Vehicle Routing and Computer Graphics

M.W.P. Savelsbergh
Centre for Mathematics and Computer Science
Kruislaan 413
1098 SJ Amsterdam
The Netherlands

1. Introduction

Due to the enormous increase in the cost of physical distribution and the economic depression of the last decade, there is a growing demand for decision support systems and optimization methods for distribution management. Although there is a great variety of distribution problems, the basic components are nearly always a fleet of vehicles with fixed capabilities (capacity, speed, etc.) and a set of demands for transporting certain objects (school children, consumer goods, etc.) between specified pick up or delivery points. The *Vehicle Routing Problem* is then to determine which of the demands are assigned to each vehicle and what route each vehicle will follow serving its assigned demand in order to minimize the cost of operating the vehicle fleet. (For further information on the vehicle routing problem see [1].)

Nowadays, a number of software packages is available for solving vehicle routing problems, some of which are based on sophisticated mathematical programming techniques. Reports indicate that the application of these packages results in a considerable saving of distribution costs. Almost all of the currently available packages are batch oriented, by which we mean that the user has no control over the solution process. The program functions as a black box: after specifying all input data the user is provided, on termination, with a final set of routes. In day to day use, it is often necessary to adjust the final set of routes due to practical considerations. Therefore many developers of vehicle routing packages are working on interactive extensions that support post optimization and route manipulation. Although these extensions contribute to the flexibility of the package, we claim that better results can be obtained if interaction between planner and computer starts at the beginning of the solution process. We propose to open up the black box and to combine the strengths of the human planner and the computer in order to obtain the

best possible results.

The intent of this paper is twofold. First we indicate that an interactive approach is well suited for vehicle routing problems and secondly we emphasize the importance of a user interface based on colour graphics. In the next section we shall present a more formal definition of the problem. After that we shall discuss some of the advantages of an interactive approach to vehicle routing and make some remarks on the impact of such an approach on the optimization algorithms to be used. A key part of every interactive system is the user interface. Not only does a clear and user friendly interface add to the acceptance of the system, it is also for a major part responsible for the quality of the results obtained. In the last sections we present our views on a colour graphics user interface for an interactive vehicle routing package and relate some of our experiences with the *Graphical Kernel System* graphics package [6].

2. The Vehicle Routing Problem

We consider the following variant of the vehicle routing problem. A vehicle fleet delivers goods stored at a central depot to satisfy customer demands. Each vehicle has a fixed capacity, and each demand uses a fixed portion of this vehicle capacity. To provide a precise statement of this problem we introduce the following notation.

Constants

K = number of vehicles.
n = number of customers to which delivery must be made. Customers are indexed from 1 to n and index 0 denotes the central depot.
b_k = capacity (weight or volume) of vehicle k.
a_i = size of the delivery to customer i.
c_{ij} = cost of direct travel from customer i to customer j.

Variables

$$y_{ik} = \begin{cases} 1 & \text{if the demand from customer } i \text{ is delivered by vehicle } k \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if vehicle } k \text{ travels directly from customer } i \text{ to customer } j \\ 0 & \text{otherwise} \end{cases}$$

Formulation of the Vehicle Routing Problem

minimize

$$\sum_{ijk} c_{ij} x_{ijk} \tag{1}$$

subject to

$$\sum_i a_i y_{ik} \leq b_k, \qquad k = 1,...,K, \tag{2}$$

$$\sum_k y_{ik} = \begin{cases} K & i=0, \\ 1 & i=1,...,n, \end{cases} \tag{3}$$

$$\sum_i x_{ijk} = y_{jk}, \qquad \begin{cases} j=0,...,n, \\ k=1,...,K, \end{cases} \tag{4}$$

$$\sum_j x_{ijk} = y_{ik}, \qquad \begin{cases} i=0,...,n, \\ k=1,...,K, \end{cases} \tag{5}$$

$$\sum_{(i,j) \in S \times S} x_{ijk} \leqslant |S| - 1, \qquad \begin{cases} S \subseteq \{1,...,n\}, \\ 2 \leqslant |S| \leqslant n-1, \\ k=1,...,K. \end{cases} \tag{6}$$

Two well known combinatorial optimization problems are embedded within this formulation. Constraints (2)-(3) are the constraints of a *generalized assignment problem* and ensure that the depot is part of each route, that every customer is served by some vehicle, and that the load assigned to a vehicle is within its capacity. If the y_{ik} are fixed to satisfy (2)-(3), then for each k, constraints (4)-(6) define a *traveling salesman problem* over the customers assigned to vehicle k.

Although the formulation given above is very compact, the number of variables and constraints involved is enormous. Furthermore the two embedded optimization problems are strongly interwoven and should be solved simultaneously in an optimization algorithm. With the currently available techniques this is impossible for problems of even moderate size. We therefore have to rely on heuristic methods.

One of the first and most used algorithms has become known as the *savings* method [3]. Initially, we suppose that every customer is served individually by one vehicle. Now suppose we link two customers i and j and serve them by one vehicle. This eliminates one vehicle and results in a saving in distance of

$$(2c_{0i} + 2c_{0j}) - (c_{0i} + c_{ij} + c_{j0}) = c_{0i} + c_{0j} - c_{ij}.$$

For every possible pair of delivery points i and j there is a corresponding saving s_{ij}. We order these savings in decreasing order and starting at the top of the list we link the points i and j unless the problem constraints are violated.

A more recent approach is due to FISHER & JAIKUMAR [4]. They separate the embedded generalized assignment problem and traveling salesman problems thus creating a *two phase* method. In the first phase, an *assignment* of customers to vehicles is obtained by solving a generalized assignment problem with an objective function that approximates the cost of the traveling salesman tours of the vehicles through the customers. In the second phase, once the assignment has been made, a *routing* of each vehicle through its set of customers is obtained by solving a traveling salesman problem.

CHRISTOFIDES, MINGOZZI & TOTH [2] developed a branch and bound method, capable of solving small instances to optimality. As is the case with all branch and bound algorithms the effectiveness is entirely dependent on the quality of the bounds used to limit the tree search. The bound they use is

related to the notion of *through-q-routes*. A through-q-route is the least cost route, with total load q and without loops, starting from the depot, passing through a certain customer and finishing back at the depot.

3. Interactive optimization

There are two main reasons why an interactive approach to vehicle routing should be preferred over a batch oriented one. One of them is based on algorithmic considerations, the other on practical ones.

The vehicle routing problem (described above) belongs to the class of NP-hard problems [7]. This class contains problems that are very likely to be inherently intractable [5]. This indicates that it is difficult to solve even small instances of the problem to optimality with a reasonable computational effort. Furthermore, the formulation given in the previous section represents the most elementary version of the vehicle routing problem. Practical problems are often complicated by additional constraints such as maximum route length for the vehicles and time windows for the customers. As a consequence, we should not insist on finding an optimal solution, but instead we should try to find a 'good' solution within an acceptable amount of time. To accomplish this we have to resort to heuristic methods. In designing a heuristic for the vehicle routing problem, we can exploit its geometrical structure in conjunction with the spatial perception capabilities of a human planner. A human planner can guide the computer by indicating subsets that are likely to contain near-optimal solutions and discarding beforehand subsets that are very unlikely to contain near-optimal solutions. This kind of man-machine interaction can substantially decrease the practical complexity of a problem instance.

A second and certainly no less important feature of interactive systems is the flexibility. Below certain aspects of practical vehicle routing are described where flexibility and human decisions are essential:

- A serious drawback of (batch oriented) computerized vehicle routing is the difficulty of taking the flexibility of the data into account. Capacity constraints or time window constraints have a certain degree of flexibility. Take for instance the time window constraints of a delivery point; arriving fifteen minutes after the closing of the window does not necessarily lead to infeasibility of the solution. Some customers will not mind very much as long as it does not happen all the time. Such decisions are easily made by a human planner, based on his knowledge of both the customer itself and its history of the past few days. It is almost impossible though to build such considerations into the software.
- By not allowing any control over the solution process, it is very well possible that in day to day planning the routes differ very much. This might lead to a decrease in travel distance and planned travel time, but, in reality, because of drivers' unfamiliarity with certain areas it might cause an increase in actual travel time. In an interactive approach the planner can weigh which of the aspects is dominant before making a decision.
- In a batch oriented software package it is more difficult to react to certain unpredictable events (road innovations, rush orders arriving when the

planning has already started, etc.).
- Beside the obvious distance, time and capacity utilization criteria other objectives, for which it is difficult to give a precise definition, often play an important role in the planning decisions (evenly spread workload for the drivers, drivers' preferences, etc.).

An interactive planning method for vehicle routing should not only contain sophisticated mathematical tools, it should also contain points where human interaction is possible and beneficial. We have chosen for a two phase method based on the FISHER & JAIKUMAR [4] approach. Especially in the assignment phase a human planner can improve the performance of the method. The objective function that approximates delivery cost is obtained by *seed routes* and the cost of inserting customers into these seed routes. A seed route is an artificial route consisting initially of the depot and a *seed point*, where the seed point indicates an area that is expected to be visited by one vehicle. The seed points are chosen by the planner based on his knowledge and intuition. He can iteratively experiment with different seed routes and immediately see the effect on the cost and routing decisions. In the routing phase the planner can create routes according to his own specific objectives or modify the computer generated ones.

4. COLOUR GRAPHICS USER INTERFACE

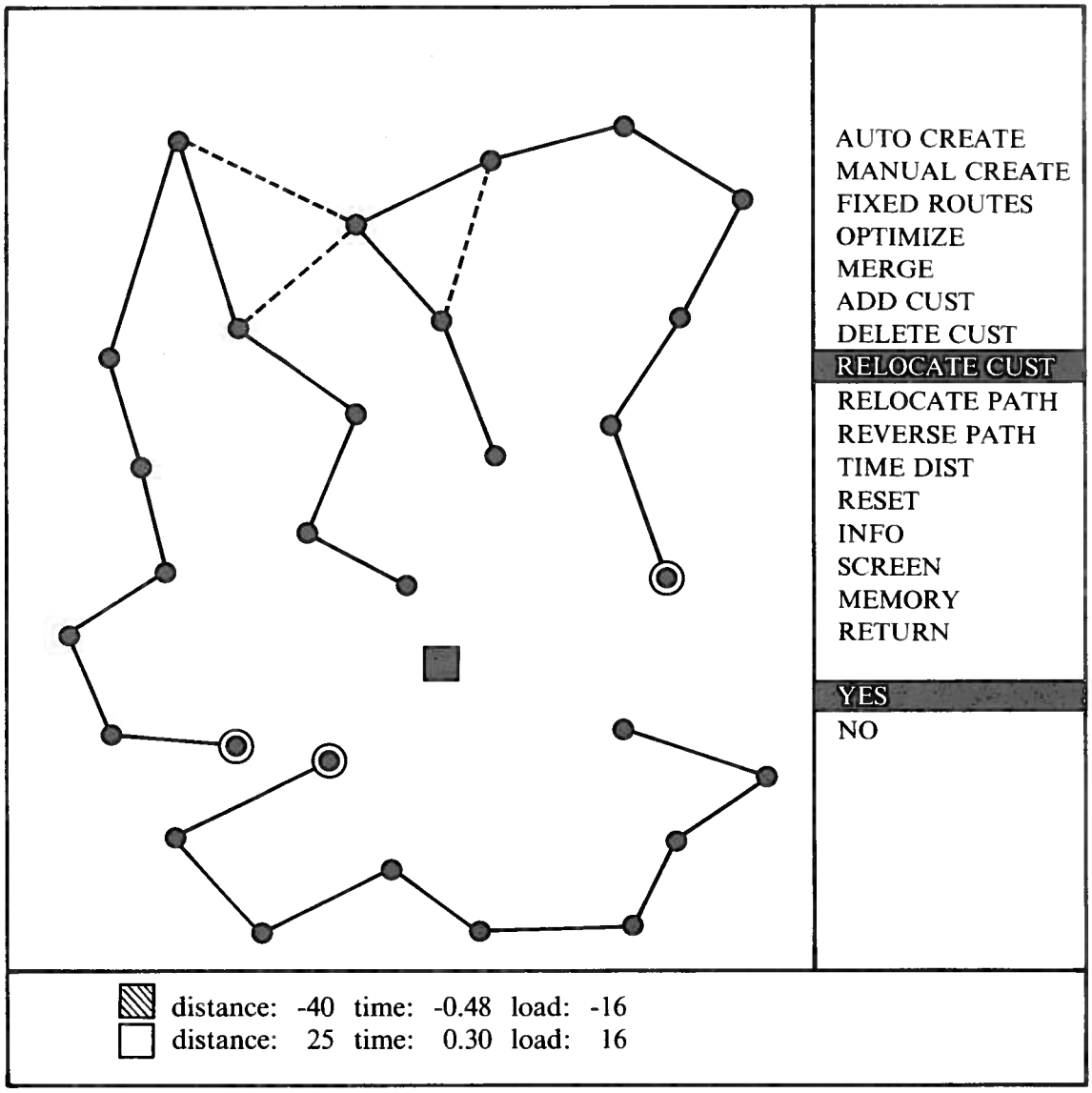
As indicated earlier an interactive approach to computerized vehicle routing is beneficial in increasing the flexibility and performance of the planning system. A very important part of such a system will be the user interface. We believe that a *graphical* user interface is a necessity. The principal argument in favour of the use of graphical displays is their effectiveness in displaying information. Data that would otherwise have to be printed out in numerical form can, with the aid of the display, be used to compose a geometrical representation of the problem which is both clear and informative.

The effect of a graphical user interface will even be stronger if *colour graphics* are used. Colour graphics provide the means to distinguish between the various routes by giving them a different colour. When dealing with only a few routes this advantage is small but when the number of routes increases it becomes more and more a necessity.

But of course a geometrical representation is just one side of the problem. The user should also have the possibility of examining part of the enormous amount of numerical data involved.

We try to keep the screen as steady as possible and therefore divide the screen into three areas:

COMMAND INFO AREA:	This area is reserved for the display of commands;
GRAPHICAL INFO AREA:	This area is reserved for the visualization of the *geometrical* representation;


AUTO CREATE
MANUAL CREATE
FIXED ROUTES
OPTIMIZE
MERGE
ADD CUST
DELETE CUST
RELOCATE CUST
RELOCATE PATH
REVERSE PATH
TIME DIST
RESET
INFO
SCREEN
MEMORY
RETURN
YES
NO
distance: -40 time: -0.48 load: -16
distance: 25 time: 0.30 load: 16

ALPHA-NUMERICAL INFO AREA: This area is reserved for all kinds of *alpha-numerical* information.

We have chosen for a menu-driven interaction mode. Menu-driven interaction has the advantage that the full range of options available to the user at any stage is plainly displayed and it prevents the user from making selections outside this range, and hence solves the problem of erroneous commands.

The commands available to the user are divided into four menus. There are two main menus, the first containing the commands applicable in the assignment phase and the other containing the commands applicable in the routing phase. In addition we have two menus for the utilities available at any stage of the program, one for the set of commands that control the display of information and one for the set of commands that define the contents of the graphical info area. The division of the commands into four subsets has enabled us to reduce the space of the screen needed to display the commands as at any stage at most two of the menus are displayed together.

A subset of the commands in the routing phase provides the planner tools for route manipulation. At any stage the planner can move customers around and adjust routes. Route manipulation takes place by applying interchange techniques [9]. The planner has two possibilities: he can either relocate or reverse the order of a string of consecutive customers. A local change consists of two steps. In the first step the planner specifies the customer(s) he wants to relocate or reverse. This results in the following:

- visualization of the proposed change in the graphical info area (the original situation also remains visible!);
- display of information concerning the change in travel time, waiting time, route length, capacity and feasibility.

It is only in the second step that the planner actually decides whether to carry out the change or not. In this way the planner is able to examine several changes before actually making one and he is protected against input mistakes.

In order to enable the planner to view only part of the graphical info area we provided two screen utilities. The first one enables the user to zoom in on a certain area and the second enables him to change the visibility of routes temporarily. Both these utilities are very useful when the planner is considering local changes.

5. GKS

For the implementation of a colour graphics interface we used the C implementation of the GKS graphics package developed at the CWI [8]. GKS is a basic graphics system for applications that produce computer generated two dimensional pictures on line graphics or raster graphics output devices. It supports man-machine interaction by supplying basic functions for graphical input and picture segmentation.

To be able to use pictures or sub-pictures again during a program's execution GKS supplies a storage mechanism called *segment*. In our application these segments are extremely useful, also for interaction. GKS has six different

logical input device classes which allow the operator to input data to the program. One of them, the PICK logical input device, returns the name of a segment to the application program identifying the sub-picture that was indicated by the operator. The additional *pick identifier* makes it possible to identify picture parts within a segment. This level of naming is provided in GKS to reduce the segment overhead for applications where a great number of picture parts need to be distinguished for input but the need for manipulation is less important. This is exactly the situation in the application we are currently discussing. In the vehicle routing problem there is a large number of customers, who need to be distinguished for input, divided into a relatively small number of routes. Therefore the most natural thing to do is to create a segment for each route and give each of the customers that make up the route a different pick identifier. As an immediate consequence a call of the REQUEST PICK function uniquely determines both the customer and the route of which it is a member.

The *segment attributes*, which allow the application programmer to modify the appearance of a segment (visibility, detectability, highlighting, transformation), also play a significant part in this application. The detectability of a segment gives us the means to control the input and shield the user from erroneous calls of the REQUEST PICK function. In addition the two screen handling utilities are almost entirely based on the segment attributes. The visibility of a segment and thus the visibility of the corresponding route can be changed by altering the visibility attribute. The effect of zooming can be accomplished by adjusting the current segment transformation.

Our choice for a menu-driven interaction mode is supported by the GKS CHOICE logical input device. GKS permits the application programmer to select the most appropriate *prompt* type for each logical input device and one of the possible types for the CHOICE logical input device is a prompt displaying character strings, representing a menu.

GKS knows three operating modes for a logical input device: REQUEST, SAMPLE and EVENT. The current implementation only supports the synchronous REQUEST mode. This means that when the application program requests an input it suspends action until the user activates a trigger to signal that the input is completed. The two remaining modes allow asynchronous input and therefore support more complex interaction. However, the absence of asynchronous input facilities did not restrict the development of the interactive application we had in mind.

The current system is being tested on an AED512, a rather simple raster graphics device of 500 by 500 pixels. This device in no way supports modern workstation features like internal segmentation, transformations and clipping and filling algorithms. All these actions are therefor carried out on the host, a VAX 11/750. This does not mean though that interaction is slow. Because the output primitives used by the application program are not too complex (no complicated fill areas or rasters) and a fast host is used, the speed of interaction is acceptable. In the future a modern screen with GKS based firmware will speed up interaction considerably and the users feedback will take on

spectacular forms.

Although the system is being developed on a large configuration, we will also implement a version that will run on a smaller (stand alone) configuration, probably an IBM PC/AT in combination with a GX-screen.

As GKS is defined device independently, no change other than the substitution of the screen type in the OPEN WORKSTATION function is necessary when the system is moved to another graphical device. This implies a good portability.

6. Conclusion

We have described a number of arguments in favour of interactive approaches to computerized vehicle routing. The arguments are based on considerations concerning algorithmic aspects, flexibility and user friendliness. We have indicated that certain optimization methods developed for the vehicle routing problem are better suited for man-machine interaction than others and described a *cluster first - route second* approach. A very important part of an interactive system is the user interface. We argued that colour graphics should form the basis of such a user interface for vehicle routing packages. A good user interface not only adds to an easy acceptance of the system, but also plays a vital part in the solution process. We also reported some of our experiences with the implementation of a colour graphics user interface with the GKS graphics package. It turned out that for the application described in this paper most of the screen handling can be done with simple basic GKS functions.

7. Acknowledgements

This research was supported by the Netherlands Foundation for the Technical Sciences (STW) through grant nr. CWI-22.0348.

References

1. L.D. Bodin, B.L. Golden, A. Assad, M. Ball (1983). The state of the art in routing and scheduling of vehicles and crews. *Computers and Oper. Res. 10,* 63-212.
2. N. Cristofides, A. Mingozzi, P. Toth (1980). Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxation. *Math. Programming 19,* 255-282.
3. G. Clarke, J.W. Wright (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res. 12,* 568-581.
4. M.L. Fisher, R. Jaikumar (1981). A generalized assignment heuristic. *Networks 11,* 109-124.
5. M.R. Garey, D.S. Johnson (1979). *Computers and Intractability: A guide to the Theory of NP-Completeness,* W.H. Freeman, San Fransisco.
6. ISO (1982). *Graphical Kernel System (GKS) - Functional Description,* Standard ISO/DIS 7942.
7. J.K. Lenstra, A.H.G. Rinnooy Kan (1981). Complexity of vehicle routing and scheduling problems. *Networks 11,* 221-227.

8. D.S.H. ROSENTHAL, P.J.W. TEN HAGEN (1982). *GKS in C,* Report IW 204/82, Mathematisch Centrum, Amsterdam.

9. M.W.P. SAVELSBERGH (1984). *Local Search in Routing Problems with Time Windows,* Report OS-R8409, Centre for Mathematics and Computer Science, Amsterdam.