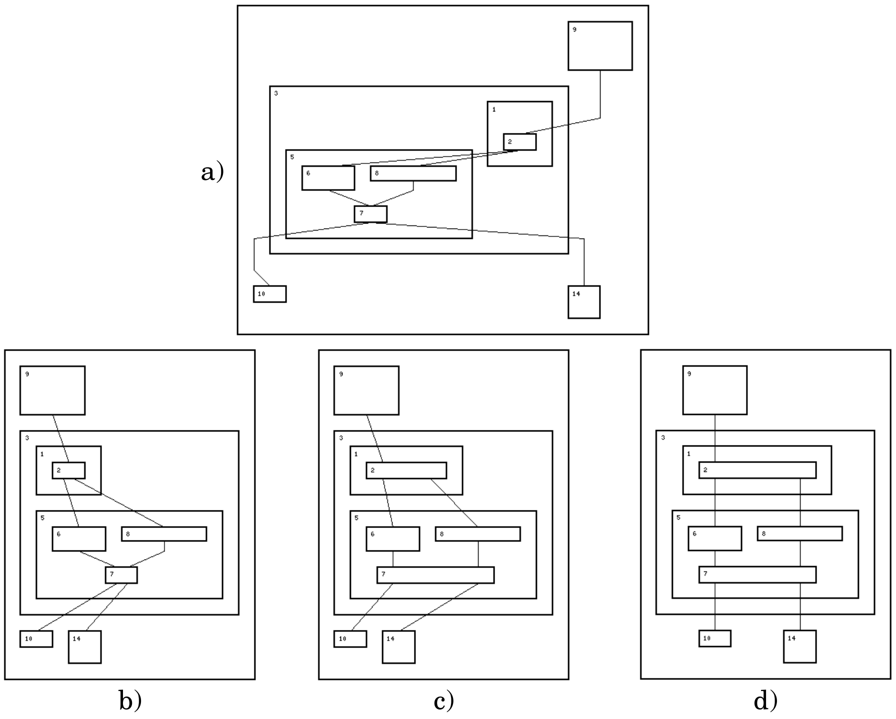


Fig. 6 - A comparison of aesthetics.

a) Original S-M layout,
b) with only multiple connections, c) local quasi-visibility, d) global quasi-visibility.