DESIGN AND IMPLEMENTATION OF AN INTERACTIVE ROUTE EDITOR

Guy Lapalme and Michel Cormier
Département d'informatique et de rech. opér. Centre de recherche sur les transports
Université de Montréal, C.P. 6128, Succ. "A", Montréal, Québec, Canada, H3C 3J7

Abstract

We present an interactive color graphic system designed for pickup and delivery route modification. We identify three fundamental types of commands and show that they correspond to those of a typical text editor.

Résumé

Nous présentons un système graphique avec un écran couleur pour effectuer la modification de parcours de véhicules de transport. Nous identifions trois types fondamentaux de commandes et montrons qu'elles correspondent avec celles d'un éditeur de texte habituel.

INTRODUCTION

Operations Research techniques now provide us with many algorithms for real life routing problems [1,2,3]. Unfortunately, being often heuristic in nature, their results must be evaluated by hand (by eye would be a more appropriate term). This evaluation must take into account not only the usual constraints (capacity, length, time duration, etc) but also other unwritten ones (overall shape of the route, going back on one self, certain types of vehicle might not be able to go over certain roads, etc). This process usually prompts some more modifications who then in turn must be evaluated.

The best way to appraise such routes is their drawing on a road map; usually many of them are on a single territory and we are interested in having a more global improvement than one route at a time; to differentiate them, we give a distinct color to each route. Hand drawing those results is very slow and tedious, so the generated routes are accepted as they are or worse they are rejected as a whole. So what we need is an automatic draughting tool who can display both the road network and the routes in color. This implies a high resolution so that the drawing appears as it would on a real map. For that purpose, we can think of two kinds of hardware: color plotters and color graphic screens. The first tool gives a very good resolution and beautiful drawings but does not allow easy interactions to achieve the fast modifications that we plan to implement. This paper describes a tool to answer that need: a route editor which displays the routes and permits easy and fast changes in them.

We will show that this system has essentially the same structure as any text editor found in computer systems today [4]. As those editors have been in use for many years now,we can expect that they are now on solid grounds and that their main commands correspond to the needs of a user changing a text already in machine-readable form. So if our commands parallel those of to the editor, we are confident that our foundamental needs will be fulfilled.

2. Structure of a text editor

We first give a brief overview of the fundamental structure of a typical interactive text editor. Its commands can be classified in three major categories:

- file reading and writing commands to choose the file to be modified and to keep it for future use
- modification commands to change the contents of the file: we can insert new lines, delete lines, move lines, change only a few characters in a line. We need also a way to indicate which lines are affected by the changes
- display commands to list the contents of lines before or after the modifications

3. Structure of the route editor

We think that a route editor should accomplish the same modification tasks on a file describing stops in routes a text editor does on a text file. If we consider the following analogies: a route corresponds to a line of text, a stop in a route corresponds to a character in a line. So we can create new routes and delete old ones; we want also to remove, add or reorder stops in a route. The main difference will be in the display commands where on one side, a text editor works on a linear basis (characters in a line) but on the other side, the stops in a route are given by coordinates in a 2-D space; this implies a more elaborate set of display commands not found in a typical text editor.

The overall scenario of an editing session is as follows: we first choose the routes to worked on; they are displayed over the road network, they are modified on line and are then written back to the file storage system. Table 1 lists the main commands.

EDITOR COMMANDS

Modification	
add	route or stop
delete	route or stop
move	stop(s) to a route
reorder	stops within a route
Display	
visibility	on/off of a route
colorswap	between two routes
zoom	in/out of a region
Others	
option settings	
undo	
help	

Table 1

4. Display commands

Those commands are very straightforward in text editors but are less so in our context because of the many variations that can occur: for example, we can display the whole network (see figure 1 were 1000 nodes and ... arcs are displayed) or just enough to display the chosen routes, the stops can be linked either with the shortest path on the network or as the crow flies (see figure 1 where stops are indicated by dots over the network). In back and white they are difficult to distinguish from the base network, but that only contributes to make our point that color graphic are essentials in these kinds of application.

Once the routes are displayed, we may need to clear up the situation for a better evaluation: for example, by changing colors of the routes, by removing a few ones from the display or even zooming in on a particular area of interest (figure 2 was obtained by zooming on a region of figure 1). Once we have a satisfactory display, we can start the modifications themselves.

The implementation of these commands is quite straightforward once we have the basic informations about the network: for each arc representing a portion of a street between two corners, we keep the node numbers of its ends and its length; for each node representing the street intersections we have their x-y coordinates and pointers to the arcs coming in and out of this node; the routes are lists of stops each comprising a node number and other problem specific informations: stop number, number of persons to pick up, time of pickup, etc.

We see that we have all the information needed for drawing the network lines to be drawn (possibly clipped) between coordinates. The routes also follow the same pattern except when the exact path between each stop has to be drawn. Memory limitations do not allow keeping constantly in memory these paths which can involve hundreds of arcs, a typical route being 30km long. So before drawing that path, we first compute it using a classical Dijkstra shortest-path algorithm stopping as soon as the destination has been permanently labelled and keeping in each node the number of the preceding node in the path. We then follow the pointers to draw the whole path. This whole procedure is usually quite fast (1 to 10 seconds) between each stop and gives the user a very good feeling about the shape of the route; this is much better than the "crow fly" approach and so it is worth it.

Changing the colors of the routes or making them visible or invisible is done almost instantly because it only involves changing a few bytes in the color look-up table.

5. Modification commands

We have three levels of modification:

- route level: we can create or delete new routes
- stop level: we can add, remove stops from a route; reordering of stops within a route has proven itself very useful in our context. It is also possible to move stops from one route to another. When the modifications are made the result is immediately reflected on the screen.
- within a stop: a number of attributes characterize a stop; for example, the quantity of goods to pick up of deliver, the time window for servicing, the names of the people to pickup, etc. Updating that information does not involve 2-D spatial information and can be done with the usual text editing functions. But care must be taken to update the global attributes of the route accordingly: for example, total quantity, total time length, total number of people, etc.

It is very important that the modifications be done almost instantly, but some of them may require a certain amount of time and memory, for example to find the shortest path between each stop in a route. To be effective, we must have a very fast computer and a high transmission rate. We have chosen to implement this system on a dedicated work station composed of a high resolution bitmap color screen with a MC68000 based microcomputer running an UNIX like operating system. This is surely a very expensive piece of equipment but we must realize that the operations we want to optimize are very costly ones where a small part of the expected savings can pay off the investment.

The implementation of the commands involves processing the lists of stops: removing stops from one list and moving them to another, reordering of lists, deleting items from lists, etc. When lists are modified, the old version of the route is redrawn using the background color and the new modified route is redisplayed.

6. Interaction aspects

The fact that we have a dedicated computer is very important to achieve a constant interaction rate. The user is not slowed down by the works of others like in a time-sharing mode. So simple commands are done almost immediately and more complicated ones take a longer time; the user expects this and can appreciate that it is his command which ties up the system.

We have also implemented a very simple interface consisting of menus and cursor positioning with a joystick, the keyboard is almost never needed. This allows direct manipulation of the objects (in our case the stops and the routes) which is recognized as one of the easiest way to interact with a system [5].

Another important aspect of the system is the possibility of undoing commands. If we realize that we made an error in specifying the nodes involved in a command or that a command did not give the expected improvements then, by touching one key, we can go back to the situation before that command. This feature is very important because in this way, one can freely experiment with the system without having to find the inverse of an erroneous command. In our system, we can currently go back over the last five commands; before that, the modifications are committed, but there is always the possibility of not saving those modifications in the case of an earlier error. The implementation of that feature is quite intricate because we have to execute the inverse of each command; in the case of reordering of stops, this inverse is almost impossible to find so the original order is kept in heap memory in case of backing up.

As it is very difficult to give a real feeling of such an interactive system by writing, we have produced an eight minute Super-8 film which shows how modifications can be made in a real context.

This system is closely related to the works of Cullen [6], Fisher [7] and Babin [8] but these systems are not as much oriented towards the interactive route editing approach as ours. Cullen and Fisher use color graphic microcomputers mainly for the display, the main computations being made on a remote mainframe linked by a telephone line. Babin features a transit planning system running on a stand alone powerful microcomputer. It aims more at display-

ing analytical results in form of tables and graphs than at showing routes over a network; this system also includes an interactive network editing module but not a route editing one.

7. Conclusion

Our experience with this tool has proven that it is very user-friendly, easy to use and very cost effective. It has allowed users to evaluate and modify more than 150 routes in two days. We have focused here on the modification aspects of the editor but it can also be used to create new routes from scratch. We are currently involved in other works to improve the editor by integrating the notion of time and other basic algorithms for example: clustering, traveling salesman, insertion, etc..

Acknowlegments

We would like to thank Serge Lafrance, Claude Mallette and Alain Choquette for bearing with us in the many program modifications which have been going on for the last two years. Special thanks also to Jacques Ferland and Jean-Marc Rousseau for believing in this project and putting much time and money into this interactive approach to routing and scheduling.

Bibliography

1) Bodin,L., Golden B., Assad A., Ball M., "Routing and Scheduling of Buses and Crews", Computers and Operations Research, Vol 10, no 2, p 69-211, 1983.

2) Chapleau L., Ferland J., Lapalme G., Rousseau J-M., "A Parallel Insert Method for the Capacitated Arc Routing Problem", Operations Research Letters, Vol 3, no 2, June 1984, p. 95-100.

3) Chapleau L., Ferland J., Rousseau J-M., "Clustering for Routing in Dense Area", European Journal of Operational Research, Vol 20, nol 1, 1985.

4) Meyrowitz N., Van Dam A., "Interactive Editing Systems", Computing Surveys, Vol 14, no 3, Sept 1982, p 321-415.

5) Shneiderman B., "Direct manipulation: A Step beyond Programming Languages", Computer, Vol 16, no 8, p 57-69, august 1983.

6) Cullen F.H., Jarvis J.J., Ratlif H.D., "Interactive Optimization in Distribution Analysis", ORSA/TIMS Joint National Meeting, San Diego, October 1982.

7) Fisher M., Greefield A., Thomson K., "Real World Experience with an Interactive Color Graphic Interface for Vehicle Routing Models", ORSA/TIMS Joint National Meeting, Orlando, November 1983.

8) Babin A., Florian M., James L., Spiess H., "EMME/2: Interactive Graphic Method for Road and Transit Planning", Transportation Research Record, 866, p 1-9, 1982.

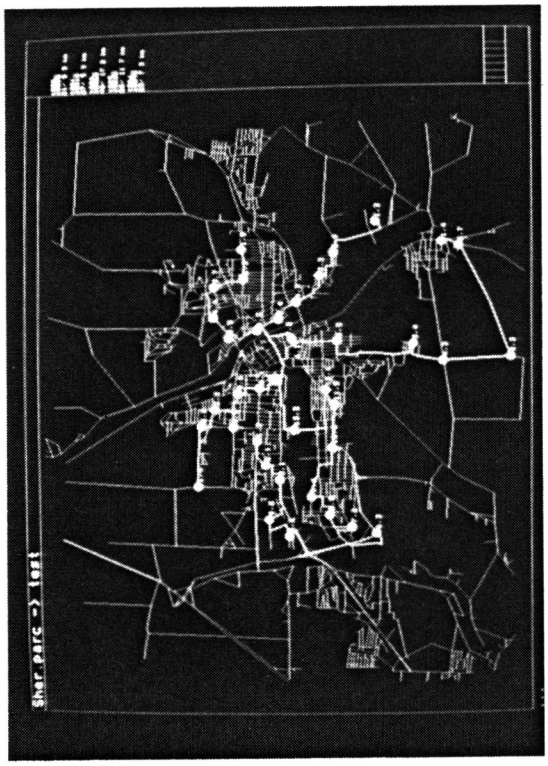

Figure 1

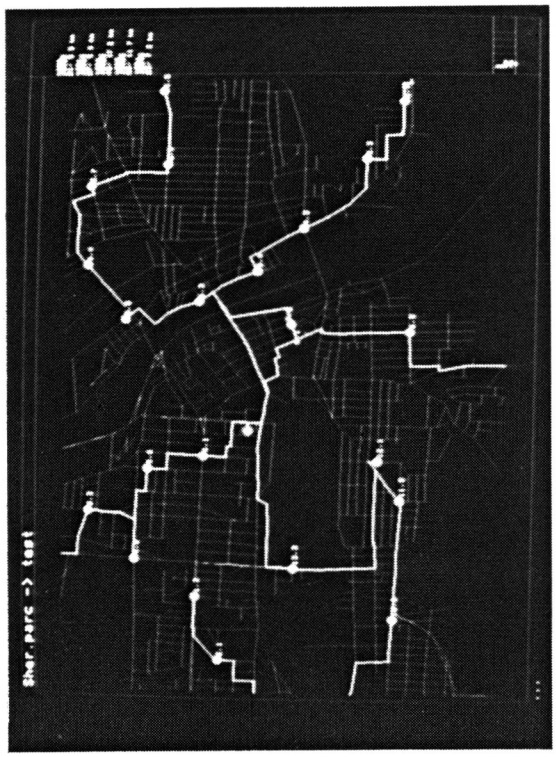

Figure 2