Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

J.M. Anthonisse, J.K. Lenstra, M.W.P. Savelsbergh

Functional description of CAR, an interactive system for computer aided routing

Department of Operations Research and System Theory Report OS-R8716 September

---

The Centre for Mathematics and Computer Science is a research institute of the Stichting Mathematisch Centrum, which was founded on February 11, 1946, as a nonprofit institution aiming at the promotion of mathematics, computer science, and their applications. It is sponsored by the Dutch Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O.).

Functional Description of CAR, an Interactive System for Computer Aided Routing (Concept, August 1, 1987)

J.M. Anthonisse, J.K. Lenstra, M.W.P. Savelsbergh
Centre for Mathematics and Computer Science, Amsterdam

CAR is an interactive software package which can be used to support operational distribution management. It has been developed at the Centre for Mathematics and Computer Science (CWI) in the period 1983-1986. This document contains a general description of CAR, a detailed description of the interface between CAR and software in its environment (data entry and report generation), and a user manual in the form of a functional description of all available commands.

1980 Mathematics Subject Classification: 90B05, 90C27, 90C50, 68U05. 69C39
Key Words & Phrases: physical distribution, clustering, routing, man-machine interaction, color graphics.

Table of contents

1. CAR: computer aided routing 2
1.1. *Introduction* 2
1.2. *Problem type* 2
1.3. *Solution method* 2
1.4. *User interface and screen* 3
1.5. *Graphical Kernel System (GKS)* 3
1.6. *Input* 3
1.7. *Output* 4
1.8. *Current implementations* 5
1.9. *Modifications and extensions* 5
2. Functional description 5
2.1. *User interface* 5
2.2. *Communication* 6
2.3. *Clustering* 6
2.4. *Routing* 7
2.5. *Information* 10
2.6. *Screen* 11
2.7. *Storage* 12
Appendix 1. Commands 13
Appendix 2. Input and output specification: *address.CAR*, *vehicle.CAR* and *plan.CAR* 14

Report OS-R8716
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

1. CAR: computer aided routing

1.1. Introduction

The costs involved in physical distribution form a substantial component of the total production and distribution costs. This implies a need for a planning system that produces economical routes. In practice, difficulties arise due to the size of the planning situation as well as the number and complexity of the side constraints.

There are many questions of distribution management, which vary with the level within the organization at which they are posed. In the case of *operational planning*, which is considered in this document, the problem situation is specified by a number of vehicles with known capabilities (such as capacities and speeds) and a set of orders to transport objects (such as products and persons) between given collection and delivery points; the plan which is to be determined allocates each order to a vehicle and routes each vehicle through its orders. Higher levels in the organization are concerned with *tactical* or *strategical* planning, and distribution management then relates to issues like fleet composition and depot location.

CAR is an interactive software package which has been developed as a tool to support operational distribution management. CAR enables the user to construct economical vehicle routes and schedules in a simple way. CAR acts as an assistant and advisor to the planner. CAR has a supporting function; it is the user who is in charge and who is responsible for making all the decisions. The use of CAR can lead to savings in physical distribution costs, a faster and simpler planning process, a more constant level of service towards the customers, a lesser dependence on the quality of the human planner, and better management information facilities.

The planning module CAR should form part of larger system. Data entry and report generation do not belong to CAR. Managing the permanent and temporary data bases of vehicles, addresses, distances and travel times, processing incoming orders and generating printed schedules are tasks of the environment.

The development of CAR was financially supported by the *Stichting voor de Technische Wetenschappen* and the *Stichting Mathematisch Centrum*.

1.2. Problem type

CAR is suited for distribution problems with the following characteristics.

- There is a single depot where several vehicles, possibly with different capacities, are stationed.
- The commodity to be transported is homogeneous in the sense that the allocation of commodities to vehicles is restricted only by the vehicle capacities.
- A vehicle can make several trips a day.
- A vehicle has a time window that specifies its availability. (This allows one to impose a maximum route duration.)
- There may be both collections and deliveries. Vehicles depart from the depot with the commodities to be delivered and eventually return to the depot with the collected commodities. Anything collected on the way is transported to the depot. Collections and deliveries may occur in any sequence on the same trip.
- An address may impose restrictions on the capabilities of the vehicle visting it. For example, it may require special loading equipment.
- An address has one or more time windows within which service must take place.
- An address can have a priority, indicating that it must be visited.
- An address is to be visited by at most one vehicle.

1.3. Solution method

The solution method used is based on the 'cluster first-route second' principle. It is a two-phase approach. The first phase clusters the addresses into groups, one for each vehicle; the second phase routes each vehicle through the addresses assigned to it. It is not a purely algorithmic approach, but a heuristic one. Man and machine (or person and processor, if you wish) cooperate in the solution process.

1.4. User interface and screen

An important component of an interactive planning system is the user interface. In developing CAR, we have chosen for a graphical user interface. The principal argument in favor of graphical displays is their effectiveness in showing information. Data that are traditionally presented in alphanumerical form can now be used to create a graphical representation. CAR provides facilities for three types of graphical representations of the problem, oriented towards distance, time and load.

Multicolor graphical representations have an even stronger effect. The use of colors allows for a simple distinction between clusters and routes. For small problems this has a minor advantage, for large problems it is a necessity.

Next to these graphical representations, the user should also be able to consult the relevant part of the huge amounts of alphanumerical data. In order to obtain a steady view of the entire screen, we have divided it into three fixed regions, which contain a *problem representation*, *commands*, and *alphanumerical information*, respectively.

The interaction is menu driven. This has the advantage that at any moment all the feasible commands are visible and it prevents the user from giving infeasible commands. The commands are divided over three primary and three secondary menus. The first primary menu (COMMUNICATION) contains commands that allow the user to change the problem instance and to write partial solutions on an output file. The second one (CLUSTERING) contains commands that constitute the clustering phase, and the third one (ROUTING) those that form the routing phase. The secondary menus provide the supporting functions. The first secondary menu (INFORMATION) contains commands to show alphanumerical information. The second one (SCREEN) contains commands to define the contents of the problem representation region. The user can ask for four different representations: the three graphical ones mentioned above and a purely alphanumerical one. In the spatial representation, he can work with separate parts of the solution by zooming in and making trips invisible. The third secondary menu (STORAGE) contains commands that enable the user to temporarily store the current solution. The clustering of commands into groups leads to an efficient use of the screen, because exactly one primary menu and at most one secondary menu can be active at any time.

1.5. Graphical Kernel System (GKS)

GKS was used for the implementation of the user interface. This is a graphical library for applications that generate two-dimensional pictures on vector or raster graphic terminals.

1.6. Input

The input is read from three files *address.CAR*, *vehicle.CAR* and *table.CAR*, the first two of which are specified in more detail in Appendix 2.

Addresses

CAR requires the following data for each address:
- collection or delivery;
- identification: postal code, description (optional);
- limitations on vehicle types;
- time windows;
- address dependent (un)loading time; summing this and the order dependent (un)loading time results in the total (un)loading time;
- number of orders;

per order:
- order number;
- priority, indicating if it must be included in the plan;
- size;
- (un)loading time.

Vehicles
The following data are required for each vehicle:
- identification;
- capabilities;
- capacity;
- availability.

Remarks
- The notions 'size of an order' and 'capacity of a vehicle' have to be further defined in consultation with the user. They have to be expressed in the same units.
- The 'capabilities of a vehicle' have to be further defined in consultation with the user. They relate to address-vehicle restrictions.
- The 'availability of a vehicle' is expressed in terms of a time window. The window also reflects the opening hours of the depot and the working period of the driver.

Distances, travel times, and coordinates
CAR uses distances, travel times, and coordinates. This information does not belong to CAR and has to be provided by the user in the form of separate tables. CAR uses the postal code as a key to these tables. Whether these tables are compiled on the basis of a road network or in any other way is immaterial.

1.7. Output
The output is written on a file *plan.CAR*, which is specified in more detail in Appendix 2.

Trips
CAR supplies the following data for each trip:
- identification of the vehicle allocated to the trip;
- load;
- load factor;
- length;
- travel time;
- waiting time;
- number of addresses;

per address (where the depot is included as the first and the last address in the trip):
- identification;
- arrival time;
- waiting time;
- departure time.

Overall summary
This contains the following data:
- number of trips;
- average load factor of the used vehicles;
- total length;
- total travel time;
- identifications of the addresses that have not been included in the trips.

Input modifications
All the changes in the input data that have been made during the planning session are listed here.

Remark
All information concerning loads consists of a delivery and a collection component.

1.8. Current implementations

CAR has been written in the C programming language and implemented on three configurations. More implementations will become available in due course.

1.8.1. IBM 6150 (PC/RT) & IBM 5080 (Graphics Display)

The operating system on the IBM 6150 is AIX, IBM's UNIX variant. This is a state-of-the-art configuration with a powerful processor and an advanced screen. We use the C implementation of GKS developed at CWI.

1.8.2. IBM PC/AT & IBM PGA (Professional Graphics Adapter)

The operating system on the IBM PC/AT is MS-DOS 3.2. A 20Mb hard disk, 640Kb internal memory, and a mathematical co-processor are required. We use the C binding of GKS developed by IBM for PC's.

1.8.3. IBM PC/AT & IBM EGA (Enhanced Graphics Adapter)

The specifications are the same as for the previous configuration.

1.9. Modifications and extensions

CAR has a modular structure, which makes it easy to change existing functions and to add new ones.

The precise output and the alphanumerical information displayed on the screen are defined in consultation with the user. This document describes a possible specification.

Extensions that could be considered are:

- multiple depots;
- collection and delivery of the same commodity during a single trip (as in dial-a-ride systems);
- heterogeneous commodities;
- division of an order over several vehicles.

2. Functional description

2.1. User interface

At the start of a planning session, CAR searches the environment for the files *address.CAR*, *vehicle.CAR*, and *table.CAR*, which should contain all the required input data, and processes them. If new orders arrive during a planning session, they can be appended to the file *address.CAR*. At the end of a planning session, CAR leaves a file *plan.CAR* in the environment, which contains data on the trips created, an overall summary, and input modifications, if any.

CAR distinguishes two types of commands. The distinction is visualized on the screen by the presence or absence of an exclamation point in front of the command. Commands preceded by an exclamation point call functions that correspond to algorithms, the results of which are generally not predictable by the user. These are the functions that generate or modify a plan automatically. Commands without an exclamation point have an easily predictable result. These are the functions that allow the user to generate or modify a plan manually. Appendix 1 lists all available commands.

One enters all commands by choosing from menus or specifying addresses with the mouse, light pen, or joystick, depending on the configuration. An address is specified by indicating the associated point on the screen. If the postal area corresponding to this point contains just this address, then one is done; otherwise, CAR lists all the addresses in the area in a menu and the user indicates the desired address.

Some commands ask for the identification of a cluster or a trip. In the spatial representation, a cluster or a trip is specified by indicating an address that belongs to it. In the three other representations, a color palette is used in which each cluster or trip has its own color.

When CAR is started, the user sees a spatial representation of the problem and the CAR 'supermenu'. This contains five commands: COMMUNICATION, CLUSTERING, ROUTING, RESTART, and STOP. It serves to select one of the three primary menus, to restart the planning (without having to process the files *address.CAR* and *vehicle.CAR* again), and to end the planning session.

2.2. COMMUNICATION

This primary menu contains the commands that enable the user to modify the problem instance and to write partial solutions to the file *plan.CAR*. CAR maintains a record of all modifications of the original data and writes these on the file *plan.CAR* at the end of the planning session.

ADD

effect: Data of addresses that have been added to the file *address.CAR* are processed.

DELETE

input: Address.
effect: The specified address is deleted from the set of addresses.

CHANGE

input: Address.
effect: The input data of the specified address can be changed.

WRITE

input: Trip.
effect: The specified trip is written on *plan.CAR*. The addresses of this trip and the vehicle allocated to it are no longer taken into consideration.

INFORMATION

effect: The secondary menu **INFORMATION** is activated.

SCREEN

effect: The secondary menu **SCREEN** is activated.

2.3. CLUSTERING

This primary menu contains the commands to generate clusters in the first phase of the planning process. A cluster is a collection of addresses with a vehicle that will visit them. The user can form and modify clusters manually or ask the system to construct them automatically, as he sees fit. One can also reenter the clustering phase from the routing phase. In that case, the existing trips define the clusters, but the orderings that define the trips disappear.

For each cluster, there is an address which serves as a seed point around which the cluster is grown. A seed point is marked with a circle around the address in question, in the color of the cluster.

An address that does not belong to any cluster is called *free*.

!SEEDS

input: Integer *m*.
effect: A set of *m* new seed points is created on the basis of distances, order sizes, address-vehicle restrictions, time windows, and existing seed points. A vehicle is allocated to each new seed point, which defines an upper bound on the total order size in the corresponding cluster.

!CLUSTERS

effect: The free addresses are assigned to a seed point, as far as the restrictions permit it.

!FIXED CLUSTERS

effect: When this function is called, it is assumed that no seed points exist. CAR searches the environment for the file *fixed.CAR*. This file contains a number of trips that have been earlier defined by the user; these are now interpreted as clusters. For each of these clusters, its inter-

section with the collection of addresses is determined. In case the intersection is nonempty, it constitutes a new cluster; the system generates a seed point and allocates a vehicle.

SET SEED
input: Free address; vehicle.
effect: The specified address will serve as a seed point. The specified vehicle is allocated to the seed point.

ERASE SEED
input: Seed point.
effect: All assignments of addresses to the specified seed point are canceled. The seed point is erased.

CHANGE VEHICLE
input: Seed point; vehicle.
effect: The specified vehicle is allocated to the specified seed point. This cancels the previous allocation.

DELETE ADDRESS
input: Address.
effect: The assignment of the specified address is canceled.

ADD ADDRESS
input: Free address; seed point.
effect: The specified address is assigned to the specified seed point.

INFORMATION
effect: The secondary menu **INFORMATION** is activated.

SCREEN
effect: The secondary menu **SCREEN** is activated.

STORAGE
effect: The secondary menu **STORAGE** is activated.

2.4. ROUTING

This primary menu contains the commands to generate routes in the second phase of the planning process. A route consists of one or more trips. A trip is a collection of addresses that have been put in the order in which they have to be visited, starting and finishing at the depot. The user can form and modify trips manually or ask the system to do so automatically, as he sees fit.

A trip is represented on the screen by linking each pair of successive addresses by a line segment. The depot is the only address that can occur in more than one trip; it occurs in each trip. To avoid congestion around the depot on the screen, we have decided to delete the line segments of which the depot is an end point. The first address on a trip is marked with a circle around it, in the color of the trip.

An address that does not belong to any cluster or trip is called *free*.

There are commands for trip optimization and trip manipulation. An important aspect of trip manipulation is the presentation: the user first sees the effect of the action he proposes, and then decides to accept it or not. Even if the proposed action results in an infeasible trip, acceptance is allowed.

Next to the spatial representation, the user can switch to three other representations of a single trip: a bar chart indicating the schedule in time, a bar chart indicating the load of the vehicle, and a complete alphanumerical survey. These representations are secondary to the spatial one because they relate to just a single trip. In all representations, trip optimization and manipulation are possible.

!TRIP

input: Cluster.

effect: For the specified cluster, a short feasible trip is generated. If it appears to be impossible to include all addresses of the cluster in a feasible trip, the assignment of some addresses is canceled.

note: If the user indicates the depot, this function is carried out for all trips. This is consistent with the fact that the depot is an address that belongs to all trips.

!FIXED TRIPS

effect: When this function is called, it is assumed that no clusters exist. CAR searches the environment for the file *fixed.CAR*, which contains a number of trips that have been earlier defined by the user. For each of these trips, its intersection with the collection of addresses is determined. In case the intersection is nonempty, a new trip is defined by visiting the addresses in the intersection in the same order as in the fixed trip; the system allocates a vehicle.

CREATE

input: Number of addresses; (vehicle).

effect: The user creates a trip in a stepwise fashion. If the address that is specified first is free, then a new cluster will be defined: a vehicle has to be allocated to it, and only free addresses may be assigned. If the first address does belong to a cluster, then a vehicle has already been allocated, and only addresses belonging to this cluster or free addresses may be assigned. The user specifies addresses in the order in which he wants them to be visited. He removes the last address of the partial trip by specifying the next to last address. He completes the trip by specifying the depot, or by specifying the last address twice in a row. During this process the user is informed about the feasibility of an extension and of the length, travel time, waiting time and load of the trip created thus far.

!IMPROVE

input: Trip.

effect: Optimization techniques are applied to shorten the specified trip. Information about changes in length, travel time and waiting time of the trip is provided.

note: If the user indicates the depot, this function is carried out for all trips.

!MERGE

input: Two trips from different routes.

effect: Optimization techniques are applied to shorten the specified trips by the exchange of addresses. It is possible that both trips are merged into one. Information about changes in length, travel time, waiting time and load of the trip is provided.

CHANGE VEHICLE

input: Trip; vehicle.

effect: The specified vehicle is allocated to the specified trip. This cancels the previous allocation.

COUPLE

input: Two trips.

effect: The specified trips are concatenated. In contrast to the function **!MERGE**, this function does not change the ordering of each trip. The largest of the vehicles allocated to the original trips is allocated to the newly created route. Information about the new departure and arrival times at the depot for each trip is provided.

DECOUPLE
input: Trip.
effect: The specified trip is decoupled from possibly preceding and succeeding trips. It becomes a separate route again.

DELETE ADDRESS
input: Address.
effect: The specified address is deleted from the cluster to which it belongs. It becomes a free address. In case the specified address belonged to a trip, information about changes in length, travel time and load of the trip in question is provided.

!ADD ADDRESS
input: Free address.
effect: The specified address is inserted at the best feasible point in one of the existing trips. Information about changes in length, travel time and load of the trip in question is provided. In case each insertion is infeasible, the address remains free.

ADD ADDRESS
input: Free address; two addresses that occur successively in a trip.
effect: The specified free address is inserted between the two successive addresses. Information about the feasibility of the insertion and about changes in length, travel time and load of the trip in question is provided.

RELOCATE ADDRESS
input: Address belonging to a trip; two addresses that occur successively in a trip.
effect: The first specified address is relocated between the other two. Note that the trips from which these addresses are taken may be the same. Information about the feasibility of the relocation and about changes in length, travel time and load of the trips in question is provided.

RELOCATE PATH
input: Two addresses belonging to the same trip; two addresses that occur successively in a trip.
effect: The path which is specified by the first two addresses is relocated between the other two. Note that the trips from which these addresses are taken may be the same. Information about the feasibility of the relocation and about changes in length, travel time and load of the trips in question is provided.

REVERSE PATH
input: Two addresses belonging to the same trip.
effect: The path specified by the two addresses is reversed. Information about the feasibility of the reversal and about changes in length and travel time of the trip in question is provided.

FREEZE
input: Two addresses belonging to the same trip.
effect: The path specified by the two addresses is fixed. This means that the path will not be changed by algorithmic commands (i.e., those preceded by an exclamation point).

DEFROST
input: Two addresses belonging to the same trip.
effect: The fixation of the path specified by the two addresses, caused by a FREEZE command, is canceled.

INFORMATION
effect: The secondary menu **INFORMATION** is activated.

SCREEN
effect: The secondary menu **SCREEN** is activated.

STORAGE
effect: The secondary menu **STORAGE** is activated.

2.5. INFORMATION

This secondary menu can be activated from the primary menus **COMMUNICATION**, **CLUSTERING** and **ROUTING**. It contains commands that enable the user to examine information about addresses, vehicles, clusters, trips, and the overall plan. These functions will be further defined in consultation with the user.

ADDRESS
input: Address.
effect: The following information about the specified address is provided:
- identification;
- limitations on vehicle types;
- time windows;
- (un)loading time;
- priority;
- size;
- arrival time;
- waiting time;
- departure time.

VEHICLE
input: Vehicle.
effect: The following information about the specified vehicle is provided:
- corresponding cluster or trip;
- identification;
- capabilities;
- capacity;
- availability.

CLUSTER
input: Cluster.
effect: The following information about the specified cluster is provided:
- allocated vehicle;
- load;
- load factor;
- number of addresses.

TRIP
input: Trip.
effect: The following information about the specified trip is provided:
- allocated vehicle;
- load;
- load factor;

- length;
- travel time;
- waiting time;
- number of addresses.

PLAN
effect: An overall summary is given:
- number of trips;
- average load factor;
- total length;
- total waiting time;
- total travel time;
- number of free addresses.

Next, a summary of each trip is given:
- load factor;
- length;
- travel time;
- waiting time.

2.6. SCREEN

This secondary menu can be activated from the primary menus **COMMUNICATION**, **CLUSTERING** and **ROUTING**. It contains commands that define the problem representation region.

When CAR is started, the user sees a spatial representation of the problem. In this representation, he can work with separate parts of the problem by zooming in and by making trips invisible. Next to the spatial representation, the user can switch to three other representations of a single trip: a bar chart indicating the schedule in time, a bar chart indicating the load of the vehicle, and a complete alphanumerical survey.

ZOOM
input: Rectangle, specified by its lower left and upper right corners.
effect: The contents of the rectangle form the new contents of the problem representation region.

FULL
effect: In the problem representation region, all trips are made visible in their original sizes.

(IN)VISIBLE
input: Trip, or the depot.
effect: If a trip is specified, it is made invisible. If the depot is specified, all trips are made visible.

SEE DISTANCE
effect: Transition from the current problem representation to the one oriented towards distance.

SEE TIME
input: Trip.
effect: Transition from the current problem representation of the specified trip to the one oriented towards time.

SEE LOAD
input: Trip.
effect: Transition from the current problem representation of the specified trip to the one oriented towards load.

SEE TEXT
input: Trip.
effect: Transition from the current problem representation of the specified trip to an alphanumerical survey.

2.7. STORAGE

This secondary menu can be activated from the primary menus **CLUSTERING** and **ROUTING**. It contains commands that enable the user to temporarily store the current solution.

STORE
effect: The current set of clusters or trips is stored. A set that may have been stored before is erased.

RETRIEVE
effect: The current set of clusters of trips is erased. The set that has been stored before is retrieved.

SWAP
effect: The current set of clusters or trips is exchanged with the one that has been stored.

Appendix 1. Commands

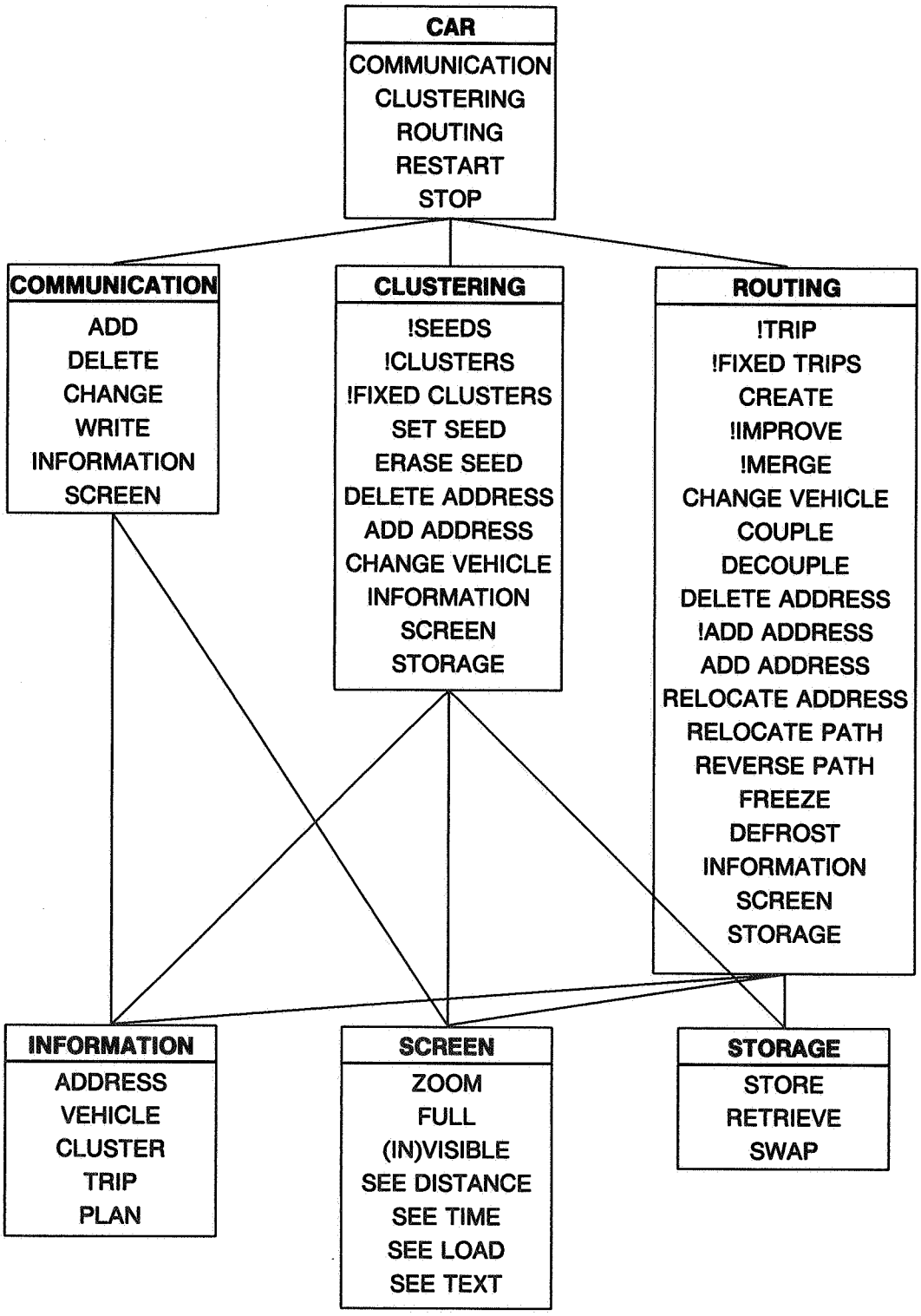

Figure 1. Relations between menus.

Appendix 2. Input and output specification: *address.CAR*, *vehicle.CAR* **and** *plan.CAR*

A data entry system and report generator do not form part of CAR. Managing permanent and temporary data bases, processing incoming orders, and printing trip schedules are tasks of the environment and not of CAR. The input is read from three files *address.CAR*, *vehicle.CAR* and *table.CAR*; the output is written on a file *plan.CAR*. A detailed specification of these files is given in Table 1.

Next to the files *address.CAR* and *vehicle.CAR*, CAR also uses the file *table.CAR*, which contains a table of distances, travel times and coordinates for the relevant postal areas, and the file *fixed.CAR*, which contains a set of trips.

TABLE 1. DETAILED SPECIFICATION.

file				#characters
address.CAR	per address[1]	collection/delivery		1
		identification		40
		limitations		8
		time windows		4/4/4/4
		(un)loading time		4
		#orders		3
		per order	order number	10
			priority	1
			size	5/5
			(un)loading time	4
vehicle.CAR	per vehicle[2]	identification		9
		capabilities		8
		capacity		6/6
		availability		4/4
plan.CAR	per trip	vehicle identification		9
		load		6/6
		load factor		3/3
		length		6
		travel time		4
		waiting time		4
		#addresses		3
		per address[3]	identification	16
			arrival time	4
			waiting time	4
			departure time	4
	#trips			2
	average load factor			3/3
	total length			6
	total travel time			6
	#included addresses			4
	#free addresses			4
	per free address	identification		16
	modifications			

1. First all delivery addresses, then all collection addresses.
2. In order of decreasing capacities.
3. In the order in which they are visited, with the depot as the first and last address.

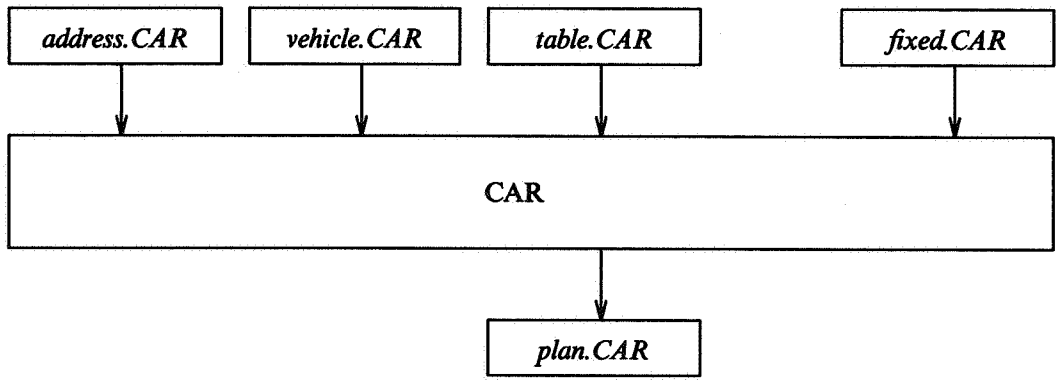

Figure 2. Input and output files.