ELSEVIER

Operations Research Letters 17 (1995) 89–101

operations research letters

NETPAD: An interactive graphics system for network modeling and optimization

Nathaniel Dean, Monika Mevenkamp, Clyde L. Monma *

Bell Communications Research, 445 South Street, Morristown, NJ 07960-1910, USA

Received 1 February 1992; revised 1 December 1994

Abstract

The practical and theoretical importance of network models and algorithms is clearly documented in the literature. This has resulted in several recent efforts to develop systems for network modeling, algorithms and/or visualization. The main goal of this paper is to describe NETPAD, an interactive graphics system for network modeling and optimization. There were several factors motivating us while developing this system. First, networks are inherently visual; so an interactive graphics interface was considered to be a vital component of the overall design. Second, data form a very important part of network modeling; therefore, we have integrated network attributes and tables into the system. Third, algorithmic needs change over time to meet users' needs for additional functionality or performance, and to meet the needs of specific applications; so we have designed the system to be customizable and expandable. Fourth, widespread use of sophisticated methods requires ease-of-use in addition to functionality; so the system includes a menu-driven user interface, standard file formats and algorithm animation. NETPAD is a portable system written in the C programming language for workstations with the UNIX operating system and the X Window System.

Keywords: Network optimization; Graphical user interface (GUI); Software; Algorithms

1. Overview

Networks are useful for modeling many practical situations, including physical networks such as the ones representing communications or transportation networks, as well as abstract networks such as those representing the scheduling of events or the allocation of resources. At the same time, research in graph theory and network algorithms have provided a wealth of tools for network analysis and optimization. The practical and theoretical importance of network models and algorithms is clearly documented in the literature.

There have been several recent efforts to develop systems for network modeling, algorithms and/or visualization. These efforts represent attempts to harness the considerable power of the available network technology into a system which is easy-to-use and meets the needs of certain communities of users. This has resulted in a proliferation of special-purpose systems and individual libraries of network algorithms.

The main goal of this paper is to describe NETPAD, an interactive graphics system for network modeling

* Corresponding author.

SSDI 0167-6377(95)00010-0

and optimization. There were several factors motivating us while developing this system. First, networks are inherently visual; so an interactive graphics interface was considered to be a vital component of the overall design. Second, data form a very important part of network modeling; therefore, we have integrated network attributes and tables into the system. Third, algorithmic needs change over time to meet users' needs for additional functionality or performance, and to meet the needs of specific applications; so we have designed the system to be customizable and expandable. Fourth, widespread use of sophisticated methods requires ease-of-use in addition to functionality; so the system includes a menu-driven user interface, standard file formats and algorithm animation.

NETPAD is an interactive graphics software system which provides an integrated environment for network modeling and optimization with features including:

- graphics for creating, displaying and manipulating networks,
- tables for entering, displaying and manipulating data,
- standard formats for network files,
- expandable library of network algorithms,
- facilities for algorithm animation,

At one level, NETPAD functions like an "electronic pencil and notepad" using a mouse, menus and multiple windows to create, manipulate and save networks; it can also be used to obtain Postscript printer output. On another level, it functions like an elaborate "network calculator" for applying available functions and algorithms to process networks. It also functions like a "network workbench and toolkit" using a library of network algorithms which can be customized and expanded to provide for rapid prototyping for specific applications. This functionality results in an easy-to-use vehicle for harnessing the power of available network modeling and algorithmic tools.

We use the term "graph" to represent a set of nodes together with a set of links, where each link consists of a pair of nodes. We use the term "attribute" to represent data values associated with the graph itself, its nodes or its links. Each (graph, node or link) attribute has a name, a data type (e.g., character, integer, float) associated with it, and data values (for each graph, node or link, respectively). We use the term "network" to represent a graph together with any number (possibly none) of types of attributes.

2. NETPAD environment

NETPAD is an interactive graphics software system which provides an integrated environment for network modeling and optimization. It is a portable system written in the C programming language for workstations with the UNIX operating system and the X window system; it is currently being used at Bellcore on SUN and DEC workstations and on 486-based IBM-compatible personal computers. It consists of an interactive graphics, menu-driven user interface which can be easily customized to fit specific users or applications. It also consists of library of network algorithms which can be easily expanded to include new or existing algorithms.

NETPAD utilizes a mouse-oriented, menu-driven user interface to provide for maximum ease-of-use for most users. Experienced users can also make use of keyboard equivalents for all menu items in order to operate more quickly once familiarity with the system is gained. The interface is further enhanced by allowing the user to easily customize most aspects of the system, including the menus themselves, items within a menu, and keyboard equivalents, to provide a "look-and-feel" to fit a particular user or application.

NETPAD allows multiple windows to be present at any time. Operations selected from a given window are applied to the network currently associated with that window. A window consists of the four main components, the status area, menu area, button area and display area. Examples of typical windows are shown in Fig. 1.

The status area is for displaying text information about current algorithm including the name of the algorithm being executed, its type (e.g., internal or external) and its status (e.g., done). The menu area displays the heading titles for the menus. The user uses the mouse to pop-up a menu in order to select an item to execute next. The button area is used to execute or kill a selected menu item, and to decide whether

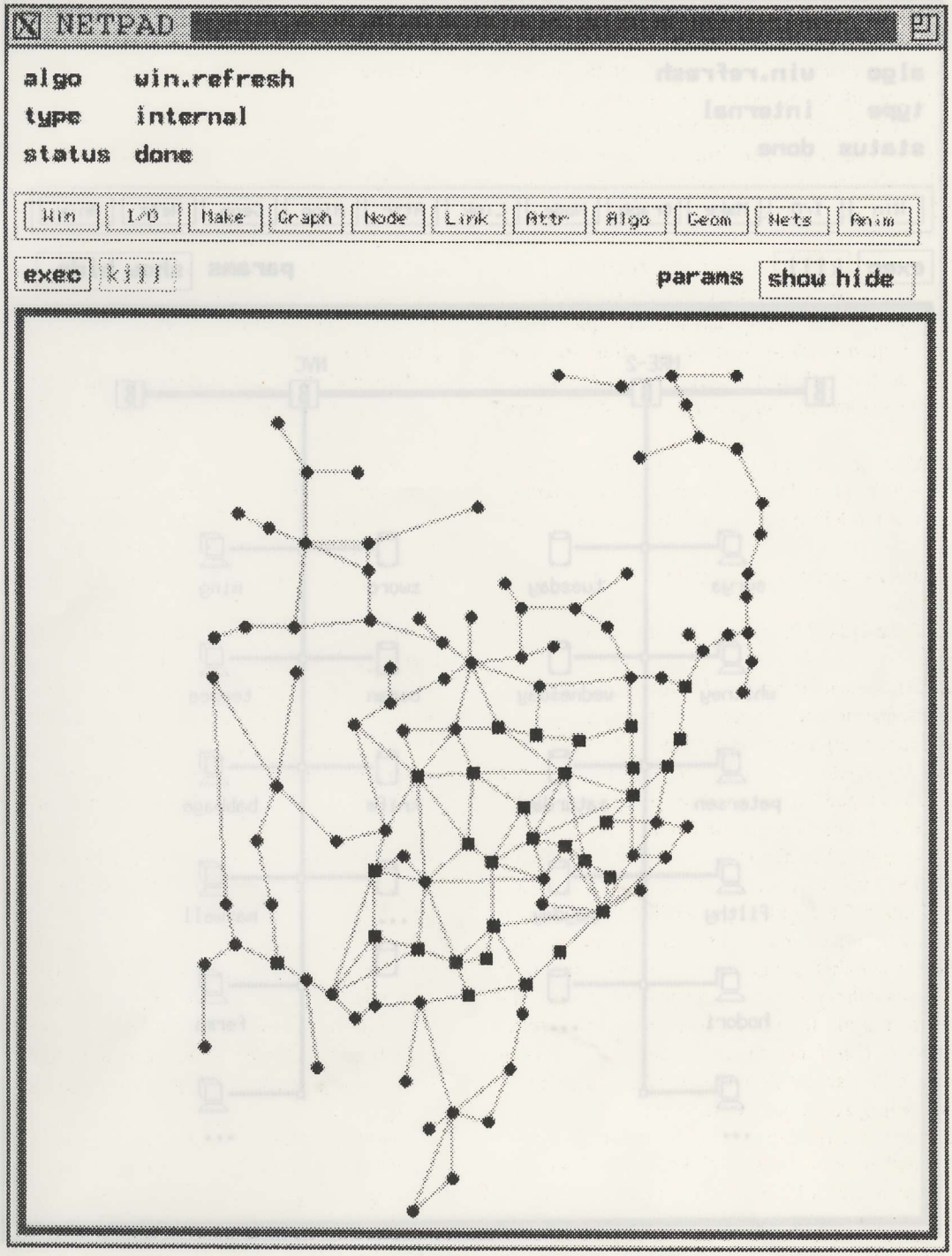

Fig. 1. Examples of typical windows.

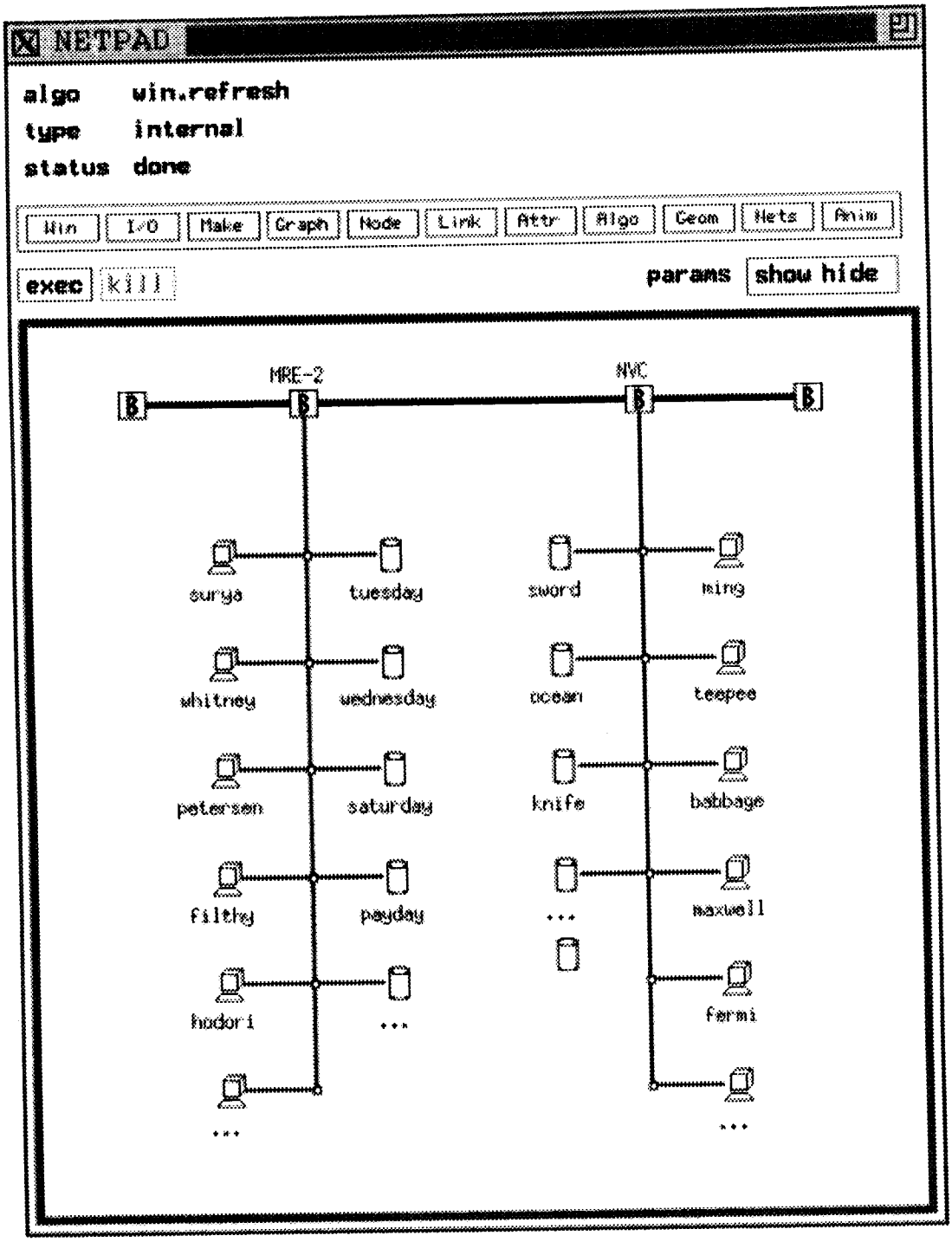

Fig. 1. (Continued.)

to show or hide parameters associated with an algorithm. The display area is for graphically displaying the network.

The number and names of the menu headings and the particular items within a menu are governed by a configuration file. NETPAD reads this file initially to set up the user's environment. The format of the configuration file consists of an entry for each menu starting with the keyword "MENU:" followed by the name of the menu (in quotes), and a list of menu items in brackets. Each menu item includes the name (in quotes), the associated executable file in the NETPAD library, and the keyboard equivalent (in quotes). A sample configuration file is shown in Table 1.

This particular configuration file would generate the menu areas for the windows shown in Fig. 1. Several of the menus are for functions normally associated with the NETPAD interface. For example, the Win menu entries are for window management, like opening a new window or quitting the current window; and the I/O window is for input/output functions like loading a network from a file, saving the current network to a file or printing the current network. The Graf, Node and Link menus are for performing operations on the entire graph, on its nodes or on its links, respectively. Typical operations include cut-and-paste; selection of nodes and/or links; operations on selected elements (e.g., deletion, change color/shape/style); changing defaults for node/link colors, node shapes (e.g., point, box, etc), and link styles (solid, dashed, directed). The Attr menu is used for handling attributes, including defining new attributes and setting attribute values. The Anim menu contains animation functions which are part for initializing and playing back an animation file in several modes (e.g., run, step, next).

The other menu items (we show here the Make, Algo, Geom and Nets menus) would normally be customized menus which could contain functions from the NETPAD library (called internal algorithms) or from a separate user program library (called external algorithms); these would be grouped into categories according to the user's preference as specified in the configuration file definition. There are many more internal and external algorithms which are omitted from discussion here for the sake of clarity.

Table 1
Example configuration file

```
MENU: "Win" {
"Open"                win.open            "wo"
"Quit"                win.quit            "wk"
}
MENU: "I/O" {
"Load"                graf.load           "GL"
"Save"                graf.save           "GS"
"Print"               graf.print          "GP"
}
MENU: "Make" {
}
MENU: "Graf" {
"Select"              graf.select         "Gs"
"Cut"                 graf.cut            "Gy"
"Paste"               graf.paste          "Gp"
}
MENU: "Node" {
"Select"              nodes.select        "Ns"
"Delete Sel"          sel.node.delete     "nd"
"Color "              nodes.color         "Nc"
"Shape "              nodes.shape         "Nt"
}
MENU: "Link" {
"Select"              links.select        "Ls"
"Delete Sel"          sel.link.delete     "ld"
"Color "              links.color         "Lc"
"Style "              links.style         "Lt"
}
MENU: "Attr" {
"Define Attr"         attr.define         "Ad"
"Set Attr Val"        attr.set            "As"
}
MENU: "Algo" {
"Planar draw"         planar.alg          "Pl"
"Path Finder"         path.alg            "Pf"
}
MENU: "Geom" {
}
MENU: "Nets" {
}
MENU: "Anim" {
}
```

3. Using NETPAD

To further clarify how the interface is used, we consider some specific examples. Selecting "Load" item in the "I/O" menu has the effect shown in the first entry in Fig. 2; note that a parameter box appears to request the name of the file to load (in this case, planar.grf) and the network is now drawn in the display area with

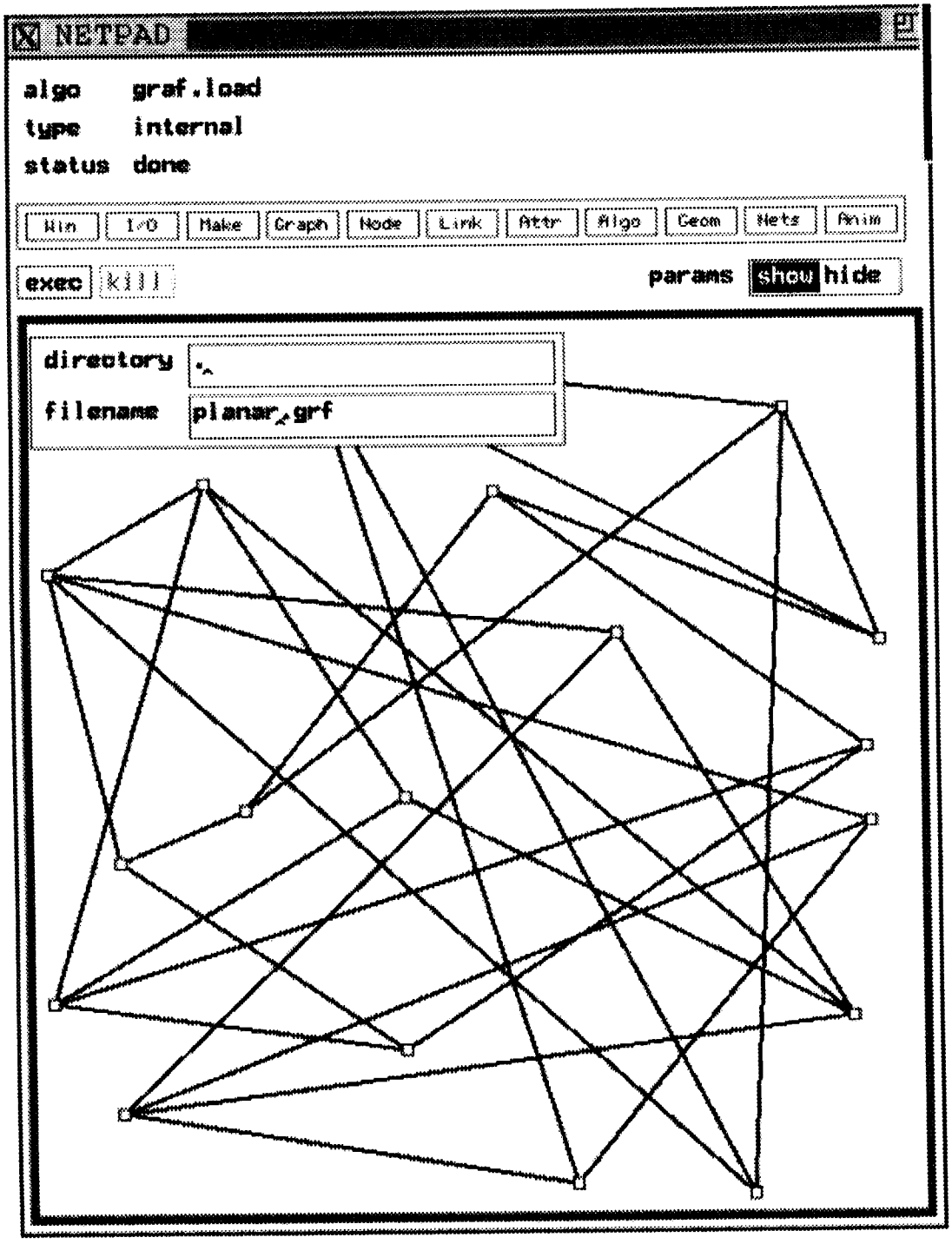

Fig. 2. Examples of menu interactions.

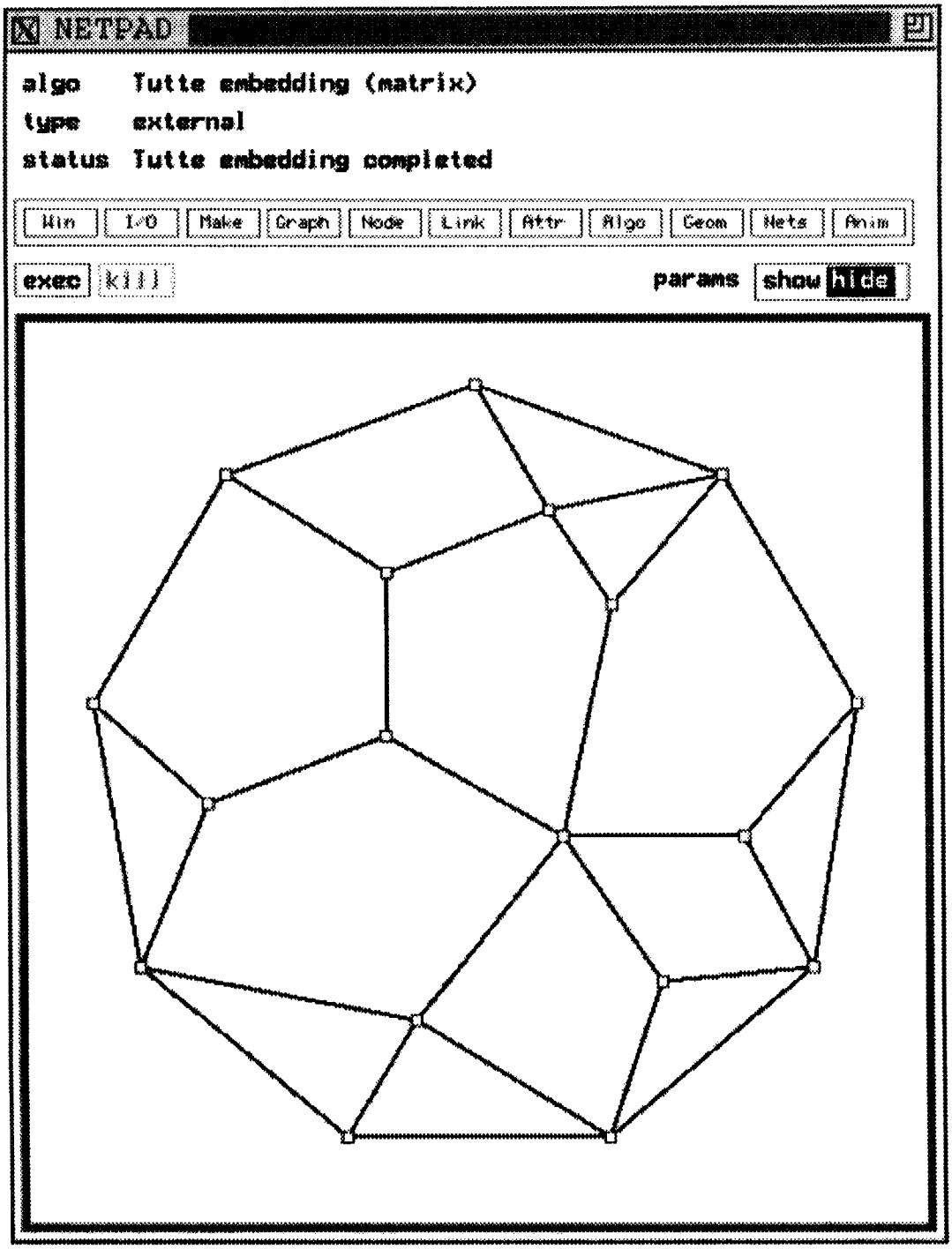

Fig. 2. (Continued.)

the information updated in the status area (namely, that the internal algorithm graf.load was completed). We could now select the "planar draw" item from the Algo menu to obtain a planar layout of the graph as shown in the second entry of Fig. 2.

To further illustrate the notion of attributes, consider Fig. 3 which shows a network with nodes labeled by city names and links labeled by distances. A network can have any number of attributes associated with the graph, nodes or links. Any one of these attributes can be chosen to be displayed as a node or link label. The position of a label can be chosen by selecting a clock position around a node and by selecting the percentage distance along a link. Fig. 3 shows the result of executing the "Path Finder" algorithm from the Algo menu on this network: note that the shortest path between the selected cities of Phoenix and St. Paul is highlighted and the status area shows the path length. We note that algorithms which provide more detailed outputs can make use of an output box similar to the parameter box used for inputs to algorithms which we described previously. Attribute values can be modified for individual nodes or links by using the Node or Link menus, respectively, or for all nodes or links by using the Attr menu to obtain an attribute table as shown in Fig. 4; an attribute table displays all node or link attributes and their values in a tabular format and allows for quickly loading, modifying and saving their values.

4. Customizing NETPAD

We have already explained how to customize the user interface using the configuration file. We now describe how to further customize the system by adding user-defined programs, called external algorithms. External algorithms are written as main programs which are linked to the NETPAD library and are separate programs residing in their own executable files. These algorithms access and manipulate networks by using functions provided by the NETPAD library. Each external algorithm has an algorithm specification file to identify its input requirements, or parameters, including their names, types and default values. This file is used by the NETPAD kernel so that the interface can automatically execute the algorithm and provide the required parameter boxes.

Each source code file for an external algorithm must begin by including the NETPAD file containing all definitions of the NETPAD data types, constants and subroutines. The internal structure of the NETPAD data objects is hidden from the user for several reasons: (1) the user is not burdened with mastering them, (2) the user is protected from accidental modification, and (3) the system is protected from malicious modification. Pointers to these objects are used to communicate between programs and the system.

There are three basic data types: networks, nodes and edges. An external algorithm accesses the data via function calls. These functions comprise the programmer interface and are the only means by which an external algorithm may interact with the system. To execute a function, the programmer generally passes a pointer to one of the three basic NETPAD data objects as input. The following paradigm is typical for external algorithms.

1. Accept the current network and parameters as input.
2. Use function calls to obtain the attribute values.
3. Place the data into private data structures.
4. Compute results using these private data structures.
5. Prepare the output network, attributes and results.
6. Return the outputs to the system.
7. Exit.

Of course, it is not essential that the algorithm perform all of these steps.

A programmer need not write, modify or re-compile any existing files in order to add a new program to the system. The user simply compiles the new program and links it with the system. Assuming that the user has a C program called pgm.c, for example, one simply needs to create a file called pgm.alg describing the parameters used as input to the program and add a line to the customization file so that the program will appear as a selection on the desired menu.

NETPAD automatically passes the current network and (if necessary) further parameters to algorithms. It allows the algorithms to return a modified network or to create and return a completely new network. Animation facilities are provided for user-defined programs, thus supporting the visualization and analysis

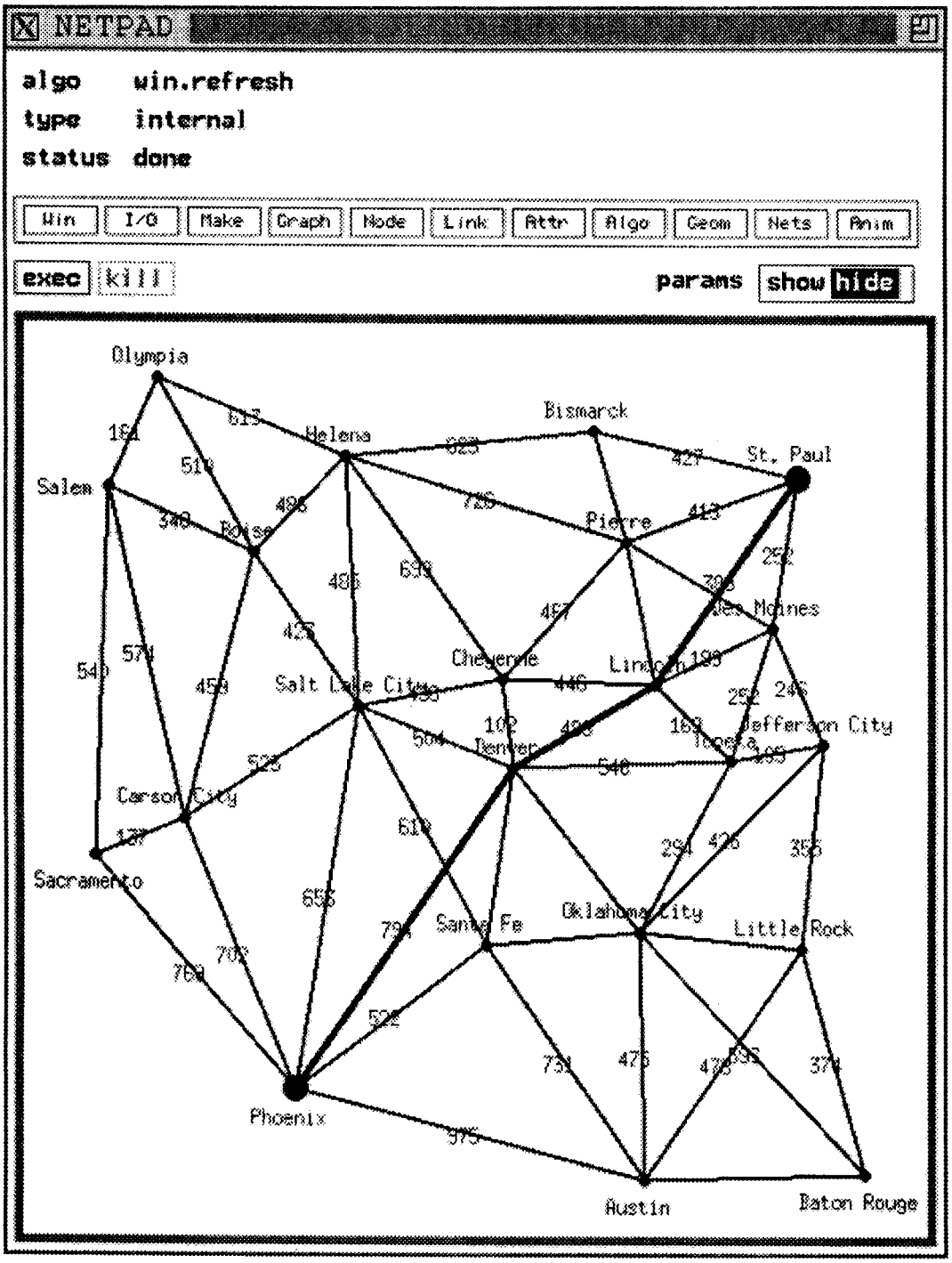

Fig. 3. Example of algorithm with attributes.

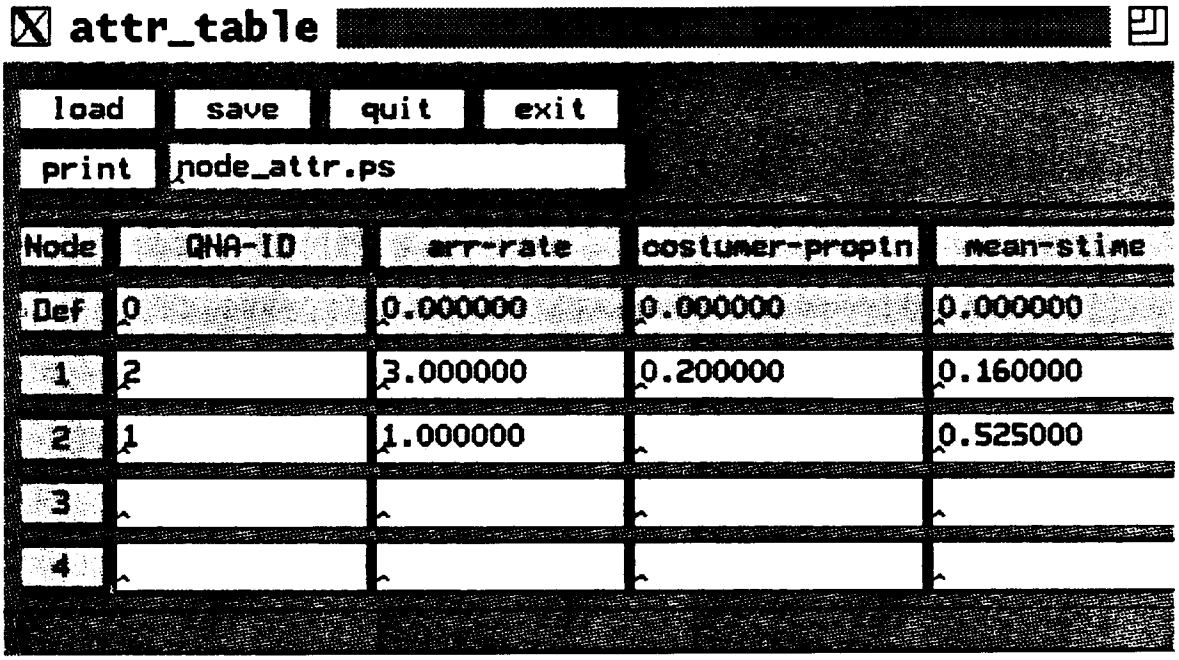

Fig. 4. Example of an attribute table.

of network algorithms. These animations are possible by either creating a text file of special commands which can be played back later like a movie or by accessing an available graphics library directly from the program.

These feature combine to make it easy for users to generate and study networks with specified properties, visualize relationships among networks and algorithms, and perform analysis and optimization for network-based problems. NETPAD can also be used to teach students about network properties and network algorithms and the user interface is flexible enough to allow a user to customize NETPAD for specific applications.

5. NETPAD architecture

The basic components of NETPAD are the kernel, the internal and external algorithms, the animation and graphics functions and the customization file. The kernel is the only component that we have not yet discussed in detail, and so we describe it now together with a brief look at the overall structure of NETPAD.

The NETPAD kernel unites the other components together into one coherent system by providing an interactive environment for executing internal and external algorithms, animating algorithms and editing networks. In order to provide display and interactive editing capabilities, the kernel uses the X library, X toolkit intrinsics and the Athena Widget set, and it is linked with the Xaw, Xtk and X libraries. Using X, the kernel manages the screen in order to visually represent the user's current networks and to update this representation in order to show network changes which result directly from user actions and from algorithm executions initiated by the user.

Following the design philosophy of the X Window System, the kernel uses the event model for handling user-computer interactions. (Other approaches involve the use of transition networks or context-free grammars and are inferior, see [9]). In order to manage the display, the kernel must maintain its own data structures which must agree with the network and any other information being displayed. The primary data structure type of the kernel is called grafdsp, which stands for network display. There is one grafdsp structure per network, and that structure contains pointers to

the various display objects associated with that network, including to its window and menus, as well as to the network itself and to the algorithms which are accessible via the window's menus. For example, when a node is created, the node is added to grafdsp, and when a node is moved by the user, the new position is recorded in grafdsp.

To execute an external algorithm, the kernel temporarily blocks the user's access to the network window, puts it into reverse video and forks a process for the algorithm. After the process is complete and exits, this is detected by a signal handler installed by the kernel which generates an event which contains information regarding the exit status of the algorithm and which network is associated with the algorithm. This then triggers another event to process the algorithm's results. More specifically, this last event will unblock the associated network window, put it back in normal video, display the result network in the same window, and (if appropriate) display an error message or any text or numerical information returned by the algorithm.

6. Potential uses of NETPAD

The potential uses of NETPAD fall roughly into three categories:

- research in network modeling and algorithms;
- rapid custom prototyping for specific applications;
- educational aspects of network modeling.

We have used NETPAD ourselves in a number of research areas including graph theory, combinatorial optimization and network design, and have used the animation feature for studying network algorithms. For example, we were able to use NETPAD to develop an algorithm for embedding a graph in the plane to approximately minimize the number of crossing when edges are drawn as straight lines. This is known as the rectilinear crossing number problem which is NP-hard (see [7]). For complete graphs, the exact crossing number is known for $n \leqslant 9$ (see [10]), but for $n \geqslant 10$ the classical construction of Jensen [13] produces a layout which, in several cases, is not as good as the layouts generated by NETPAD (see Table 2). We are hopeful that by observing enough instances we will be able to derive a better general construction procedure.

Table 2
Upper bounds for crossing number of complete graph on n nodes

n	Previous best	NETPAD
4	0	0
5	1	1
6	3	3
7	9	9
8	19	19
9	36	36
10	62	62
11	102	102
12	156	154
13	231	229
14	328	327
15	453	449
16	612	609
17	808	806
18	1047	1019
19	1338	1322

NETPAD provides a great deal of functionality which makes it possible to rapidly build a working prototype for specific applications. This has been done in several Bellcore projects ranging from obtaining automatic "nice" drawings of wiring diagrams, to developing tool for managing computer networks, as well as work for a network design and analysis tool for packet networks.

NETPAD is also a useful educational tool for studying network modeling algorithms and applications. For professors teaching courses on algorithms, NETPAD could be used to explain concepts which might otherwise be difficult to comprehend. It could be used as part of a lab for experimentation or in conjunction with projects for students to gain hands-on experience with algorithms. The visual and interactive nature of NETPAD might also stir some enthusiasm in students who would otherwise have little or no interest in the course. NETPAD is being used as part of the educational program at Rutgers University in conjunction with the DIMACS NSF Center for Discrete Mathematics and Theoretical Computer Science. This effort is aimed at high school level students and teachers to motivate students to pursue careers in mathematics and computer science.

7. Some existing systems

This section contains a sample of existing network modeling and analysis systems to illustrate a range of considerations that may be of interest to potential users of such systems. None of these systems contain the full range of features available in NETPAD. The systems for the Macintosh and the IBM PC have an obvious computational disadvantage imposed by the hardware, and the necessary hardware for the UNIX-based systems is more expensive and, therefore, not as readily available.

7.1. IBM PC-based programs

TRAVEL is a software package developed by Boyd, Pulleyblank and Cornuejols [2] for the traveling salesman problem. It is an interactive, menu-driven system which runs on an IBM PC. The system allows the user to choose among various heuristic algorithms and lower-bounding procedures to obtain solutions with provably near-optimal performance. Color graphics is used to display an animation of the algorithms as they are running.

INDS (Interactive Network Design System) is a software package developed by Monma and Shallcross [17] for the 2-connected network design problem, which arises in the design of survivable networks. It runs on an IBM PC, incorporates the TRAVEL user interface, and incorporates several heuristic algorithms [18].

FIBER OPTIONS is a software package developed by Cardwell et al. [3] specifically for designing survivable fiber optic networks. It uses the methods of INDS to design the network topology and uses other methods for handling aspects of the problems, like placing equipment, bundling demands and multiplexing traffic. These algorithms have been shown to produce optimal or near-optimal solutions to real-world problems. The user interface was written and designed so that it could run on several computing and graphics platforms. FIBER OPTIONS is used within Bellcore and the Bell Client Companies to plan cost-effective, survivable interoffice fiber optic communication networks. It is available to other organizations for outside licensing [1].

CARDD (Computer-Aided Representative graph Determiner and Drawer) is an expert system that constructs a graph with properties defined by the user. It was developed by T. Haynes, L. M. Lawson and M. W. Powell (personal communication) and uses a forward chaining inference algorithm; i.e., once an invariant is resolved, it is never eliminated. The properties are specified by setting values for any subset of the available set of eight invariants: number of nodes, number of edges, maximum degree, minimum degree, independence number, maximum clique size, chromatic number and domination number.

NETSOLVE is an interactive software package developed by Jarvis and Shier [12] for network manipulation and optimization. It utilizes a command language rather than a menu-driven interface and has a library of optimization algorithms. It runs on an IBM PC and does not use graphics.

7.2. UNIX-based programs

The GMP software system was developed by Esfahanian (personal communication with some of its users). It uses SUN's Sunwindows window system which is inherently less portable than systems using X Windows.

The GraphPack software system was developed by Goldberg et al. [8]. It runs under X Windows and includes a language called LiLa (which is based on the C programming language with additional primitives like sets, graphs, trees, etc.) to simplify the coding of new algorithms. It does not have a graphics interface.

The Combinatorica software system was developed by Skiena [19] and it is actually a collection of programs written in Mathematica (which must be purchased separately) and runs on a variety of UNIX-based computers. It does not have a graphics interface.

Devitt and Colbourn [6] have developed a system for investigating network reliability problems. It is an interactive, algebraic environment which provides a package of routines coded in the MAPLE language. It does not have a graphics interface.

7.3. Macintosh-based programs

Three versions of a program called CABRI are mentioned in [4], one running on a Macintosh, another on a PC-compatible, and a third version for workstations that uses the BWE window management toolset from Brown University. (Only the Macintosh version was

available to us.) It contains several network editing and analysis functions similar to NETPAD.

Groups & Graphs [14] is a program for manipulating graphs and groups. It contains various group theoretic algorithms, such as computing the automorphism group of a graph and determining whether two graphs are isomorphic. It does not have a graphics interface.

8. NETPAD availability

A version of the NETPAD software is currently being used within Bellcore. This software is a research prototype system. In addition, several documents are available which provide much more detailed information about the NETPAD system, including a User's Guide [5], a Programmer's Guide [15] and a Reference Guide [16].

NETPAD was designed to run in a workstation environment with the Unix operating system and under the X Window System. It is currently running on a SUN and DEC workstations as well as a 486-based IBM-compatible PC. To take full advantage of NETPAD, it is necessary to have adequate processing power (e.g., comparable to the machines cited above), memory (e.g., 16 MB RAM), disk space (e.g., 100 MB hard disk) and display technology (e.g., a large screen and color are useful).

The NETPAD source code and documentation are available on a royalty-free basis to universities for research, educational or academic purposes under a Software License Agreement these can be obtained by "ftp" from flash@bellcore.com in directory /usr/spool/ftp/pub/nate.

References

[1] Bellcore, FIBER OPTIONS: Software for designing survivable optical fiber networks, Bellcore, 1989.

[2] S.C. Boyd, W.R. Pulleyblank and G. Cornuejols, "TRAVEL-An interactive traveling salesman package for the IBM personal computer", *Oper. Res. Lett.* **6**, 141–143 (1987).

[3] R.H. Cardwell, C.L. Monma and T.H Wu, "Computer-aided design procedures for survivable fiber optic networks", *IEEE J. Selected Areas of Communications* **7**, 1188–1197 (1989).

[4] M. Dao, M. Habib, J.P. Richard and D. Tallot, "CABRI, an interactive system for graph manipulation", manuscript.

[5] N. Dean, C.L. Monma and M. Mevenkamp, NETPAD User's Guide, Bellcore, document, 1991.

[6] J.S. Devitt and C.J. Colbourn, "On implementing an environment for investigating network reliability", Technical report, University of Waterloo, 1991.

[7] M.R. Garey and D.S. Johnson, "Crossing number is NP-complete", *SIAM J. on Algorithms and Discrete Methods* **4**, 312–316 (1983).

[8] M. Goldberg, E. Kaltofen, S. Kim, M. Krishnamoorthy and T. Spencer, "GraphPack: A software system for computations on graphs and sets", manuscript.

[9] M. Green, "A survey of three dialog models", *ACM Trans. on Graphics* **5**, 244–275 (1986) .

[10] R.K. Guy, "Latest results on crossing numbers", *Recent Trends in Graph Theory*, Springer, New York, 1971, pp. 143–156.

[11] F. Harary and A. Hill, "On the number of crossings in a complete graph", *Proc. Edinburgh Math. Soc.* **13**, 333–338 (1962).

[12] J.P. Jarvis and D.R. Shier, "NETSOLVE: Interactive software for network optimization", *Oper. Res. Lett.* **9**, 275–282 (1990).

[13] H.F. Jensen, "An upper bound for the rectilinear crossing number of the complete graph", *J. Combin. Theory* **11**, 212–216 (1971).

[14] W. Kocay, "Groups and graphs, a Macintosh application for Graph Theory", *J. Combin. Math. Combin. Comput.* **3**, 195–206 (1988).

[15] M. Mevenkamp, NETPAD Programmer's Guide, Bellcore, document, 1991.

[16] M. Mevenkamp, NETPAD Reference Guide, Bellcore, document, 1991.

[17] C.L. Monma and D.F. Shallcross, "A PC-based interactive network design system for fiber optic communication networks", in: Sharda, Golden, Wasil, Balci and Stewart (eds.), *Impacts of Recent Computer Advances on Operations Research*, Elsevier, New York, 1989.

[18] C.L. Monma and D.F. Shallcross, "Methods for designing communications networks with certain two-connected survivability constraints", *Oper. Res.* **37**, 531–541 (1989).

[19] S.S. Skiena, *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, Reading, MA, 1990.