Interactive Network Edit and Display Program for the Apple II Personal Computer

ROBERT B. DIAL

An interactive network edit and display program (NED), a key module in a set of programs for training transportation planners, is described. This interactive graphics program for the construction and display of highway network models acts like an electronic blackboard on which links can be drawn and their attribute values written. The NED program is written for the Apple II personal computer in UCSD Pascal, and its hardware requirements are minimal: a television set, the language card, and at least one 5.25-in. disk drive. In spite of the limited speed and capacity of the Apple, the capabilities and productivity of NED rival those of similar programs on much larger computers. The major shortcoming of the program is that it can process only small networks of the scale used for training or sketch planning.

The interactive network edit and display (NED) program is intended to be a key module of an integrated set of programs for use in training transportation planners. A user-friendly program for the construction, modification, and display of highway networks, NED is written for the Apple II microcomputer in UCSD Pascal. Its hardware requirements are minimal: namely, an Apple II outfitted with the language card, one 5.25-in. disk drive, and a home television receiver. In spite of the limited capacity and speed of the Apple II Pascal system, the functional capabilities and productivity of NED rival those of comparable programs written for much larger computers. The major shortcoming of the program is that it can process only small networks of the scale used for training or sketch planning.

PROGRAM CAPABILITIES

NED can build a new highway network from scratch or modify an existing network. It permits the user to draw a network on the screen, display the attribute values posted on their respective links, and delete links or nodes. A windowing feature enables the user to deal only with subnetworks at a scale that aids editing and clarifies the display. At the end of a session the network can be saved in a file named by the user for input into other programs or for reuse by NED.

Conceptually, NED acts like an electronic blackboard on which the user sketches a network by drawing its links and writing on them the value of their attributes, such as speed and capacity. Users find it natural to deal with a network presented in NED's two-dimensional graphic display.

PROGRAM FEATURES

NED is easy to use. An enormous amount of network coding can be done in a short time. This efficiency is due mostly to the graphic display and the ability to interactively modify it along with the network it represents. All user communication with NED is accomplished by means of an interactive dialogue in which NED issues prompts and lists alternative replies. The user types in a service request as a command responding to NED's prompt. All prompting is self-explanatory, and all legitimate responses are listed whenever screen space permits.

One feature that makes NED easy to use is that all of its commands require only a single keystroke. This is a feature of the UCSD Pascal command language that is valued by all its users. Only in entering file names or numbers does a user have to depress more than one key. Therefore, in the discussion that follows, a command should be understood to mean a one-character code typed in without a carriage return. For example, "type the Cancel command" means strike the C key. In addition, the phrase "link attribute name" should be interpreted to mean a single character. For example, the link attribute name for speed is the letter S.

The NED program is consistent both visually and semantically. Prompts always appear at the bottom of the screen. The menu of acceptable commands always appears vertically on the right of the screen. The network always fills the remaining space. Visual continuity is good. NED nearly always plays back user input and writes its response where the user's eye is currently fixed. The same command key always means the same thing in the same context or an analogous thing in a different context. For example, the Quit command always transfers the user to the top command level, and the same set of commands is used for left, right, up, and down movements whether they be for moving a window, a node or link, or a node or link locator.

The user need not fret about making a mistake in using NED. The Apple II costs nothing to use and is nearly indestructible. Furthermore, NED is self-protective: it always carefully edits input, only accepting that within the legitimate domain of values. Whenever the user does something unexpected, NED invariably responds with an audible beep, refusing the input and stubbornly reissuing its prompt. It is difficult to make NED accidentally crash, and the networks it outputs contain no data errors.

PROGRAM WEAKNESSES

In spite of its capabilities, the NED program does have some serious shortcomings, due primarily to limited main storage. The major shortcoming of the program is that it can process only tiny, rigidly defined networks. The maximum number of nodes (or zones) is 99, and each node can have a maximum of 4 nodes connected to it. Thus, the maximum number of links is 396. Furthermore, there are only 7 link attributes, their names are hard-wired, and their values can only be postive integers within fixed ranges.

There are three reasons for these constraints:

1. NED was built for tutorial purposes. It could be argued that in the classroom a general data base facility is excessive and a network of 99 nodes is large enough to teach the fundamentals of highway network analyses.
2. I considered it better to sacrifice network size for well-engineered (i.e., reliable and maintainable) code, extensive functional power, and good human factors.
3. Larger personal microcomputers are already appearing on the market (e.g., the IBM personal computer for which the NED code will be drop-in-compatible). These newer machines have storage capacity that would permit NED to work with networks of more than 1,000 nodes. Thus, thanks to UCSD Pascal and the wonders that are being done with semiconductors,

NED can be made a more powerful program without resorting to bit squeezing.

Another obvious shortcoming of NED is the absence of color in the graphics. Again the hardware is the culprit. In the initial stages of program development, color was used. But maintenance of a color display required a little more memory, and NED ultimately did not have it to give. More important, with the Apple computer, resolution of the screen becomes critically degraded when color is used. In black and white, the screen has a grid of 280 pixels in the x-direction and 192 in the y-direction. Although this is crude enough, in color the x-direction is effectively reduced to 140 pixels. Besides fatally compromising digitizing accuracy, this coarse grid causes links that are not horizontal or vertical to be displayed as colorful thunderbolts.

The coarseness of the Apple screen also precludes displaying many links at one time. Numbers are conventionally drawn in a 7- by 8-pixel matrix. This means that a digit requires a good deal of screen area and links have to be quite far apart to display posted values of attributes so that they do not overwrite one another. This means that only one attribute can be posted at a time and often that the window size must be reduced more than would otherwise be desirable.

FUNCTIONS

The specific functions of NED fall into eight categories:

1. Starting the session,
2. Windowing,
3. Posting link attribute values,
4. Setting default link attribute values,
5. Changing link attribute information,
6. Delineating network topology,
7. Help messages, and
8. Quitting the session.

With the exception of starting the session, each of these categories corresponds precisely to one top-level command, given by its capitalized letter. In addition, because each constitutes a separate overlay segment, there is a slight delay when it is invoked while its code is loaded into memory. Except for starting and quitting, all functions can be invoked at any time and as often as desired. The functions present in each overlay attempt to minimize the number of times the user must leave it.

Starting the Session

Starting up NED means loading the program and then responding to prompts that ask whether a new network is being created or an existing one is being modified. In the latter case, the user provides the file name of the existing network. If a file name is given that NED cannot match up with a file presently on line, it will restate the prompt, giving the user endless opportunities to get it right.

Windowing

The windowing routine is usually the first command the user issues in a session with the NED program. The windowing routine locates the portion of the network or geography on which the user wants to work. The windowing facility is needed because the limited size of the display screen requires that all but the smallest networks be dealt with in visual subsections. Windowing permits the user to select and scale subsections and also to restrict link group updates to appropriate subsets of links.

The windowing routine is entered by typing the Window command (the letter W). When it is invoked the first time, the windowing routine clears the screen and draws the entire network, surrounded by a square window frame. With this frame the user can specify the (square) subsection of the network he wants displayed. This is done by resizing and repositioning the floating square frame superimposed on an unchanging display of the network. When the user quits the windowing routine, the subarea currently within the frame is blown up to fill the entire display. That is, the network is redrawn so that the frame is coincident with the periphery of the screen. Thereafter, until the user rewindows, any editing or posting can reference and affect only the blown-up subsection of the network.

Posting Link Attribute Values

Posting link attribute values means simply printing the value of one of the link attributes at the center of the displayed link. Each link has a value for seven link attributes: road type, lanes, speed, capacity, distance, time, and volume. This routine allows the user to post any of the seven link attributes on all of the displayed links with a single keystroke. Only values that satisfy a certain simple constraint can be posted. In the posting routine, the user can also simultaneously change the posted attribute value on all links by using a conditional algebraic (Algol-like) expression. For example, if the user wished to increase the number of lanes (L) on all the links currently appearing on the screen whose volume (V) to capacity (K) ratio is greater than 90 percent, he would type the following conditional algebraic expression: L = IF 100 V/K > 90 THEN L+1 ELSE L.

Setting Default Link Attribute Values

NED maintains and uses four sets of default values (CDV) for each of the seven link attributes. They are used for link group updating and for assigning attribute values to links that are newly created in the topology editor. These values can be changed at any time by using the default value editor. The user simply types the link attribute name followed by its new CDV.

Changing Link Attribute Information

Any of the seven attribute values of any of the displayed links can be changed in the link information editor. The link information editor operates in two modes: one mode lets the user scramble quickly around the network, changing any link's value of the one currently posted attribute, and the other mode displays and edits the values of all of the attributes on any preselected a-b node pair for both directions. In either mode the current default value can be given a link simply by typing D. The reason for the two modes is that it is impractical to display all seven of the link attributes on all of the displayed links simultaneously.

Delineating Network Topology

In the NED program, network topology simply means how the nodes are located and interconnected with links. The topology editor allows the user to draw new nodes anywhere on the screen while hooking them together with links. The user can also move or delete existing links and nodes. All is done graph-

ically with a small set of streamlined commands that permit the user to lay out or modify the network quickly.

The topology editor does not deal with link attribute values. In the topology editor all newly created links are automatically considered two-way and are assigned current default values in both directions. This makes it possible to do the bulk of the coding quickly. The user then goes back to the link information editor to fine tune the attribute values of the new links or set links to a one-way direction.

Help Messages

Messages explaining the commands of each of the preceding categories are obtained by using the Help command. When the Help command is input, an explanation of all the top-level commands appears. Then, by giving any of the functional commands (e.g., Window), the user receives a message that explains the command. (Because all help messages reside in a separate overlay segment, any specific help message takes a couple of seconds longer to get than it would if it could fit into the same overlay segment as the function it explains.)

Quitting the Session

At the end of a session, the current state of the network can be saved either in the same file as the original input or in a new file named by the user. Prompts for determining where to write the network file are self-explanatory. The user may not leave a session without saving the network.

FUTURE ENHANCEMENTS

NED is but one, albeit essential, program in a planned multiprogram suite of transportation software for the Apple II computer. Among the modules that directly use or provide NED data and are currently being coded or tested are the following:

1. A network data input routine that uses the graphics tablet,
2. A user-optimal equilibrium traffic assignment model, and
3. A gravity-type trip distribution model.

The following modules are now in the planning stage:

1. A geographic data base management system,
2. A dual independent map encoding (DIME) file editing and geocoding suite,
3. An interactive graphics transit network coding module,
4. A transit patronage estimation model,
5. A transit system cost model, and
6. A land use forecasting model.

All code will be written in UCSD Pascal. This will ensure a code that is intelligible, structured, and maintainable and also reasonably portable. Because more and more machines, including some minicomputers, have a UCSD Pascal compiler, it will be possible one day to run NED and its companion software on a wide variety of computers.

ACKNOWLEDGMENT

I would like to pay tribute to the Apple II computer for its high reliability, low cost, and the fine Pascal development system of K. Bowles. Of course, I am really acknowledging the brilliance of the people who are producing semiconductors and the systems that use them. I would also like to thank three friends for their assistance: R. Fisher, for providing valuable feedback in the early functional design stage of the NED program; G. Paules, for shouldering extra work to free me to work on NED; and J.D. Foley, for his fine course in interactive computer graphics.

I conceived, designed, programmed, coded, debugged, and documented the NED program, working only with the Apple II. The software, this paper, and all their bugs are solely my responsibility.

Publication of this paper sponsored by Committee on Computer Graphics and Interactive Computing.

Notice: The Transportation Research Board does not endorse products or manufacturers. Trade and manufacturers' names appear in this paper because they are considered essential to its object.