Automatic Drawing of Compound Digraphs for a Real-Time Power System Simulator

Gilles Paris
IREQ (Institut de recherche d'Hydro-Québec)
1800, montée Ste-Julie, Varennes (Québec) Canada J3X 1S1

Abstract. Two implemented versions of a compound digraph algorithm using an object-oriented approach are shown for use in the field of automatic drawing of electrical circuit diagrams. One is a test version with minimal user interface, the other a full-fledged editor part of a graphical user interface project currently under development. Orthogonal drawing of edges and interactive implosion and explosion of compound vertices are key features of the application.

1 Introduction

This demonstration shows two implementations of the algorithm published in [1] for the automatic drawing of *compound digraphs* (directed graphs that have both *inclusive* and *adjacent* relations among vertices). This algorithm is to be included in IGSIM [2], a graphical user interface for IREQ's real-time power system simulator. The work presented here is still preliminary and will evolve along with IGSIM.

2 Application Context and Justification

2.1 The Hybrid Real-Time Power System Simulator

The existing simulator is constructed with both digital and analog components, and so has hardware control panels, but the next version, still a prototype, will be fully digital, and will rely mostly on software control panels and graphical user interface.

Although well equipped with control, database and plotting software, the simulator is lacking a schematic-based user interface, so that general-purpose text-processing software has to be used to produce diagrams as documentation only.

2.2 Single-Line Electrical Diagrams

The main type of diagram used in power studies is the single-line schematic diagram where vertices are either electrical symbols or stretchable bars, while edges are usually represented orthogonally. More general electronic circuit schematics may also be used at a lower level, but they are not considered as part of the automatic drawing, and are represented as a fixed-content compound vertex. This facility is supported but not yet sufficiently developed to be discussed further.

2.3 Other Types of Diagrams

Block diagrams are used frequently in power system analysis, so representations of a type similar to [1] are also possible with this automatic drawing facility. The implementation currently is in one of two modes: *block diagram* or *single-line diagram*.

2.4 Use of Circuit Analysis Programs

An important part of power system design is done with highly specialized software tools such as the ElectroMagnetic Transients Program (EMTP), MatLab, and Stability Study Program, some of which have low-level interface, so there is a need for an automatically generated schematic derived from the many existing input files.

2.5 Why Directed Graphs?

Electrical circuits are undirected by nature, but there is a tendency, because of the topographical characteristics of our network, to order components from generating to consuming centers. This feature may be used to specify the layout in a mostly directed way. Remaining cycles in the graph are automatically resolved by the algorithm itself.

2.6 High Performance Requirements

Any interactive change to a compound digraph forces a total redrawing, so care was taken to implement the algorithm efficiently, using inline functions, pointers and integer arithmetic as much as possible, as well as minimizing structure traversals and recomputations. Iteration limits may also be set as compromise between speed and readability.

3 Reusable Object-Oriented Implementation Using C++

3.1 Independence of Graphical User Interface Development

One goal of this approach was to separate this work from IGSIM development.

3.2 Independence of Drawing Primitives

An alternative goal was to isolate in a separate C++ class the drawing primitives, so that specific classes may be derived for integration in any program written in C++. This is shown by the two different versions of the algorithm in this demonstration.

3.3 Independence of Input Formats

A third goal was to remain independent of any specific input format for graph description. This goal was met by designing a simple tree language using overloaded C++ operators. This feature, along with C++ stream operators, means that a graph (or compound vertex) specification may reside in memory, be directly coded as an expression in the application, or be input from a file.

4 Extensions to the Original Algorithm

The direction of edges may be vertical or horizontal. In the following discussion, vertical direction is assumed. The metrical layout portion of [1] has been modified as follows to support the single-line style.

4.1 Separation of Multiple Input and Output Connections to Vertices

In single-line mode, a vertex having multiple input and/or output edges connected to it is stretched horizontally and edges are ordered according to the x position of connected vertices.

4.2 Orthogonal Edge Drawing

Oblique edges generated by [1] are drawn with three orthogonal segments. A separate horizontal corridor is allocated for each vertex at the same compound level and edges connected to a vertex are laid out in it. This also stretches the drawing vertically.

5 The First Version: An RPC-Based Test Implementation

This version draws directly on an event-driven canvas and receives interactive commands from a separate process using Remote Procedure Calls (RPC). Vertices are drawn as simple labeled rectangles. Positions are in pixels.

5.1 Goal

The goal was to have an independent test-bench for the algorithm along with a method of exercising interactive operations without having to write a proper user interface. Fast unstructured graphics isolate the performance of the algorithm from the drawing itself and thus ease subjective performance evaluation with interactive operations.

5.2 Interactive Commands

Commands consist of a single letter followed by arguments which may be simple integers, strings, or arbitrary length constructor expressions of the input language. Possibilities are *creation/destruction* of lists of edges and of sub-trees or single vertices, *implosion/explosion* and *visibility* of compound vertices, *structural changes* of the inclusion relation, *direction* and *mode* of drawing. A number of *built-in test graphs* may also be constructed by invoking the corresponding function name.

Commands may be used to change the size and labeling of rectangles, vary iteration limits and distances separating rectangles in and between compound levels.

6 The Second Version: Integration with an Unidraw-Based Editor

6.1 A Snapshot of Work in Progress for IGSIM

This editor is based on Unidraw [3]. Because IGSIM is undergoing changes, a snapshot version was taken for this implementation, and does not include control panels. It is a generic schematic editor.

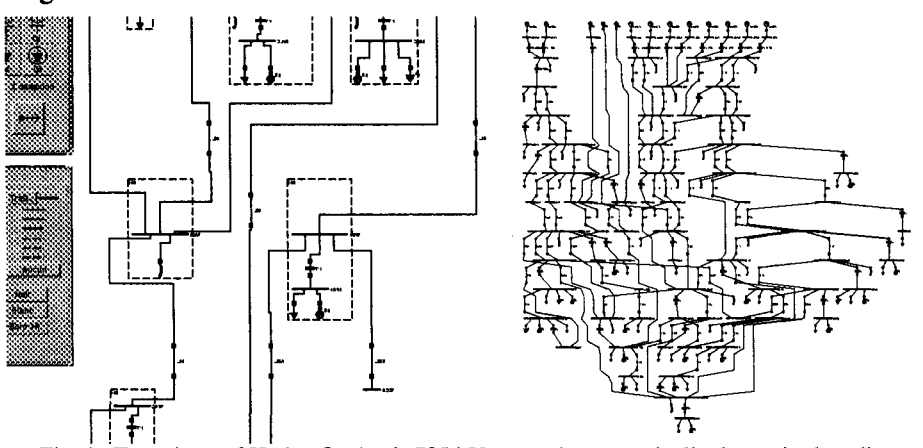

Fig. 1. Two views of Hydro-Québec's 735 kV network automatically drawn in the editor

6.2 Integration of Automatic Drawing Classes

All classes implementing the automatic drawing algorithm were first included without changing the editor. Code was then added to by the author to reimplement the drawing class by repositioning or modifying elements created with the editor.

A special menu was added to control the automatic drawing function. These exercise the same functions as the RPC-based version, except that selection is done directly on the drawing instead of referencing elements by their number.

Edges which are conceptually made of multiple line segments have to be reimplemented as a single multiline in the editor. Unidraw's concept of a generic multiline modified by a transformation matrix poses some distortion problems with orthogonal edge redrawing when moving connected objects. A temporary fix was included.

6.3 Structured Versus Unstructured Graphics

The use of hierarchical structured graphics is essential for picking objects, and Unidraw's implementation pays great attention to optimization of redrawing, but it is quite apparent that a loss of performance appears when repositioning large graphs.

7 Further Work

7.1 Filter Programs

A filter program transforming EMTP files in the input language is the next step.

7.2 Better Orthogonal Routing of Edges

Orthogonal drawing of edges adds too much vertical stretch to the drawing. Better use of each corridor must be made without impairing performance. The barycenter heuristic also creates ambiguities in some vertical paths for complex graphs by the alignment of unrelated vertices. These ambiguities do not appear with oblique edges.

7.3 Bidirectional Edge Drawing

A more difficult task is automatic drawing of edges in both directions. A simple way to approach this is by having the direction local to a compound vertex, but direction changing poses non-trivial problems which must first be addressed theoretically.

8 Conclusion

Although incomplete, this work is a first step in introducing automatic drawing techniques in the field of electrical network diagrams at IREQ.

References

1. Kozo Sugiyama and Kazuo Misue: Visualization of Structural Information: Automatic Drawing of Compound Digraphs. IEEE Transactions on Systems, Man and Cybernetics, Vol 21, No. 4, July/August 1991.
2. Marie Rochefort, Nathalie De Guise and Luc 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.
3. John M. Vlissides, Mark A. Linton: Unidraw: A FrameWork for Building Domain-Specific Graphical Editors, Stanford University, 1989.