DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 15 1994

METANET: A System for Network Analysis

CLAUDE GOMEZ AND MAURICE GOURSAT

August 4, 1992

ABSTRACT. This paper describes METANET, a software package for graph and network analysis. It includes a CACSD system for automatic control which provides the user with a comprehensive language and interpreter for manipulating graph objects. It manages a library of FORTRAN programs encoding algorithms for solving classical graph problems and minimal cost flow network problems. A graphic editor helps easily create and modify graphs. Tools are given to randomly generate new networks. METANET is an open system where the user can add his own programs. METANET's capabilities are illustrated by an example. There the pipe network problem is solved using multigrid methods.

1. Introduction

METANET is a system for network analysis. The aim of METANET is twofold: to use various solution methods and to try new methods. It requires a UNIX system with X Window, and it is composed of two subsystems: a Computer Assisted Control System Design (CACSD system) and a graphic system. They can work on different hosts and they communicate by means of UNIX sockets.

The graphics system is written in C and C++. It allows the display and the modification of graphs, and it uses the X Window System.

The CACSD system is called Ψlab and is derived from the BASILE system. It is a MATLAB like CACSD system for automatic control developed at INRIA. It provides the user with a comprehensive language and interpreter together with the Ψlab functionalities.

METANET is an open system. You can easily add your own programs as new functions known by the interpreter, for instance for testing new methods.

1991 *Mathematics Subject Classification.* Primary 90B10, 68U05; Secondary 90C35, 68U07.
This paper is in final form and no version of it will be submitted for publication elsewhere.

After presenting the whole system in §2 (graphics and interpreter), we describe the type of graphs the system can process and the corresponding data files in §3. Then the various types of graph and network problems METANET can solve are presented in §4. The corresponding functions are listed in §5. Finally, two examples of using METANET are given in §6: the first small example shows the use of the language for a maximum flow problem and the second example tests a multigrid method for solving the pipe network problem.

2. METANET: the system

The METANET system is composed of two independent subsystems which communicate by means of UNIX sockets: `xmetanet` and Ψlab. This allows the use of the two systems on different hosts. All the computations are made in Ψlab, then you can execute it on a powerful remote computer and use a local host for graphic display. You can also use METANET across the network, using `xmetanet` as a graphic server.

When the system is activated, two windows are displayed. The first one is the graphic window with menus described in §2.1. The second one is the interpreter window described in §2.2.

2.1. Graphic window. The graphic system is written in C [8] and C++ [14]. In fact, we use the GNU compilers `gcc` [13] and `g++` [15] made by Free Software Foundation. But you can use standard C and C++ compilers. Graphics are of major interest for network problems and have been developed in METANET by using the X Window System Version 11 Release 4 [6] with the Athena Widgets library [11].

The graphics window can be used to create, modify and delete graphs. The graphics window is mouse sensitive. The mouse can be used to select an object (arc or node) by clicking once in it. Then it becomes highlighted.

Menus. There are three levels of menus. We call them the Begin menu, the Study menu and the Modify menu. They are briefly described below.

Begin Menu: This menu is displayed when METANET is activated. The items are:

- **Quit:** For quitting METANET.
- **Load Graph:** For loading an existing graph. The graph is displayed in the Study menu.
- **Load and Compute Graph:** The same as Load Graph item, but the metanet data file is recomputed from the graph data file (see §3.2).
- **New Graph:** For creating a new graph. The Modify menu is displayed.
- **Delete Graph, Copy Graph, Rename Graph:** For deleting, copying and renaming a graph.

Study Menu: This menu is the standard menu displayed when we want to use METANET to perform computations. The items are:

- **Quit:** For quitting the Study menu and entering the Begin menu.

Object Attributes: If an object (arc or node) is highlighted, its attributes are displayed in a window. Otherwise the attributes of the graph are displayed.

Find Node, Find Arc: For highlighting a node or an arc. If the node or the arc is not visible, the graph moves to display it.

Modify Graph: For entering the Modify menu.

Graphics: For defining a scale factor for displaying the graph (1.0 by default).

Modify Menu: This menu is displayed when we want to modify an existing graph or we want to create a new one

The mouse can be used to modify the graph. When you click where there is no object, a new node is created. If you click in a node and another node is highlighted, then a new arc is created from the latter node to the former node. You can create many arcs from one node to another. You can also move a node by clicking in it and moving the mouse while the button is down. The items are:

Quit: For quitting the Modify menu

Delete Object: For deleting the highlighted object (arc or node).

Number Object: For numbering the highlighted object (arc or node). You can also automatically number nodes and arcs.

Object Attributes: For modifying the attributes of the highlighted object.

Find Node, Find Arc: The same as in the Study menu

Create Source, Create Sink, Remove Source/Sink: For creating or removing sources and sinks.

Save Graph, Rename and Save Graph: For saving the graph.

2.2. Interpreter. The interpreter window provides the user with a comprehensive MATLAB like [**10**] language. In fact, we have in this window the Ψlab system derived from the BASILE system [**5**] which is a CACSD system for automatic control developed at INRIA. It is written in FORTRAN.

Ψlab manages a big library of FORTRAN programs performing numerical computations in automatic control with vectors, matrices and rational function objects

A graph is described in Ψlab by a list with the characteristics of the graph and the attributes of arcs and nodes. When a graph is loaded, its list is created. In particular, its node-arc incidence matrix is computed and put into the list in a classical sparse form (pointer, arc and node vectors). In fact, all the computations are made with this incidence matrix.

A path or a tree is a Ψlab row of arc numbers and a node set is a Ψlab row of node numbers.

The important point is that we can use these objects in numerical computations and create new functions (called macros) with loops and conditionals. So, a user can test in a very fast way his own algorithm: he encodes his algorithm in a

macro and he can execute it immediately to try it (macros are interpreted) The macro can use the functions of METANET dealing with graphs and networks An example is given in §6.1.

In the future, it will be possible to generate FORTRAN code from macros

3. Data structures and data files

3.1. Graph objects. METANET can handle two types of graphs: directed and undirected. The differences between these two types of graphs are reflected by their graphic representation and their internal representation.

Notice that an edge is an undirected link between two nodes and that an arc is a directed link from a tail node to a head node Any type of graph is internally represented by its nodes and its arcs, never by its edges This means that each edge of an undirected graph is internally represented by two arcs.

There can be more than one arc between two nodes But loops (arc from one node to itself) are not allowed.

Object attributes Each graph has a name, at least one arc and two nodes. Arcs and nodes have internal and user numbers Internal numbers are consecutive and given at the creation of arcs and nodes All the computations are made with internal numbers User numbers can be given by the user They can be any positive integer number. The graphic window only knows about user numbers

For network flow problems, a node can be distinguished as a sink or a source In the graphic window, they are drawn differently. Each node has the following attributes:

- Internal number and user number
- Demand: it can be positive or negative (supply)
- Type: plain, sink or source.
- Coordinates

Each arc has the following attributes:

- Internal number and user number.
- Physical characteristics: unitary cost, minimum capacity, maximum capacity, length, quadratic weight and quadratic origin, and weight
  The quadratic weight and the quadratic origin are respectively the values w_u and $\bar{x}_u$ in the quadratic cost $\frac{1}{2}w_u[(x_u - \bar{x}_u)^2]$ (see §4.2)
  The weight of the arc can also represent the resistance of the arc

3.2. Data files. Graphs are saved into data files The problem is that the data file must be human readable (for instance for modifying it), must be quickly loaded into the system and must contain all the data for displaying the graph into the graphic window.

The only way to reach this objective is to have two data files We call them the "graph data file" and the "metanet data file" The former is an ASCII file and contains minimal data for a graph. It can be modified by the user in a text editor It is described below. The latter is a binary file created by METANET

and contains complementary data (graphics values and redundant data for fast loading)

Structure of the graph data file. The graph data file is an ASCII file with the following structure:

```
GRAPH TYPE (0 = UNDIRECTED, 1 = DIRECTED) :
<one line with 0 or 1 according to the type of the graph>
NUMBER OF ARCS :
<one line with the number of arcs>
NUMBER OF NODES :
<one line with the number of nodes>
****************************************
DESCRIPTION OF ARCS :
ARC #, TAIL NODE #, HEAD NODE #
COST, MIN CAP, CAP, MAX CAP, LENGTH, Q WEIGHT, Q ORIGIN, WEIGHT
<a blank line>
<two lines for each arc>
****************************************
DESCRIPTION OF NODES :
NODE #, POSSIBLE TYPE (1 = SINK, 2 = SOURCE)
X, Y
DEMAND
<a blank line>
<three lines for each node>
```

For an undirected graph, ARC is replaced by EDGE.

This file describes completely the graph and is the only one needed as input in METANET. In other words, this is the file you have to generate when creating a new graph. The metanet data file is automatically created by METANET the first time a graph is loaded

You can modify an existing graph by using the Modify menu (see §2.1). But it is also possible to do it by hand by modifying the graph data file described above. Then, you have to use the "Load and Compute Graph" item of the Begin menu (see §2.1) to update the metanet data file

3.3. Generating graphs. When testing new algorithms, it is very useful to have tools to automatically generate graphs

Such tools are given in METANET. They are described below.

3.3.1. *Making a graph data file template.* This tool is used to create a graph data file template. It asks for the graph name, the number of arcs, nodes, sources and sinks. Then a graph data file is created and you can modify it by hand

3.3.2. *Netgen.* This tool is a modification of the famous netgen program for generating networks suitable for testing algorithms [**9**]

The program asks for the main characteristics of the network and then a

graph data file is created which can be the original graph or the corresponding augmented graph (only one source and one sink)

Unfortunately netgen computes no coordinate for the nodes

3.3.3. *Mesh*. This tool is a program derived from a finite element one. It asks for the main characteristics of the network and gives a triangulation of the plan From this triangulation, a network is generated with the node coordinates. The generated graph is planar.

4. Graph and network problems

We describe in this section the graph and network problems METANET can solve. The functions used to solve them are described in §5.

4.1. Graph problems. METANET can directly solve a number of graph problems. It can compute:

- the connected and strong connected components of a graph,
- shortest paths,
- a circuit in a directed graph,
- minimal weight spanning trees,
- the transitive closure of a graph,
- Eulerian paths.

4.2. Flow problems. We consider a connected directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ and $\mathcal{A}$ are respectively the sets of nodes and arcs There are n nodes and m arcs With each node i is associated a real number b_i which can be positive (supply, source node), negative (demand, sink node) or zero (pure transshipment node)

A flow on $\mathcal{G}$ is a vector x having a value on each arc and verifying the first Kirchhoff law:

$$\sum_{j:(i,j)\in\mathcal{A}} x_{i,j} - \sum_{j:(j,i)\in\mathcal{A}} x_{j,i} = b_i \qquad \forall i \in \mathcal{N}$$

With each arc (i, j) (from node i to node j) are associated two numbers $l_{i,j}$ and $u_{i,j}$ which are respectively the lower and upper bounds for the flow Moreover a cost $c_{i,j}$ is associated with each arc.

METANET can compute the maximum capacity path from node i to node j It is a path for which the minimum capacity of all the arcs is maximum

The general minimum linear cost network flow problem consists in finding a flow which minimizes the linear cost on the arcs and which verifies the capacity constraints. We call it problem $(\mathcal{P})$:

$$(\mathcal{P}) \quad \begin{cases} \min \displaystyle\sum_{(i,j)\in\mathcal{A}} c_{i,j}x_{i,j} \\ \displaystyle\sum_{j:(i,j)\in\mathcal{A}} x_{i,j} - \sum_{j:(j,i)\in\mathcal{A}} x_{j,i} = b_i \qquad \forall i \in \mathcal{N} \\ l_{i,j} \le x_{i\,j} \le u_{i,j} \qquad \forall (i,j) \in \mathcal{A} \end{cases}$$

METANET can solve problem $(\mathcal{P})$ and various problems related to problem $(\mathcal{P})$ as we describe below.

If we suppose that vector b is equal to 0, i.e. there are only pure transshipment nodes, we obtain problem $(\mathcal{P}_1)$.

Moreover, we can assume that the graph has only one source and one sink. If the graph has more than one source and one sink, it is possible to link all the sources to a single super source and to link all the sinks to a single super sink (this is the augmented graph).

With these assumptions we can compute the maximum flow from the source to the sink according to the flow bounds on the arcs. This is problem $(\mathcal{P}_m)$. The value of this flow is called the value of the flow on the return arc (virtual arc from sink to source).

With theses assumptions, we can also impose the value of the flow on the return arc and solve problem $(\mathcal{P}_1)$ for this flow value. This is problem $(\mathcal{P}_2)$.

All these flow problems are linear. We can consider a minimum quadratic cost network flow problem. We call it problem $(\mathcal{Q})$:

$$(\mathcal{Q}) \quad \begin{cases} \min \displaystyle\sum_{(i,j)\in A} \Gamma_{i,j}(x_{i,j}) \qquad \text{with} \quad \Gamma_{i,j}(x_{i,j}) = \frac{1}{2} w_{i,j}[(x_{i,j} - \bar{x}_{i,j})^2] \\ \displaystyle\sum_{j:(i,j)\in A} x_{i,j} - \sum_{j:(j,i)\in A} x_{j,i} = b_i \qquad \forall i \in \mathcal{N} \\ l_{i,j} \le x_{i,j} \le u_{i,j} \qquad \forall (i\ j) \in A \end{cases}$$

where $w_{i,j}$ and $\bar{x}_{i,j}$ are given.

If we suppose that the vector b is equal to 0, i.e. there are only pure transshipment nodes, we obtain problem $(\mathcal{Q}_1)$.

4.3. Pipe network problem. The pipe network problem consists of finding the flow x on the arcs and the potential π at the nodes verifying the matrix equations:

$$(\mathcal{PN}) \quad Ax = b \qquad A^T\pi = -Rx \qquad \sum_{i=1}^{n} b_i = 0$$

where A is the classical $n \times m$ node-arc incidence matrix, b is the supply at the nodes and R is the $m \times m$ diagonal matrix of the resistances of the arcs. The first two equations are the Kirchhoff laws and the last one is necessary for the system to be feasible (supply equals demand).

We show how METANET can be used to solve this problem in §6.2.

5. METANET functions

METANET has about fifty functions for graph and network problems. See §4 for the description of the various network problems.

Most of these functions have the list describing the graph as an optional argument. If this argument is omitted, the current graph is taken as the value of the Ψlab variable `the_g`.

Recall that a path or a node set is represented in Ψlab by a row vector of integers.

5.1. Graphic functions. There are two main functions dealing with graphic display:

`showp(p)` highlights the path `p` in the graphics window.

`showns(ns)` highlights the node set `ns` in the graphics window.

They allow the visualization of results of computation and even the display of intermediate graphical results during a computation

5.2. Graph functions. The function `loadg(name)` loads the graph called `name` and returns the list describing it.

When performing computations in METANET, we can create or obtain new graphs described by lists They are internal to the interpreter A function saves such a new graph as graph data file; that is, `createg(g)` where `g` is the list describing the new graph Then, we can load and display it into the graphics window.

The functions which solve the graph problems described in §4 1 are resident in METANET. For instance `sconnex()` returns the number of strong connected components, `sconcomp(i)` returns the node set of the component number `i`, `ansp(i,j)` returns the arc number shortest path from node `i` to node `j` and so on.

All these functions come from FORTRAN subroutines described in [1]

There are also functions for manipulating paths and node sets, for instance for obtaining the node set corresponding to a path.

5.3. Network functions. These are the functions which solve the network problems described in §4 2

For instance, we have:

- `maxcpp(i,j)` returns the maximum capacity path from node `i` to node `j` and the the capacity value.
- `maxf()` returns the maximal flow array and the value of the flow, solving problem $(\mathcal{P}_m)$.

For minimum linear cost network flow problem, we have:

- `mincfr()` returns the array of flows for minimum cost network flow problem $(\mathcal{P})$ by using a relaxation method and the value of the cost. This method is due to Bertsekas [**3, 4**].
- `mincf0()` returns the array of flows for minimum cost network flow problem $(\mathcal{P}_1)$ and the value of the cost
- `minicf(v)` returns the array of flows for minimum imposed cost network flow problem $(\mathcal{P}_2)$ and the value of the cost `v` is the given value of the imposed flow

For minimum quadratic cost network flow problem, we have:

- `minqflow(eps)` returns the array of flows for quadratic minimum cost

network flow problem ($\mathcal{Q}_1$). eps is the precision of the iterative algorithm.

`mincfr` function comes from FORTRAN subroutine described in [2] and all of the others come from FORTRAN subroutines described in [1]

5.4. Adding new functions. A main feature of METANET if that a user can add his own programs as new functions in the interpreter. This is a very important characteristic. For that, there are two solutions.

First, he can make macros which are complete programs in the Ψlab language. These macros are then loaded into METANET and readily executed without being compiled (they are interpreted). This is a very fast way to test algorithms. Indeed, Ψlab has a very simple syntax for performing matrix computations and has the structures of a complete language: loops, conditionals. Moreover METANET functions can be used and combined into these macros. A small example of a macro is shown in §6.1. The only drawback is that the running time of a macro is not as fast as compiled C or FORTRAN programs.

Second, the user can even include his own C or FORTRAN programs into METANET. There is an interface called InterΨ which does the job. You have to describe the FORTRAN or C program and the corresponding Ψlab function in an ASCII file. Then executing InterΨ creates the needed software to be included in METANET. And now, you have to recompile METANET, but the process is easy to do.

6. Sample sessions of METANET

6.1. Maximal flow path. We first create a new graph by using the program `mesh` described in §3.3. We call it "mesh100". It has 100 nodes and 284 arcs, one source and one sink and the arcs are capacitated.

We want to compute the maximal flow array from the source to the sink and get the path where this flow is non zero.

The `maxf` function returns the flow array and we create a new macro for computing the desired path. In Ψlab, a // introduces a comment. The macro is:

```
// macro maxpath which returns path p
function p=maxpath()
  // [v,f] = sequence returned by MEIANEI function maxf
  //   v = value of the flow, f = array of maximal flow
  [v,f]=maxf()
  j=0
  // loop on f to get the non zero values of f and put them
  //   in row vector p
  for i=1:arcnum(),
    if f(i) <> 0 then j=j+1; p(1,j)=i; end
  end
```

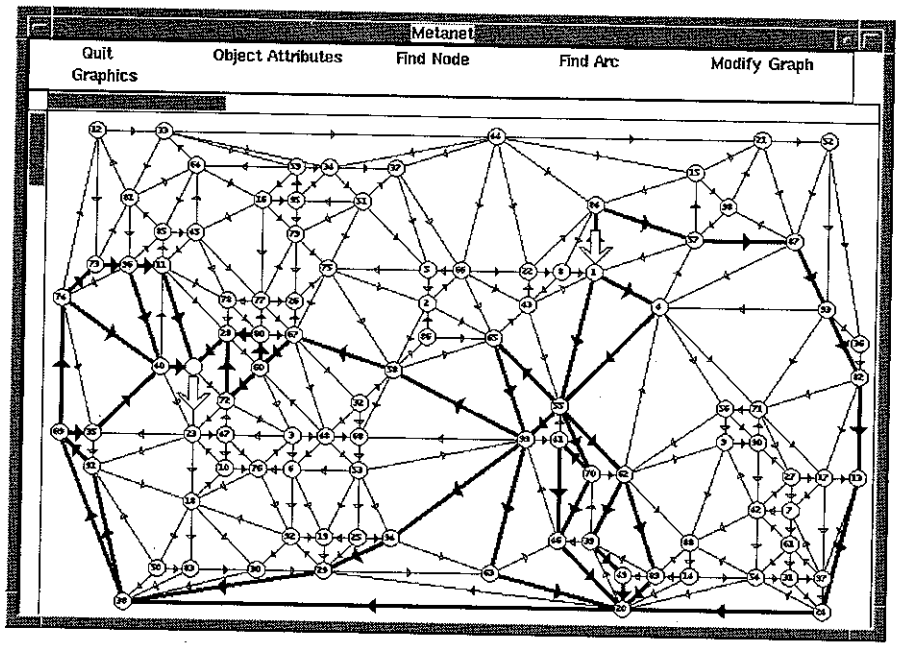

FIGURE 1. Path for maximal flow in network "mesh100"

We save this macro in the file `maxpath.sci` and the session in the interpreter window is:

```
-->the_g=loadg('mesh100'); // load graph mesh100, put it in the_g
-->getf('maxpath.sci') // load file maxpath.sci
-->p=maxpath(); // compute path p
-->showp(p) // display path p
```

Figure 1 shows METANET graphic window with highlighted displayed path

6.2. Pipe network problem. Solving the pipe network problem consists in solving problem $(\mathcal{PN})$.

We can transform the equations in the following way. The rank of A is $n-1$. If we call A' the A matrix with its last line deleted, it has full rank. We call b' et π' the corresponding $n-1$ vectors. The potentials are defined up to an additive constant and we can choose $\pi_n = 0$. Then we can eliminate the flows from system $(\mathcal{PN})$ and we have the following matrix equation for the potentials:

$$(A' R^{-1} A'^{T})\,\pi' = -b'$$

The matrix of this system is sparse, symmetric and non singular.

So, we can use algebraic multigrid methods [12] to solve this system. These methods are very efficient for such parse systems. These are iterative methods. They solve the (sparse) linear system $Mu = b$ by means of multigrid cycling

process (coarse grids) i.e. they solve a sequence of smaller systems of equations:

$$M^m u^m = b^m \quad m = 1, 2, \ldots, q \quad \text{with} \quad n = n_1 > n_2 > \quad > n_q$$

To compute the sub-matrices (coarser grids), the process is purely algebraic i.e. the algorithm uses only the information explicitly contained in matrix M by deleting rows and columns with same index. For the network problem, deleting row i and column i of the matrix is equivalent to delete node i. To each sub-matrix corresponds a subnetwork and we can use METANET to visualize the various subnetworks corresponding to the various grids used by the algorithm.

So, we can see how the algorithm works and try to make geometric remarks to be re-used for other algorithms.

It is very easy to code such a program in METANET. First, we add the FORTRAN multigrid subroutine as a new function and then we write a macro which uses it and returns a list of the subnetworks. And we only have to use the function `createg` (see §5.2) to get them into METANET.

We have applied this new macro to network "mesh100" already used and shown in Figure 1. The resistance of the arc is proportional to its length.

The successive subnetworks are shown in Figure 2, Figure 3, Figure 4 and Figure 5.

For instance, from these subnetworks, we can see that nodes 33 and 47 seem very important. In fact they correspond to two groups of nodes, (33, 41, 55) for node 33 and (10, 23, 47, 72) for node 47, and many arcs are linked to these groups.

7. Conclusion

We have tried to show that METANET can be a useful tool not only for analyzing networks by using the resident functions and graphics, but also for easily creating new functions and testing new algorithms.

The system is still under development, and much work must be done:

- To improve graphics, for instance to have various drawings of nodes or to allow the user to draw himself his nodes
- The user must be able to easily add new attributes for arcs and nodes.
- To add new functions: for graph theory, for non linear cost problems and many other flow network problems.
- To generate FORTRAN code from Ψlab macros

References

1. G. Bartnik and M. Minoux, *Graphes algorithmes logiciels*, Dunod, 1986.
2. D. P. Bertsekas and P. Tseng, *RELAX: A Computer Code for Minimum Cost Network Flow Problems*, Annals of Operations Research, vol. 13, 1988, pp. 127–190.
3. ———, *Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems*, Operations Research, vol. 26, no. 1, January-February 1988.
4. D. P. Bertsekas and J. N. Tsitsiklis, *Parallel and Distributed Computation*, Prentice Hall, 1989.

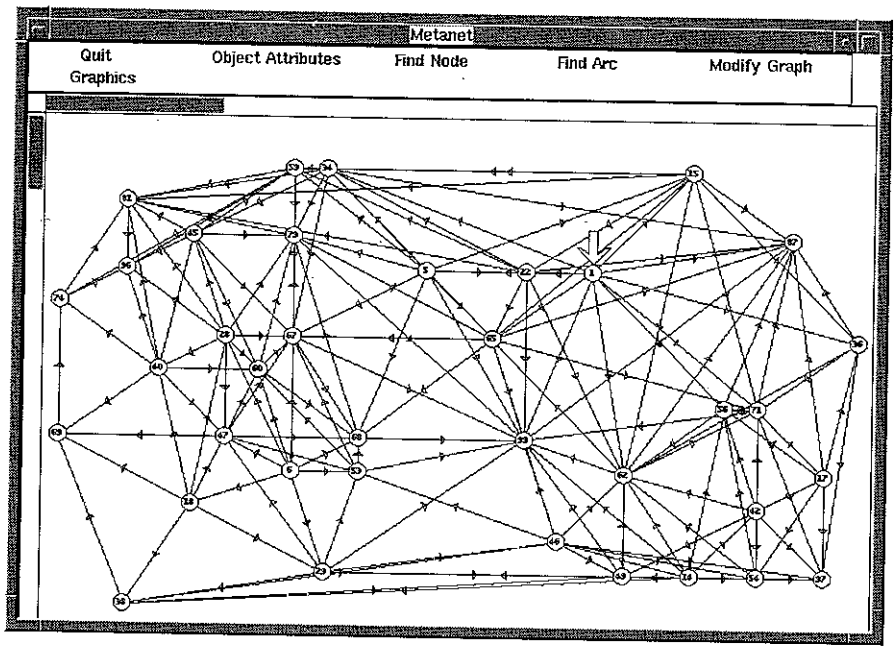

FIGURE 2. Grid 2 for network "mesh100"

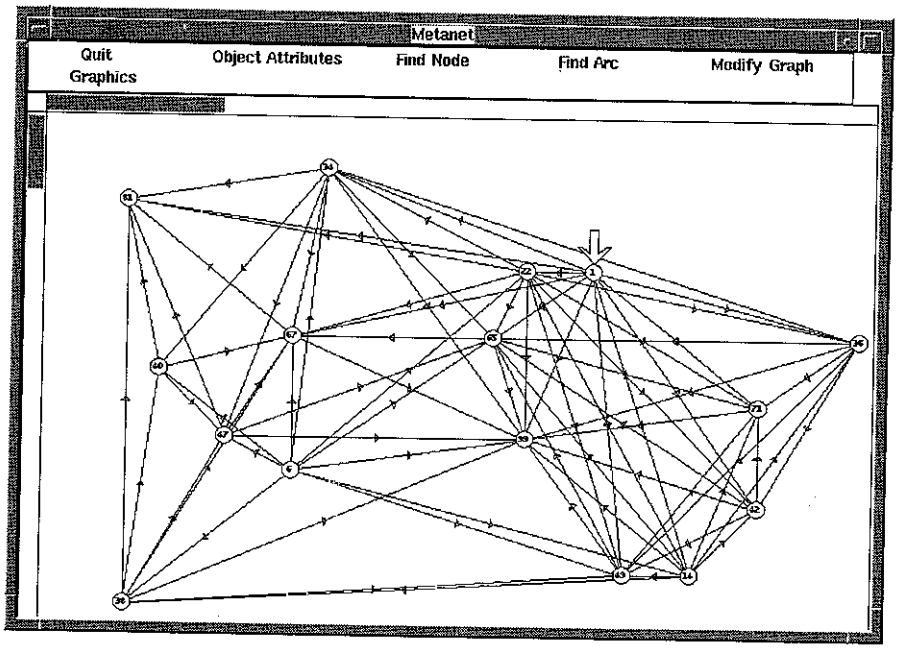

FIGURE 3 Grid 3 for network "mesh100"

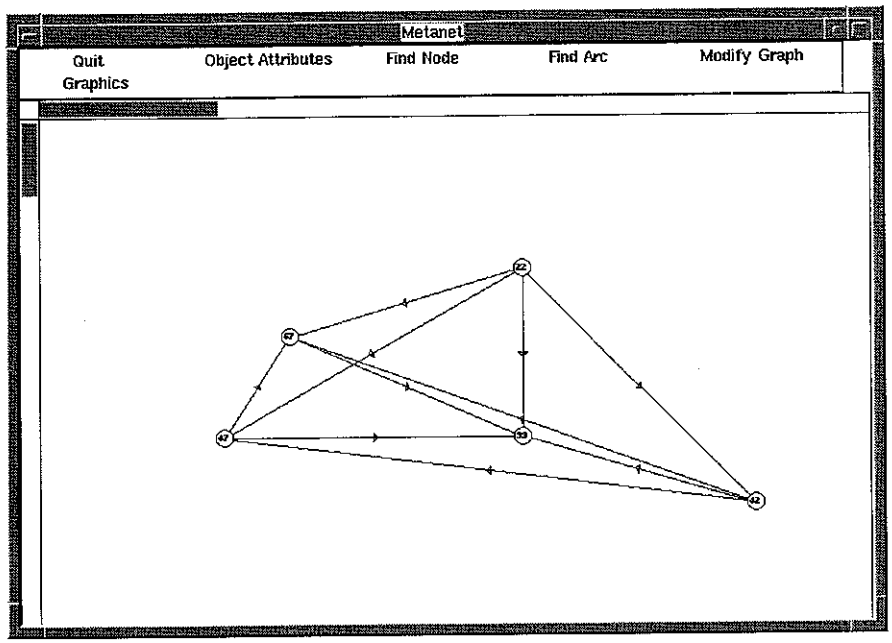

FIGURE 4 Grid 4 for network "mesh100"

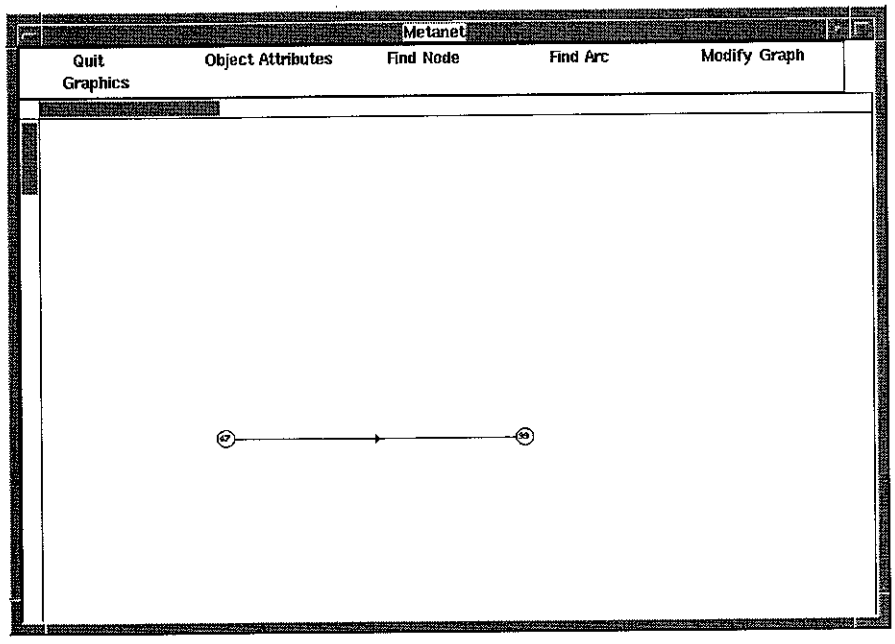

FIGURE 5 Grid 5 for network "mesh100"

5. F. Delebecque, C. Klimann and S. Steer, *BASILE, guide de l'utilisateur*, INRIA, 1989.
6. J. Gettys, R. W. Scheifler and R. Newman, *X Window System, Xlib - C Language X Interface, X Version 11 Release 4*, MIT, 1989
7. M. Gondran and M. Minoux, *Graphes et algorithmes*, Eyrolles, 1985
8. B. W. Kernighan and D. M. Ritchie, *The C Programming Language*, Prentice Hall, 1978
9. D. Klingman, A. Napier and J. Stutz, *NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation and Minimum Cost Flow Network Problems*, Management Science, vol. 20, no. 5, January 1974.
10. C. Moler, *MATLAB User's Guide*, Technical report CS81-1, Department of Computer Science, University of New Mexico, 1982
11. C. D. Peterson, *X Window System, Athena Widget Set - C Language Interface, X Version 11 Release 4*, MIT, 1989.
12. J. W. Ruge and K. Stüben, *Algebraic Multigrid*, in Multigrid Methods edited by S. F. McCormick, Frontiers in Applied Mathematics, SIAM, 1987.
13. R. M. Stallman, *Using and Porting GNU CC*, September 1989.
14. B. Stroustrup, *The C++ Programming Language*, Addison-Wesley, 1986
15. M. D. Tiemann, *User's Guide to GNU C++*, August 1989.

INRIA, Domaine de Voluceau, BP 105, 78153 LE CHESNAY CEDEX, FRANCE
E-mail address: Claude.Gomez@inria.fr

INRIA, Domaine de Voluceau, BP 105, 78153 LE CHESNAY CEDEX, FRANCE