Study of sidewalks in the UPC campuses of Barcelona through route planning techniques

Hernane Borges de Barros Pereira
Advisor: Professor Lluís Pérez Vidal
Universitat Politècnica de Catalunya
pereira@lsi.upc.es

May 11, 2000

Abstract

In this work we will present a study with respect to the sidewalks in the UPC campuses zone of Barcelona by ArcInfo GIS. We have done a specialized bibliographic reference work and an analysis of a first version of the route planning application, which were used as support material. The main results are (1) a theoretical study on route planning (2) updating of the digital cartography, (3) implementation of the Dijkstra's algorithm and some comments about Dantzig's and A* search algorithms, (4) analysis, design and development Router 2.0 application, and (5) CD recording with all the updated information about this research.

Key words: Sidewalk, Geographic Information System, ArcInfo, Route Planning, Route Optimization, Dijkstra's algorithm, Dantzig's algorithm, A* search algorithm

Contents

1 Introduction

Route planning is a technique that allows the selection of paths that an agent needs to know to arrive at its destination. A wide range of this research field is related to transport management in Geographic Information Systems - GIS - which will be the principal approach of our research.

Nevertheless, we comment on others fields such as robotics (motion planning), nomadic computing, simulation, networks and integrated circuits in which route planning techniques can be applied to solve specific problems. We believe that this general view is a contextual frame for our research.

The first goal of this work has been the recovery of data saved in several places. But research proceeded and many other tasks became necessary. We have perceived the need to develop an application that has been implemented and summarized all the work done.

During the data recovery process we have identified the Router application developed on ArcInfo (ArcInfo Macro Language - AML) with support from C. We have tried to run this application, but we failed. We have started the analysis of the what had been made and we have proposed to develop a new version: the Router 2.0 application. This has implicated in studying the AML language and rewriting of the Dijkstra's algorithm in C language. In parallel, we have done a bibliographic study about research fields where the route planning was concerned. In addition, we have updated of the acquired digital cartography.

In sum, the main results are (1) a theoretical study on route planning, (2) updating of the digital cartography, (3) implementation of the Dijkstra's algorithm and some commentaries about Dantzig's and A* search algorithms, (4) analysis, design and development Router 2.0 application, and (5) CD recording with all the updated information about this research.

1.1 Organization of work

This technical report has begun with a brief description of route planning problem and the general overview about our research. The second section will present the route planning analysis, which is composed of (1) some definitions, (2) summary of related works, (3) graph overview, (4) some comments on the Dijkstra's, Dantzig's and A* Search algorithms, and (5) some research issues in the route planning research field. The third section will comment

on the conducted work, that is, (1) data recovering, (2) obtention of the updated digital cartography, (3) Router 1.0 application analysis, (4) analysis and development of the Router 2.0 application, and (5) CD recording with all the updated information about this research. The conclusions will are provided in the fourth section followed by suggestions for future research activity in the fifth section. Finally, the sixth section will finish the technical report with the appendix that is composed of information about (1) Router 1.0 application sources, (2) Router 2.0 application sources, and (3) more information about digital cartography used to carry out this research.

1.2 Conventions used in this technical report

To clarify information presented in this technical research report, the following typographical conventions have been used:

- All text that relates to code (e.g. C or ArcInfo commands) is shown in a `verbatim typography`
- When we refer to directories and subdirectories in Section 3.6, these are shown with *directory* and *..subdirectory* respectively.

2 Route planning

There is a wide range of fields where the techniques and algorithms of route planning can be used and in many cases are indispensable (see Figure 1). The idea of this section is to present an overview on the planning of good routes. In addition, we survey some related works on this research field and a graph overview, comment on the Dijkstra's, Dantzig's and A* Search algorithms, and enumerate some research issues.

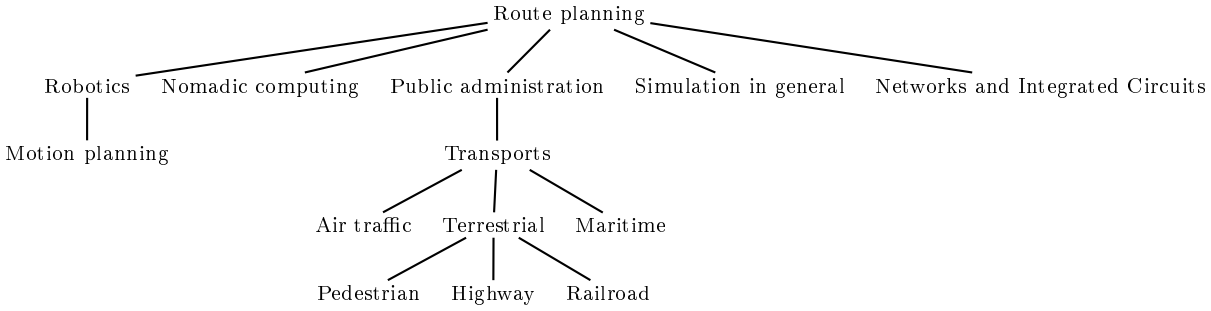


Figure 1: General framework. Applications of route planning techniques.

According to Laurini and Thompson (1992), one of the most important spatial problems is route planning. To plan a route is not a generic task, but it rather depends on the context in which we are addressing our efforts with the goal of developing an application that offers support to the problem in question.

Route planning is a technique that allows the selection of path in which a mobile object or agent (e.g. vehicle or pedestrian) can move from an initial point (origin, source) at time t_1 to a final point (destination, target) at time t_2 (where $t_1 + t_2$ is time constraint) considering all the possible static and dynamic variables which can be considered constraints (e.g. an irregular land or the interdiction of a road due to any incident), and that are related to the area in question (i.e. a specific context). Economic aspects, although not evident, can be taken in consideration too.

Much research is being developed to solve the problems of definitions of methods and algorithms applied to route planning, as much in the spatial field as in the temporal one (*scheduling*), but according to Lapalme et al. (1992), many constraints and aspects should be taken into consideration.

2.1 General concepts

For better contextualization, we present some terms that are used in this research field.

- Agent: A person or mobile object (e.g. a robot and/or aircraft) that will go from a origin-point to a destination-point;
- Route: A way taken by the agent from an initial-point (origin, source) to a final-point (destination, target);
- Best route: The best way taken by agent from an origin-point to a destination-point. The best route can basically be stated by time and distance criteria. Many issues (e.g. constraint and impedance problems) are taken into account to decide what is the best route;
- Initial-point: The origin point. From where the agent starts the path;
- Final-point: The destination or target point. To where the agent finishes the path;
- Constraint: A restriction, obstacle or limitation of liberty to the process of moving. Some authors including Pellazar (1998), Kingsley, Klesczewski and Smith (1998), Chen (1996), Iakovou, Douligeris, Li, IP and Yudhbir (1999), and others comment on the constraints, which depend on the specific context;
- Impedance problems: These problems are effective impeding of a path or route, which we can solve them through alternate ways, therefore, the spatial database must currently be updated (indispensable requirement);
- Costs: Although it appears redundant, cost is what a thing costs which, in our case, is the best route according to the selected algorithm, that is the price paid for the route calculus. Moreover, the cost model must take into account the constraints and impedance problems. Some authors such as Prieto, Rueda, Quintana and L. (1997), Pellazar (1998), Lee and Fishwick (1995), Gudaitis, Lamont and Terzuoli (1995) and others survey models for the cost calculus module;
- Static variables: These variables or factors are not changing (e.g. the absolute distance between two points);

- Dynamic variables: Different from the static variables, these variables are changing intermittently (e.g. the speed of the agent when going to destination-point throughout the selected path). These variables are directly linked to impedance problems which provide them with new values[1].

In the following Figure 2, we survey a conceptual map of an area to route planning.

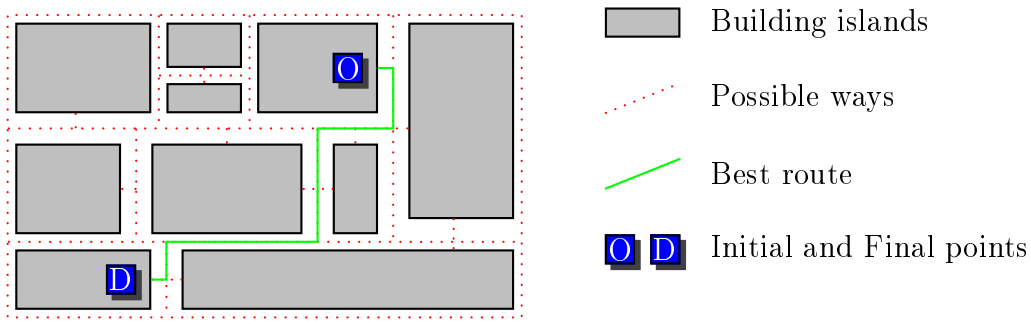


Figure 2: Conceptual map of an area to route planning.

Therefore route planning is composed of some sub-fields, such as scheduling and shortest path problems. We will concentrate our research on the best and shortest path problem. Chen (1996) presents a good and concise geometric definition for the shortest path problem:

> "(...) Given a d-dimensional space strewn with obstacles, compute a path P connecting two points p and q such that the path P does not intersect the interior of any obstacle and that the total length of P (based on a certain metric such as the Euclidean) is minimized. (...)".

Furthermore, Chen (1996) comments on the high cost problems of incorporating geometric shortest path into practitioners' application systems. The author observes that some factors are obstacles for implementation, such as the dependence on "(...) other sophisticated geometric procedures and data structures (e.g. visibility, Voronoi diagrams, triangulation, space point-location, space decomposition, etc.). (...)".

[1] For another approach, see Gudaitis et al. (1995, p. 172) who propose two categories of variables which are "considered when planning a real mission route: (1) Ridged variables and (2) Flexible variables"

2.2 Related works

There are many application fields of route planning techniques (in our research subject, the best and shortest route), such as robotics, nomadic computing, transport administration, simulation, mobile telephony, computer architecture, remote exploration, and so on.

Here we present a bibliographic summary and the state of the art on the topic. We will also comment some works carried out by several researchers in diverse areas.

Lee and Fishwick (1995) state that route planning is a subset of the *decision making* which, in turn, influences many research areas such as simulation, software engineering and artificial intelligence.

2.2.1 Robotics

Hwang and Ahuja (1992) present an extensive paper on motion planning and they comment on general issues in gross-motion planning. Their work is concentrated on autonomous motion planning and they comment that the "(...) Motion planning can be static or dynamic, depending on whether the information on the root's environment is fixed or updated. (...)" (p. 220). This concept is very important to define constraint in route planning. Also, they present a subsection on search methods of the best route[2].

Beslon, Biennier and Hirsbrunner (1998, p. 39) "(...) argue that reactive agents can perform the conveying task very efficiently as long as they are able to coordinate their displacements and to cooperate in such a way as to optimize their paths. (...)". They present a distributed implementation-oriented architecture "based on beacon-identified loading points" and they comment that "thanks to this simple scheme, agents are able to perform multi-robot distributed path-planning". In addition, they analogize the social insects and their proposal because there are some aspects of management of some ant species with respect to "establish optimized route paths" (p. 41).

Wang, Mehdi and Gough (1998) use the services of the spatial database by object-oriented methodology to define the best route for autonomous guided vehicles (AGV). The authors comment on three strategies of search (forward search, backward search and mixed forward and backward search). They avoid searching into a circle due to the AGV method carries out search into

[2]For mode details, see Hwang and Ahuja (1992, p. 237-239)

a graph. In addition, they comment on "what is the best criterion", which can be (1) shortest path (i.e. distance criterion), (2) shortest time (i.e. time criterion), and (3) cost criterion based on human imitation.

2.2.2 Nomadic computing

According to La Porta, Sabnani and Gitlin (1996, p. 3), "(...) the nomadic computing environment is characterized by mobile users that may be connected to the network via wired or wireless means, many of whom will maintain only intermittent connectivity with the network. (...)". In addition, they present a study on techniques for addressing challenges for nomadic computing and their relative effectiveness, and they examine how those techniques (in this case, caching techniques and network-based proxies) "are used to assist in routing and improve the performance in networks that support mobility".

Other authors such as Gerla and Tsai (1995) and Maffeis, Bischofberger and Mätzel (1996), develop research in this field.

2.2.3 Transports

We identify some commercial applications that offer support to the planning and optimization of routes. The GEOROUTE ^{TM} application of Giro is a proposal of the graphical software package that permits the management of information associated with street networks (Lapalme et al., 1992).

In their article, Gudaitis et al. (1995), present in general terms, criteria related with traveled distance and data taking by radar that are combined with a simple cost function to assess routes and they use a specific parallel A* algorithm (for general details, see Subsection 2.3.3). The authors define the mission route planning problem - MRP. According to the mission in question, an assessment of way criteria will be carried out and a specific route will be selected. According to Gudaitis et al. (1995, p. 171), "the first step towards finding a solution technique for MRP is developing computer models to represent the real-world.". Their costs MRP model for the realization of route assessment is based on reality, therefore they classify the involved factors in such situations in two groups: (1) rigid variables and (2) flexible variables.

Goto, Matsuzaki, Kweon and Obatake (1986) present the CMU Sidewalk Navigation System architecture that consist in a navigation system able to control the displacement of a vehicle, that has the following modules: (1)

route planning, (2) local path planning, (3) vehicle driving, (4) perception, and (5) map data. This system uses the CMU campus (the test site with many characteristics such as ways and intersections) to carry out tests.

Strothotte, Fritz, Michel, Raab, Petrie, Johnson, Reichert and Schalt (1996) comment on the MoBIC project and system, a new travel aid which "(...) assists travellers in exploring information about the localities to which they wish to travel. (...)" (p. 140). The authors used some test participants with different abilities who even have no experience with computer. "Participants appreciated the fact that they could control the flow of information from the system" (p. 144). They put an end to their paper, commenting on the user requirements research in order to improve the mobility of the travellers.

Iakovou et al. (1999) use the services of a network low model in order to solve the marine routing problem of the hazardous materials. They comment that "(...) for the marine environment, links/routes are only roughly defined and are estimates based on the assumption that the vessels will follow the shortest distance between the port of origin and the receiving port (...)" (p. 36). In addition, the authors illustrate through (1) a hypothetical problem and (2) a case study (i.e. marine transportation of oil products) the complexities of the problem.

2.2.4 Simulation in general

Many of the above authors' researches use some simulation processes, methods or environments. Thus, the presented sub-sections are not to each other excluding.

Lee and Fishwick (1995) present a research based on automated route planning and use simulation techniques to support their research. The authors argue that "simulation-based planning is useful for route planning under various conditions including uncertain locations and events with potential adversarial activity" (p. 1087). In addition, Lee and Fishwick (1995, p. 1093) comment that the SBP method has several advantages, such as (1) it is "able to capture the effects more accurately and completely", (2) using "standard simulation analysis methods in tuning the simulation models to closely reflect actual processes", (3) to enable "the evaluation of the plan as a natural result of simulating different models in the system", (4) to allow "finer tuning of the execution process", (5) ease extendability "the set of models to include additional properties in testing plans".

Wiley and Keyser (1998) present a study on the transport simulation. They use analysis techniques and a data collection properly designed to offer support to the transportation incident management decisions.

Kannan, Martinez and Vorster (1997) present a framework for the integration of the project and process levels taking into account dynamic strategies with the following goals: (1) to formalize the work planning process of the earth, and (2) to develop a methodology to the representation and evaluation of the plan.

Martin (1999) comments many simulation levels as support tools in the railroad projects, and he describes with details some characteristics such as (1) the principal topics considered in the railroad design/project, (2) the train and its engine, and (3) the power sources supply. The author also present some applications (RailPlan^{TM}, Animator^{TM}, PowerPlan^{TM} and TrainPlan^{TM}) used for the management and the simulation of a railroad network. The author considers that the simulation will have a fundamental role in railroad development.

Kingsley et al. (1998) present a study on a logistics model of coast guard operations and they comment on a simulation model that allows to carry out measurement of the effectiveness "of a new buoy tender design with respect to logistics support" (p. 755). The authors consider that the environment is very important to plan aid visits. In addition, they present a route planning module, which is called by the simulation model, and that takes into account some constraint imposed on context, such as depth of water, distance, fuel, daylight, and so on.

2.2.5 Networks and Integrated circuits

The route planning can be used to networks. Bestavros and Matta (1997) present a study on Virtual Circuit (VC) routing algorithms which are used to achieve the Quality of Service (QoS) requirements in the high integrated networks. The authors consider that the "routing algorithms are responsible for the selection of the particular route - which should have sufficient resources to satisfy the application packets (or cells) to reach their destination. (...)" (p. 183). In addition, they comment on a routing scheme based on the route selection that may be expected to happen, that is "probabilistic selection of route" which, considering their demonstration, is better than the traditional least-loaded routing.

Others authors such as Dommety, Veeraraghavan and Singhal (1997) (research based on optimization of the route in mobile Asynchronous Transfer Mode - ATM networks), Medhi (1995) (research based on an approach for dimensioning ATM-based networks which use to the virtual path concept), Prieto et al. (1997) (research based on "performance-driven placement algorithm for automatic layout generation of analog IC's"), Masson, Escassut, Barbier, Winer and Chevallier (1991) (research based on association between floor planning and routing, and chip assembly; b. architecture and capabilities of the CHEOPS System), Wu and Marek-Sadowska (1995) (research based on non-conventional routing optimization scheme using low complexity algorithms), and Decasper, Dittia, Parulkar and Plattner (1998) (research based on new architecture which allows *plugins* - code modules - "to be dynamically added and configured at run time"; moreover, plugins have got efficient mapping, high perform and easy integration with custom hardware features) are carrying out route planning research on network and circuit fields.

2.3 The algorithms on the best route

Several papers on route planning comment on the algorithms used to solve the path planning, shortest route, route optimization, and so on. Authors such as Hwang and Ahuja (1992), Derniame and Pair (1971), Prasad (1990), Lee and Fishwick (1995), Gudaitis et al. (1995), Gerke (1998) and Pellazar (1998) comment on the application of one or more methods used to solve the problems of their researches.

However, Pijls and Kolen (1992) present a good paper in which a general framework for shortest path algorithms is commented, including among others Dijkstra's and A* search algorithms. The authors discuss about "that all algorithms are special cases of one algorithm in which some of the non-deterministic choices are made deterministic, termination and correctness can be proved by proving termination and correctness of the root algorithm." (p. 1)

2.3.1 Graph overview

We are not interested in to present an extensive study of graph, but to further a brief summary on graph that comes to contribute as support material (knowledge necessary) for the understanding of the algorithms that will be commented on the next subsections.

Directed and undirected graphs are natural models of arbitrary relationships among data objects (Aho, Hopcroft and Ullman, 1983). However, this is a general definition. If we consider our case study (i.e. sidewalks in the UPC campuses of Barcelona), we can define a graph with all the paths from one UPC building to all the others UPC buildings. Furthermore, if we use a basic mathematical definition, we have $G = (V, A)$, where

- V is a set of vertices, nodes or points (in our case, UPC buildings);
- A[‡] is a set of edges or arcs (in our case, (sub-)ways between UPC buildings);
- An arc is an ordered pair of vertices (Aho et al., 1983; Pijls and Kolen, 1992);

[‡]This is a flexible notation. We can find in the literature, authors as Aho et al. (1983) who use E notation and as Pijls and Kolen (1992) who use A notation. We prefer to use A notation because we are familiarized with ArcInfo GIS which the *arc* word is often used.

- s is the source vertex (i.e. origin or initial-point);
- t is the target vertex (i.e. destination or final-point);
- d is the distance between two vertices;
- D is a set of distances between the vertices;
- p is one of the possible paths which is composed of a sequence of vertices;
- P is a set possible paths $(p_1, p_2, p_3, ..., p_n)$;
- l is the length (distances' sum) of the path p,
- L is a set lengths of P.

In the following Figure 3, we show a example of a graph.

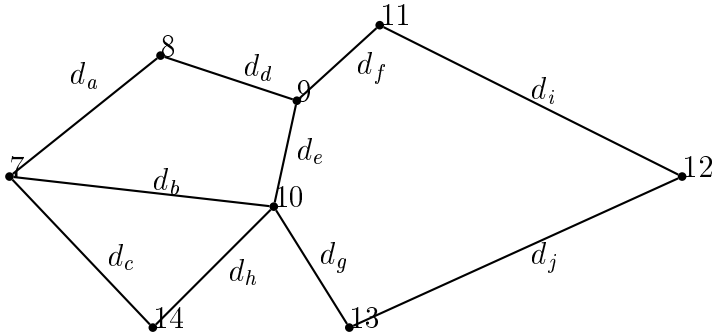


Figure 3: Example of a graph representation.

Notice that according to Figure 3 and the above basic mathematical definition, we have:

- $V = 7, 8, 9, 10, 11, 12, 13, 14$;
- $A = (7,8), (7,10), (7,14), (8,9), (9,10), (9,11), (10,13), (10,14), (11,12), (12,13)$
- $D = d_a, d_b, d_c, d_d, d_e, d_f, d_g, d_h, d_i, d_j$
- $s = 7$
- $t = 13$
- $p_x = (7,14), (14,10), (10,13)$

- $l_{p_x} = d_c, d_h, d_g$

In the following sub-sections, we present the three algorithms used to solve the route planning problems: Dijkstra's, Dantzig's and A* search algorithms.

2.3.2 Dijkstra's Algorithm

One of the Edsger W. Dijkstra's contributions to computer science, including the algorithm that solves the problems of finding the shortest path in a graph (from an initial-point to a final-point), named Dijkstra's algorithm.

In the literature, we have found some practical researches such as Prasad (1990, p. 585) who uses the Dijkstra's algorithm as one of the proposed heuristics, Hwang and Ahuja (1992, p. 237-239) who present a subsection on search methods and they comment that "if the shortest path is desired", we should use the A* search or Dijkstra's algorithm which "(...) of $O(n^2)$ is the most efficient. (...)" (p. 239), Aho et al. (1983) use Dijkstra's algorithm to explain directed graph, and Pijls and Kolen (1992) argue, according to their framework, that Dijkstra's algorithm is a instance of A* algorithm with heuristic estimate function equal to zero.

To proceed, we present Dijkstra's algorithm summary[4]:

1. It finds the nodes connected to the destination point;
2. It puts them in a queue;
3. It assigns each of them the cost-destination (weight of the edge connecting it to the destination);
4. The expanded destination node is marked as the parent of the nodes in the queue;
5. It selects the node with the smallest cost-to-destination (node$S$);
6. It deletes the node$S$ from the queue;
7. It finds nodes connected to node$S$;
8. It computes the cost-to-destination for each node by adding the weight of the connecting edge to the cost-to-destination of the parent node;

[4]This algorithm has been cited according to Hwang and Ahuja's explanations in Hwang and Ahuja (1992, p. 239)

9. If any of the children nodes has a previously computed cost-to-destination greater than the newly computed one:
   (a) its cost is changed to the new one;
   (b) the cost of all its descendants are updated;
10. It repeats the above process until all the nodes are assigned a cost;
11. It returns the minimum-cost sequence of edges between the origin and the destination point that has been retrieved by following the parents beginning from the origin node.

The basic algorithm is presented in C code[5] (Aho et al., 1983, p. 205). Notice that this representation is the adjacency matrix.

```
(01) void Dijkstra(int origin, GRAPH g)
(02) {
     ...
(03) S = origin;

(04) for(i=1; i <= n; i++)
(05)    D[i] = C[origin][i];

(06) for(i=0; i < (n-1); i++) {
(08)       choose a vertex w in V-S such that D[w] is minimum;
(09)       add w to S;
(10)       for each vertex v in V-S
(11)          D[v] = min(D[v], D[w]+C[w,v]);
(12) }

(13) return;
(14) }
```

We have implemented Dijkstra's algorithm using adjacency list (for a complete implementation in C, see Appendix 6.2.4). In the following code, we survey the brief implementation which takes into account the adjacency list representation.

```
(01) void Dijkstra(int origin, GRAPH g)
(02) {
     ...
(03) S[0] = origin;
(04) D[origin] = 0;
(05) for(i=1; i <= n; i++) {
(06)    D[i] = L[origin][i];
(07)    predecessor[i] = -1;
(08) }
(09) fill the PQ (priority queue) with all vertices from g (starting from
     g[origin])
(10) j = 1;
(11) while(PQ != NULL) {
```

[5]Some line are written in natural language

```
(12)    choose a vertex w in PQ such that D[w] is minimum;
(13)    S[j] = w;
(14)    for each vertex v in V-S
(15)       if(D[v] > D[w]+C[w,v]) {
(16)          D[v] = D[w]+C[w,v];
(17)          predecessor[v] = w;
(18)       }
(19)    j++;
(20) }
(21) return;
(22) }
```

2.3.3 A* search

We have comment in the subsection above that according to the framework presented by Pijls and Kolen (1992), Dijkstra's algorithm is a instance of A* algorithm with heuristic estimate function equal to zero. In addition, Rabuske (1992) comments that the A*'s and Dijkstra's algorithms are fundamentally the same.

Hwang and Ahuja (1992, p. 237-239) present a subsection on search methods and they comment that "if the shortest path is desired", we should use the A* search or Dijkstra's algorithm.

A* search method summary[6]:

1. "There must be an underestimate of the cost from the current configuration to the goal[7]. A straight-line distance gives such an underestimate (obstacles only increase the path length).";
2. It sums all "costs from the start configuration to the current configuration and the cost-to-go" (*total cost*);
3. "*Total cost* gives a lower bound on the actual cost";
4. It "generates the children of the research configuration whose *total cost* is the smallest";
5. "When a solution is found, it does not generate the children of any reached configuration whose *total cost* is greater than the *total cost* of the solution";
6. "The better the cost-to-go approximates the actual cost, the more A* can prune out partially generated paths, making it more efficient".

[6]This algorithm has been cited according to Hwang and Ahuja's explanations in Hwang and Ahuja (1992, p. 237)

[7]Goal is what we call destination in this report

Lee and Fishwick (1995, p. 1091) comment that they used the A* search method "to reduce the amount of computation", but it depends on their ability to "build a heuristic function which could estimate the cost of the remaining route".

Gerke (1998, p. 2464) comments on the application of the A* search method "(...) to a graph representation configuration space. (...)", which is one of a few presented optical approach for route planning.

Gudaitis et al. (1995, p. 171) argue about the A* algorithm, it "is computationally designed and implemented to provide an effective method for evaluating the various environmental models". In addition, the authors present a new parallel A* algorithm in pseudo-code format and they comment that, according to results of the "speedup experiments", the new parallel A* algorithm had best performance than the sequential A* algorithm.

2.3.4 Dantzig's Algorithm

Derniame and Pair (1971) present a study on route planning problems using graphs and they comment the following Dantzig's algorithm, which calculates the whole minimal distances among the points of a graph.

```
(01) void Dantzig(GRAPH g, int limite)
(02) {
        ...
(03)    omega = 100000;
(04)    zr = 100000;
(05)    infinito = 99999.99;
(06)    lr = 99999.99;

(07)    Initializing the matrix di[i][j] with infinite number, except for when i is equal to j, which its
        value will be zero

(08)    fill the the matrix di with all distances between the vertices from GRAPH g

(09)    for (i=1;i<limite;i++)
(10)    {
(11)       for (z=0;z<=i-1;z++)
(12)       {
(13)          b[z]=di[i][z];
(14)          c[z]=di[z][i];
(15)       }
(16)       for (j=0;j<=i-1;j++)
(17)       {
(18)          w=di[i][j];
(19)          if (w < infinito)
(20)          {
(21)             for(z=0;z<=i-1;z++)
(22)             {
(23)                MulValues(j, w, z, di[j][z], zr, lr);
(24)                AddValues(zr, lr, pt[i][z], b[z], pt[i][z], b[z]);
```

```
(25)            }
(26)         }
(27)         w = di[j][i];
(28)         if ( w < infinito )
(29)         {
(27)            for(z=0;z<=i-1;z++)
(28)            {
(29)               MulValues(i, w, j, di[z][j], zr, lr);
(30)               AddValues(zr, lr, pt[z][i], c[z], pt[z][i], c[z]);
(31)            }
(32)         }
(33)      }

(34)      for (j=0; j<=i-1; j++)
(35)      {
(36)         MulValues(j, b[j], i, di[j][i], zr, lr);
(37)         AddValues(zr, lr, pt[i][i], di[i][i], pt[i][i], di[i][i]);
(38)      }

(39)      for (z=0;z<=i-1;z++)
(40)      {
(41)         for (j=0;j<=i-1;j++)
(42)         {
(43)            MulValues(i, c[z], j, b[j], zr, lr);
(44)            AddValues(zr, lr, pt[z][j], di[z][j], pt[z][j], di[z][j]);
(45)         }
(46)         for (z=0;z<i;z++)
(47)         {
(48)            di[i][z] = b[z];
(49)            di[z][i] = c[z];
(50)         }
(51)      }

(52) // End of file dantzig2.c
(53) return;
(54) }
```

The Dantzig's algorithm runs basically four loops. The first loop is presented between the lines 09 and 33 and its goal is to calculate the superior and inferior parts of the general matrix. The second loop verifies the principal diagonal of the matrix (lines 34 and 38) and the third loop carries out the final calibration of the matrix (lines 39 and 51).

Furthermore, the Dantzig's algorithm uses the following functions to update the total distances of all the vertices. First, the *MulValues()* function is used to identify the existence of a connection between two nodes and if the connection exist the function adds it to the path until the current origin. The new destinations and the length are passed to the *AddValues()*, the second function.

```
void MulValues(z, l, z1, l1, zr, lr)
{
   if ( l1 == infinito || z == omega && l == 0 )
   {
       zr = z1;
```

```
        lr = l1;
    }
    else
    {
        zr = z;
        lr = l + l1;
    }
}
```

In addition, the *AddValues()* function is used to identify if the new calculated path by *MulValues()* function is shorter than the last one. If the condition is true then the last path will be replaced by new calculated path, otherwise there is not changing.

```
void AddValues(z, l, z1, l1, zr, lr)
{
    if ( l <= l1 )
    {
        zr = z;
        lr = l;
    }
    else
    {
        zr = z1;
        lr = l1;
    }
}
```

These two functions are called jointly four times as we can see in the lines (23-24, 29-30, 36-37, and 43-44) of the *Dantzig()* function presented at the beginning of this section. In addition, after implementation of the Dantzig's algorithm we have perceived that this algorithm does only calculate the total distances of all the vertices.

2.4 Research issues

There are several open problems with respect to route optimization which is an active area of research. We enumerate below a list of some research issues on route optimization area.

1. Geographical 3D analysis: What are the procedures for this analysis taking into account the route planning task?
2. How to introduce impedance control in route planning algorithms?
3. What are the implications of temporal GIS with respect to the route planning procedures?
4. What are the actions that affect the procedure in a desired route?
5. How to control the cost (technical, economic, political, etc.) to solve transportation problems?
6. Can we define a general model to solve the above commented problems?

3 Finished work

In this section, we comment on our conducted work, that is, (1) data recovering, (2) obtention of updated digital cartography, (3) Router 1.0 application analysis, (4) analysis and development of the Router 2.0 application, and (5) CD recording with all the updated information on this work.

3.1 Data retrieval

First, we have started compiling all the existent material on many media types that hold route planning contents. The majority of this material was of no interest to our work. The only data set that interested us was the Router 1.0 application backup saved in a streamer media.

3.2 Obtention of updated digital cartography

The digital cartography has been acquired of the *Oficina de Atención al Ciudadano* on the Plaza San Miquel in Barcelona. Stated before, our interest is focused in the UPC campuses of the university zone of Barcelona. In the selection process of the interested area, we have identified that five sub-areas were necessary.

3.2.1 Digital cartography analysis

In the acquisition of the five sub-areas we have requested all the existent layers (for more details, see Appendix 6.3):

- *Urban layer*
- *Way layer*
- *Cadastral register layer*
- *Altitudinal register layer*
- *Topography layer*
- *Signs layer*
- *Urban installation layer*
- *Boundary layer*

In the following Table 1, we survey the relationship between the features and sub-areas.

We present in Figure 4 the position of the urban map sub-areas, that is a visual result of the urban features after to proceed to the creation of the ArcInfo coverages and joining operation.

Table 1: Coverage names.

Layer features	Sub-areas				
	1	2	3	4	5
URB - *Urban* **VIA** - *Way* **CDS** - *Cadastral register* **ALT** - *Altitudinal register* **TOP** - *Topographical register* **RET** - *Signs* **MOB** - *Urban installation* **AMB** - *Boundary*	L174	L178	M171	M175	M176

3.2.2 Creation of the ArcInfo coverage

The acquired digital cartography was in microstation format (".DGN" - design file) and it was necessary to convert it to ArcInfo format. The ArcInfo GIS has two commands used to perform this operation. "IGDSINFO reads the IGDS file and display information about it. IGDSARC converts a design file into an ArcInfo coverage." (E.S.R.I., 1998).

The following examples are extracted from our procedures, therefore the presented values here are applied to the digital cartography of the university zone case of Barcelona. We start with the IGDSINFO example:

```
Arc: IGDSINFO
Usage: IGDSINFO <igds_file> {STANDARD | OPTIONAL}

Arc: IGDSINFO l174urb
```

Now, we present the IGDSARC example applied to our case:

```
Arc: IGDSARC
Usage: IGDSARC <in_igds_file> <out_cover> {2D | 3D}
  {xmin} {ymin} {xmax} {ymax} {OVERLAP | INSIDE} {text_width}

Arc: IGDSARC L174urb.DGN L174urb
Enter layer names and options (type END or $REST when done)
```

```
============================================================
Enter 1st layer and options:  urban1 * * * * * item
Enter 2th layer and options:  end

Do you wish to use the above layers and options ? (Y/N) y

After translation, a summary of what was translated is displayed on your screen.

679 elements placed:
Cell: 38
Line: 637
Text Element: 42

Range of elements placed:
(25547.900,82361.142) to (25905.072,82827.144)

X delta = 353.021, Y delta = 466.002

Master Units of Measure:  m
```

For each one of the sub-areas, we must apply the IGDSARC command, that is:

```
Arc: IGDSARC L178urb.DGN L178urb

Arc: IGDSARC M171urb.DGN M171urb

Arc: IGDSARC M175urb.DGN M175urb

Arc: IGDSARC M176urb.DGN M176urb
```

Once the ".DGN" files have been converted to an ArcIno coverage (i.e. physically a sub-directory of hard-disk), we have to create the whole topology composed of line and point (BUILD command), and polygon (CLEAN command) attributes.

Note: For more details, see *Creating an ARC/INFO coverage from a design file* section from E.S.R.I. (1998).

3.2.3 Joining the ArcInfo coverage

We should join the five sub-areas into one area. For this, we have to create the five sub-areas topology (CLEAN and BUILD commands). The CLEAN command creates the polygon topology (".PAT" ArcInfo format file) and the BUILD command creates the point and line topologies (".AAT" ArcInfo format file), thus we must apply the following commands for each one of the converted sub-areas[8]:

```
Arc: CLEAN L174urb L174urbok 0.1

Arc: BUILD L174urb line
```

After this, we use the MAPJOIN command that "combines up to 500 adjacent coverages containing polygon or networked features into one coverage and recreates topology" (E.S.R.I., 1998). The following example has been used for our case:

```
Arc: MAPJOIN
Usage: MAPJOIN <out_cover> {feature_class...feature_class | template_cover}
  {NONE | FEATURES | TICS | ALL} {clip_cover}

Arc: mapjoin urbs poly
Enter coverages to be MAPJOINed (Type END or a blank line when done):

======================================================================
Enter the first coverage: l174urbok
Enter the second coverage: l178urbok
Enter the third coverage: m171urbok
Enter the fourth coverage: m175urbok
Enter the fifth coverage: m176urbok
Enter the third coverage: <cr>

Done entering coverage names (Y/N)? y

Do you wish to use the above files (Y/N)? y
```

[8] That is, we must create the topology of the L178URB, M171URB, M175URB and M176URB, and so on.

```
MAPJOINing files...
Arc:
```

Now, we create the line topology of the urbs coverage (BUILD command) because the polygon topology has been created by MAPJOIN command. Finally, we have the coverages (urbs, vias, cdss, alts, tops, rets, mobs and ambs) of the our study area. The following figure represents a visual result of the above procedures:

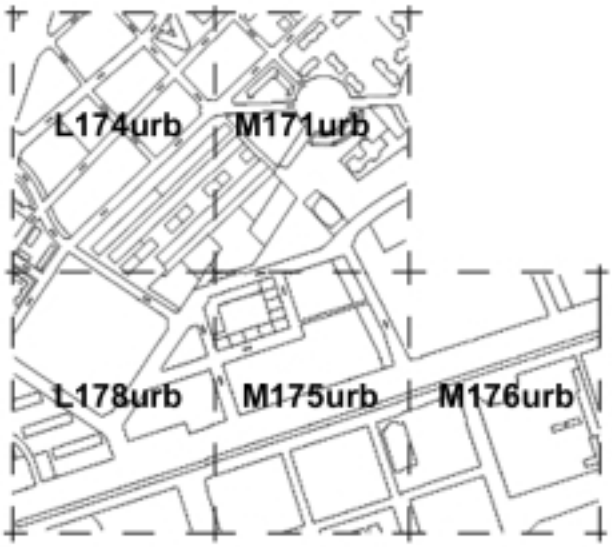


Figure 4: Position of the sub-areas.

3.3 Spatial data base update

The updating procedure is cumbersome, because also textual data (e.g string of names), geometrical and topological data are necessary.

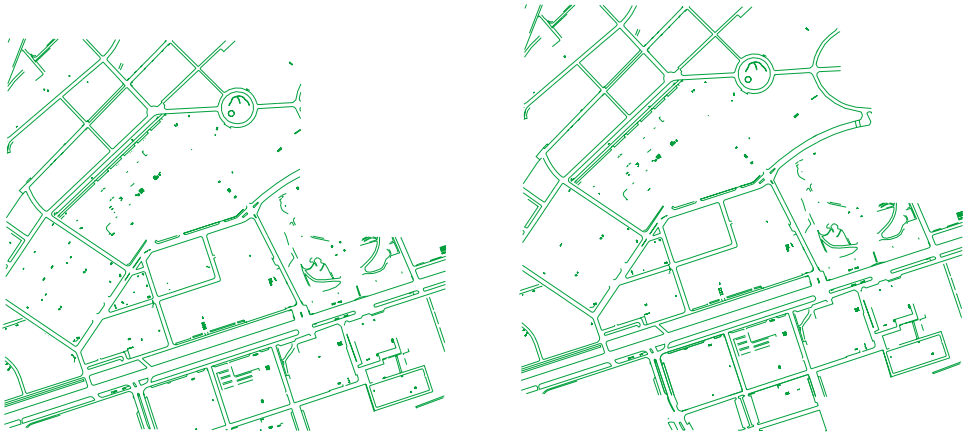
The Spatial data base - SDB - of the UPC campuses placed in the university zone of Barcelona was not updated (e.g. some new streets have been built and this information was not registered in the SDB).

After the creation of the ArcInfo coverage and joining the ArcInfo coverage procedures, we have observed that there were none of the topological necessary data. Thus, we have started to draw and to edit the topological data that did not exist. The updating procedure immediately was necessary for urbs, vias and cdss coverages (urban layer, way layer and cadastral (buildings) layer, respectively).

(a) Updated urban coverage

(b) Updated and edited urban coverage

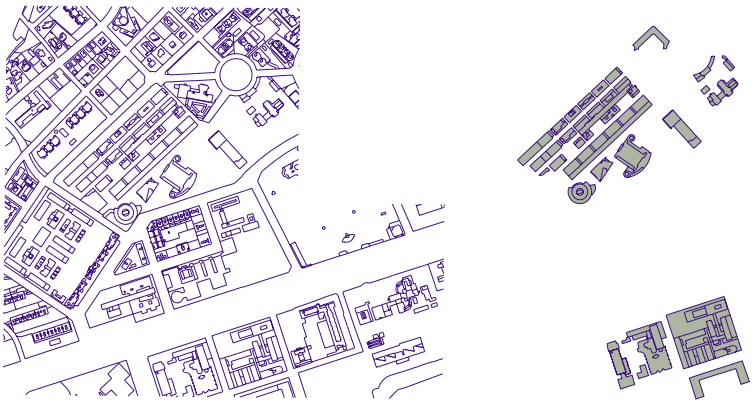
Figure 5: Differences between the urban coverages.



(a) Updated way coverage

(b) Updated and edited way coverage

Figure 6: Differences between the ways coverages.



(a) Updated cadastral coverage

(b) Updated and edited cadastral coverage of the UPC buildings

Figure 7: Differences between the cadastral coverages.

3.4 Router 1.0 application analysis

The Router 1.0 application uses a geographic information system, and its main objective is to define the best way between two UPC buildings (origin and destination points) placed in the South and North campuses of the university zone of Barcelona.

The Router 1.0 application has been developed in *ArcInfo Macro Language* - AML. The C language has been used to calculate the best route by Dijkstra's Algorithm, and this module produces a temporary file (resultat.TMP) that is used by AML module. We present the Router 1.0 application structure in the following framework (Figure 8).

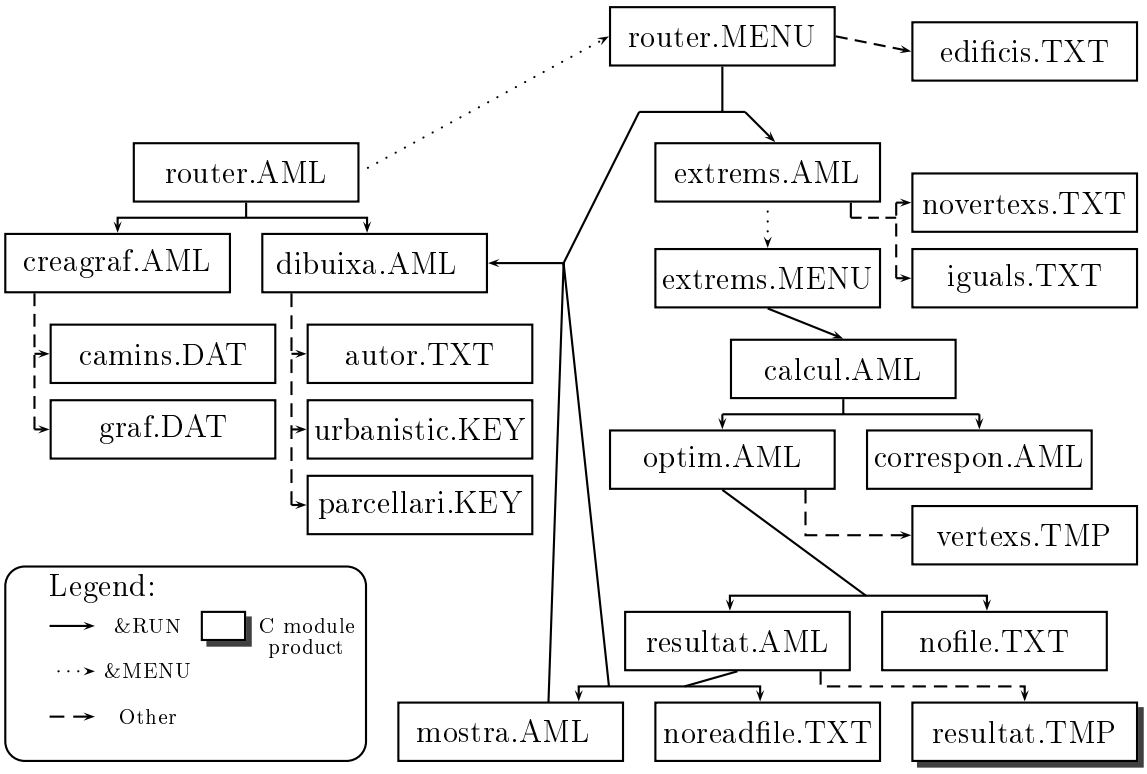


Figure 8: Router 1.0 application source connections.

3.5 Router 2.0 application development

However, the Router 1.0 application did not run properly. There were a several problems in the module that calculated the best route. Therefore, we have decided to implement a new version of the Router 1.0 application that "runs" without problems; it was named Router 2.0.

First, we have defined the initial architecture (Figure 9) of the Router 2.0 application which was composed of three modules:

- MANAGEMENT Module: It is responsible for the application management, that is this module must be able to maintain in execution all the application modules;
- MAP Module: This module is responsible for the maps' control. The application uses this module to the managing of all map functions, such as draw, redraw, zoom and so on;
- ROUTE Module: This module is responsible for the best route calculus. It uses the Dijkstra's algorithm which is written on C.

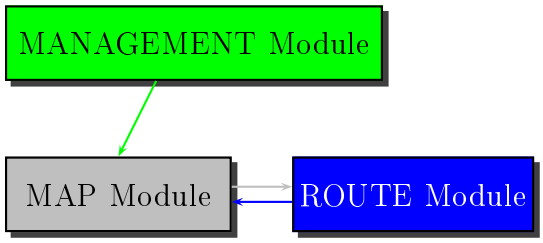


Figure 9: Initial architecture of the Router 2.0 application.

Due to the new implementations (e.g. the printing, zoom and draw control modules) the complexity of the Router 2.0 source connections has become larger. In the following figure, we present the Router 2.0 application source connections.

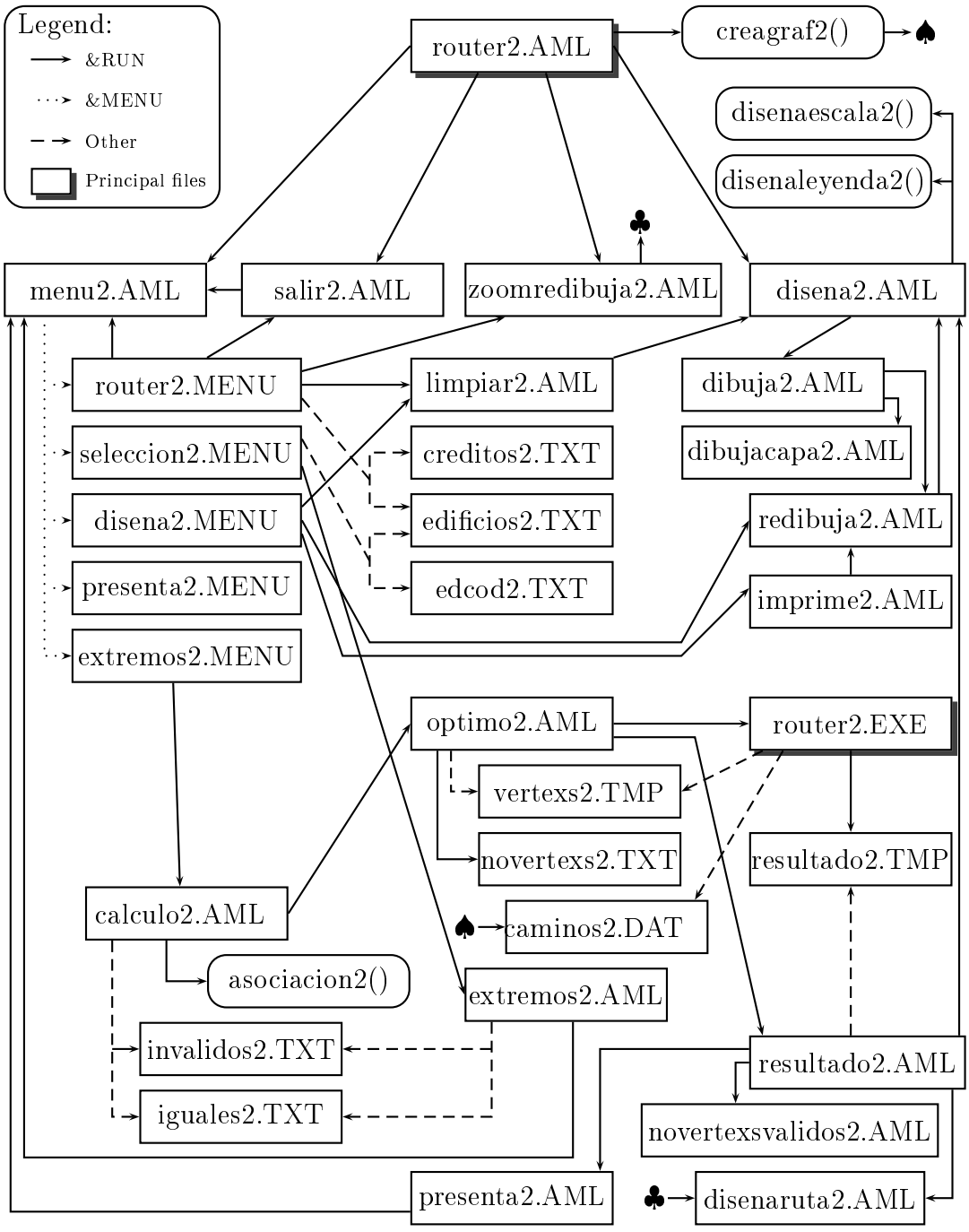


Figure 10: Router 2.0 application source connections.

When we start Router 2.0 application the main window is displayed, which is composed of three basic elements: (1) the ArcPlot area, (2) the main menu (*Estudio de Planeamiento de Rutas a los Campi de la UPC*) and (3) draw-coverage tool (*Disena las Capas*), and showed in Figure 11.

Figure 11: The main Router 2.0 application window.

We can display and delete the wished coverages for using of check boxes in the draw-coverage tool, which is a important tool to handle visual geometric data. In Figure 12 we present a example of visualizing of three coverages of the university zone of Barcelona.

In addition, draw-coverage tool has five buttons. The first button is named *Limpiar* (Clear), which its function is to clear the ArcPlot area. The second button is called *Redisenar* (Re-draw). We can use this button to re-draw all the selected coverages. The third button is named *Imprimir* (Print). This button makes possible to have paper copy of the displayed map. In the lower position, we have the cancel button, named *Cancelar*. Finally, the clear-route button, called *Limpiar Ruta* is displayed and its function is to clear the last calculated route.

To choose a specific route, we have to select the first option of the main menu, *Rutas* (Short-path). On the other hand, if we select this option a

Figure 12: The main Router 2.0 application window. Three selected layers.

sub-menu will be displayed, which has two options (see Figure 13). The first option is named *Definir ruta* (Give path) is the core of the Router 2.0 application.

Figure 13: *Rutas* sub-menu of the Router 2.0 application.

In following Figure 14 and Figure 15, we show the window sequence for selecting and confirming of the wished path starting from a origin point to destination point.

After calculation the Router 2.0 application displays the calculated short-path by Dijkstra's algorithm. A red line is draw in the map connecting the two given points. In addition, a little window is displayed with longitudinal and time information. In following Figure 16, we show the result windows.

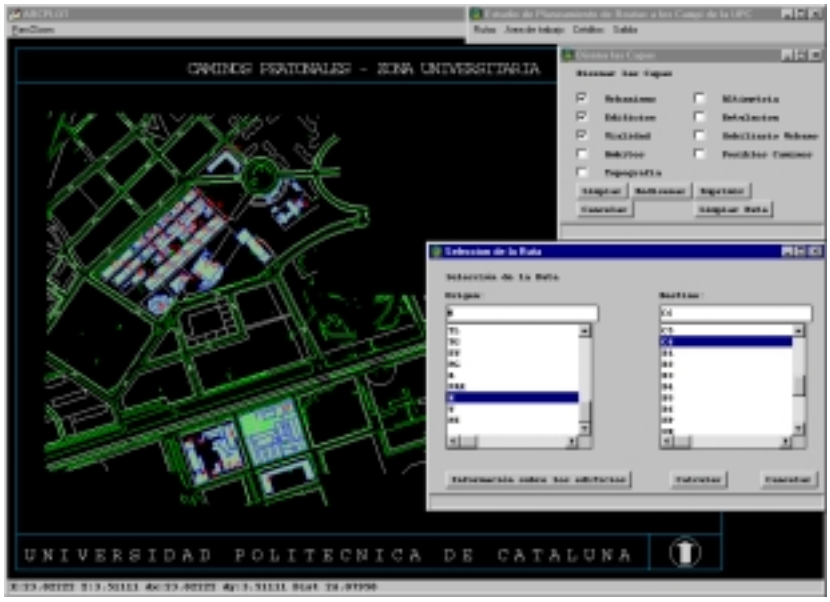


Figure 14: Route selecting window.

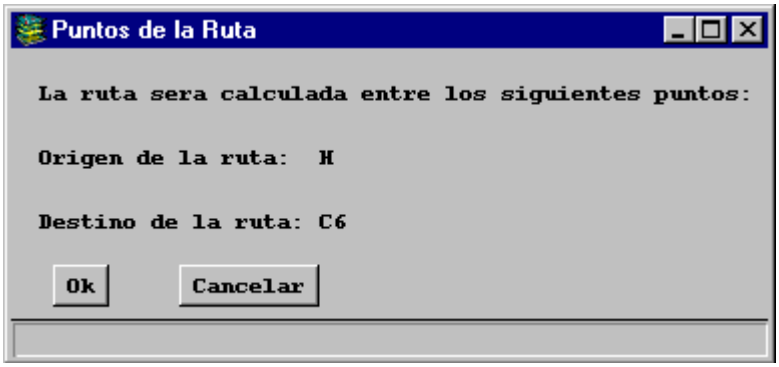


Figure 15: Selected route confirming window.

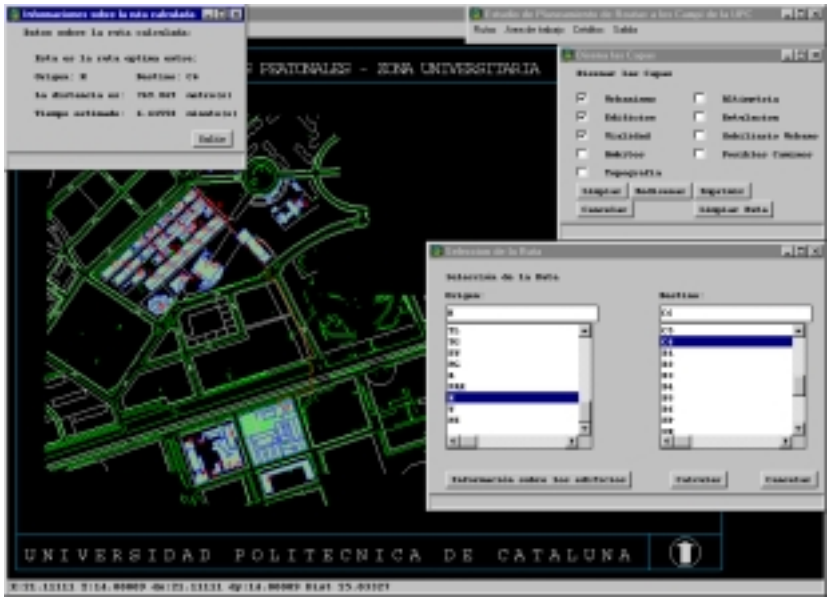


Figure 16: Result displaying window.

In the second option of the main menu (see Figure 17), named *Area de trabajo* (Workspace), we have five sub-options, which are composed of (1) draw-coverage tool, (2) Re-draw, (3) Clear, (4) Zoom of area (*Zoom de Area*), and (5) Normal Area (*Area Normal*) options. The last two options are used to zoom in and zoom out in the map.

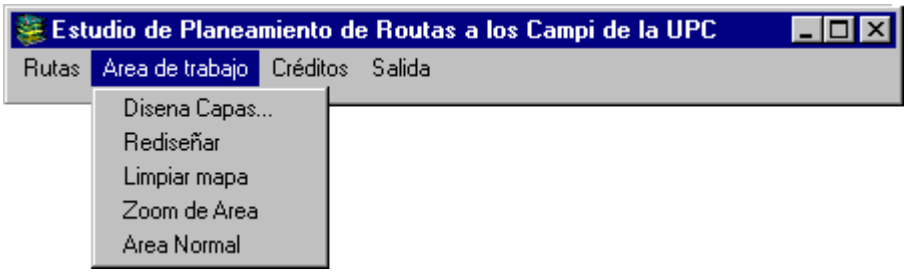


Figure 17: *Area de trabajo* sub-menu of the Router 2.0 application.

3.6 CD Recording

We have consider that the whole work that has been developed must be saved on a safe and portable media. Thus, we have saved it on a CD.

3.6.1 Contents structure

Figure 18 presents the upper level of the CD structure.

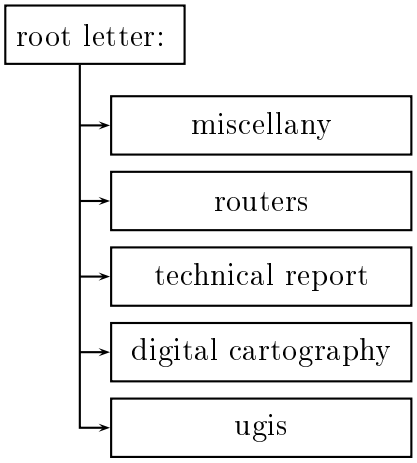


Figure 18: The upper level of the CD structure.

1. In the *Digital Cartography* subdirectory (see Figure 19) we have put in the digital cartography of the university zone of Barcelona in ".DGN" format (for more details see Section 3.2.1 and Appendix 6.3);

2. In the *Miscellany* subdirectory we have put in miscellaneous files which can be used as support material to understand, for instance, how the A* search algorithm works, or to learn some ArcInfo basic lessons and so on;

3. The *Routers* subdirectory is composed of following three subdirectories (see Figure 20):

   - In the *..Router1* subdirectory, there is the first version of the Router application developed by J. Cesar in 1996. The *..router1* subdirectory is composed of (1) spatial data base (coverage: caminets, camins, info, land, northarr, parcellari, parcelles, ulogo, urbanistic and urbans), (2) C application (PFCJCesar), and (3) the AML programs and other necessary files (for more details see Appendix 6.1);
   - The *..Router15* subdirectory is composed of several files in "Borland C++ Builder" format;
   - In the *..Router2* subdirectory, there is the third version of the Router application developed by Hernane Pereira in 2000. The

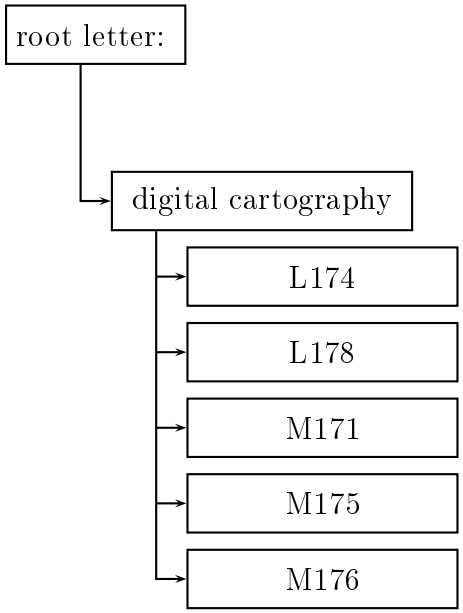


Figure 19: The digital cartography level of the CD structure.

..router2 subdirectory is composed of (1) spatial data base (coverage: altsok, ambsok, camsok, cdssok, edupc, escalaok, info, leyenda, mobsok, retsok, topsok, urbscompl and viascompl), (2) C application (router2 c), (3) the AML programs (aml) and (4) other necessary files (for more details see Appendix 6.2).

4. In the *Technical Report* subdirectory (see Figure 21) we have put in this document in ".PS" format. In addition, this subdirectory is composed of following subdirectories:

   - In the *..Articles* subdirectory we have put in some(qt. exata) papers about route planning subject in ".PDF" format which we have used to proceed our research;
   - In the *..Figures eps* subdirectory we have put in some figures in ".EPS" format presented in this report;

5. In the *ugis* subdirectory we have put in the ArcInfo Tutorial, version 7.2.1 (for more details E.S.R.I. (1992)).

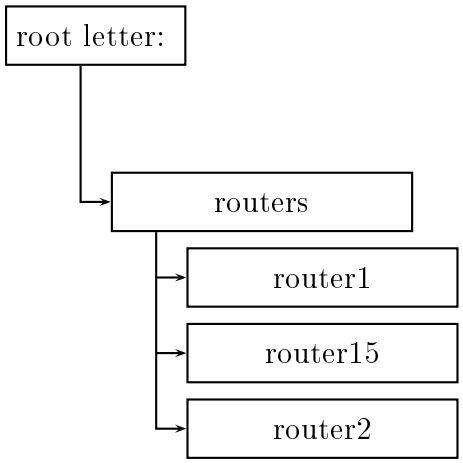


Figure 20: The routers level of the CD structure.

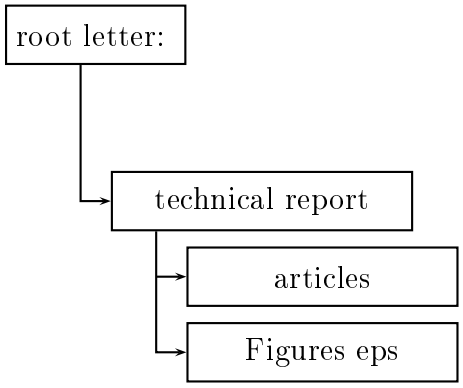


Figure 21: The technical report level of the CD structure.

4 Conclusions

This technical research report has presented a research on route planning, which is composed of six tasks with the aim of contextualizing and to further our research. These tasks are: (1) data recovering, (2) obtention of the updated digital cartography, (3) Router 1.0 application analysis, (4) analysis and development of the Router 2.0 application, (5) the studies on the Dijkstra's, Dantzig's and A* search algorithms, and (6) bibliographic overview. In addition, we have recorded whole work developed on a CD (for more detail, see Section 3.6).

5 Future work

We have perceived the need of implementing other algorithms such as Dantzig's and A* search algorithm in order to examine in detail the their outcomes (e.g. distance criterion, time criterion and memory access criterion) and to compare them to Dijkstra's algorithm outcomes. For this, we will use the C language to proceed the Dantzig's and A* search algorithm implementations.

In addition, we will carry out a research on 3D geographic and spatial analysis focused on route planning. For this, I will propose to develop some additional modules (e.g. impedance control, animation and simulation) to the Router 2.0 application.

6 Appendix

In the present section we attach general information about our research. It is composed of (1) Router 1.0 application sources, (2) Router 2.0 application sources, and (3) digital cartography.

6.1 Router 1.0 application sources

Table 2: Router1 application source names.

AML files	MENU files	DAT, TXT, KEY AND TMP files	C files
ROUTER.AML	ROUTER.MENU	CAMINS.DAT	ROUTER.C
CREAGRAF.AML	EXTREMS.MENU	GRAF.DAT	ARCINFO.C
DIBUIXA.AML		AUTOR.TXT	CAMINS.C
EXTREMS.AML		EDIFICIS.TXT	CJTVERTX.C
CALCUL.AML		NOVERTEXS.TXT	CPARS.C
OPTIM.AML		IGUALS.TXT	ERROR.C
CORRESPON.AML		NOFILE.TXT	FAUXS.C
RESULTAT.AML		NOREADFILE.TXT	GRAFS.C
MOSTRA.AML		URBANISTIC.KEY	ARCINFO.H
		PARCELLARI.KEY	CAMINS.H
		VERTEXS.TMP	CJTVERTX.H
		RESULTAT.TMP	CPARS.H
			ERROR.H
			FAUXS.H
			GRAFS.H
			GENERAL.H

6.1.1 ArcInfo Macro Language - AML

```
/* ROUTER.AML ***********************************************************
&args device
&if [null %device%] &then &return Sintaxi: &RUN ROUTER <station_file>
&else &do
  &messages &off
  &run creagraf.aml
  ARCPLOT
  DISPLAY %device% SIZE 900 700
  &station %device%
  PAGESIZE 36 24.5
  &setvar .inici 'zz'
  &setvar .desti 'zz'
  &run dibuixa.aml
  &menu router.menu &pulldown &size &frame 350 100 ~
                    &stripe 'Calcul del cami optim'
/*  &setvar nofares [delete graf.dat]
  &messages &on
&end /* end do
&return

/* CREAGRAF.AML *********************************************************
&setvar nofares [delete graf.dat]
```

```
TABLES
SELECT CAMINS.AAT
UNLOAD graf.dat FNODE# TNODE# LENGTH CAMINS-ID
QUIT
&setvar file [open graf.dat openstatus -read]
&setvar nofares [close %file%]

/* DIBUIXA.AML ********************************************************
&messages &off
CLEAR
UNITS PAGE
LINESYMBOL 1
BOX .5 .5 35.25 24.5
LINE 19 24.5 19 .5
LINE 19 5 35.25 5
LINE 26 .5 26 5
LINE 19 10.75 35.25 10.75
TEXTSIZE .4
TEXTFONT 'UNIVERS MEDIUM'
TEXTCOLOR WHITE
MOVE 20 .75
TEXT 'Escala 1:1000'
TEXTFONT 'TRIUMVIRATE CONDENSED'
MOVE 19.25 10
TEXTFILE AUTOR.TXT
TEXTFONT 'TIMES BOLD'
TEXTSIZE 1
TEXTFIT 'ZONA UNIVERSITARIA' 20 23 35 23
TEXTFONT 'CGTIMES'
TEXTSIZE .8
TEXTFIT 'CAMINS PEATONALS' 21 22 34 22
MAPEXTENT URBANISTIC
MAPLIMITS .5 .5 19 24
MAPPOSITION CEN CEN
SHADESET COLORNAMES
ASELECT URBANISTIC POLY
RESELECT URBANISTIC POLY t_carrer = 'ca'
POLYGONSHADES URBANISTIC 29 /* Gris fosc
ASELECT URBANISTIC POLY
RESELECT URBANISTIC POLY t_carrer = 'pv'
POLYGONSHADES URBANISTIC 26 /* Blanc
ASELECT URBANISTIC POLY
RESELECT URBANISTIC POLY t_carrer = 'vo'
POLYGONSHADES URBANISTIC 33 /* Gris clar
ASELECT URBANISTIC POLY
RESELECT URBANISTIC POLY t_carrer = 'al'
POLYGONSHADES URBANISTIC 68 /* Verd primavera
ASELECT URBANISTIC POLY
RESELECT URBANISTIC POLY t_carrer = 'cn'
POLYGONSHADES URBANISTIC 83 /* Groc
ASELECT URBANISTIC POLY
RESELECT URBANISTIC POLY t_carrer = 'mo'
POLYGONSHADES URBANISTIC 27 /* Negre
POLYGONSHADES PARCELLARI 53 /* Turquesa fosc
TEXTSIZE .4
TEXTFONT 'UNIVERS MEDIUM'
TEXTCOLOR BLACK
LABELTEXT PARCELLARI ID_EDIFICI
ASELECT PARCELLARI POLY
TEXTCOLOR MAGENTA
RESELECT PARCELLARI POLY ID_EDIFICI = [quote %.inici%]
LABELTEXT PARCELLARI ID_EDIFICI
```

```
ASELECT PARCELLARI POLY
RESELECT PARCELLARI POLY ID_EDIFICI = [quote %.desti%]
LABELTEXT PARCELLARI ID_EDIFICI
ASELECT PARCELLARI POLY
LINESYMBOL 6
LINECOLOR RED
RESELECT CAMINS ARCS pertany = 1
ARCS CAMINS
LINESYMBOL 1
MAPEXT NORTHARR
MAPLIMITS 19.25 1.25 25.75 5
MAPPOSITION CEN CEN
ARCS NORTHARR
LINESYMBOL 1
LINECOLOR WHITE
TEXTCOLOR WHITE
KEYAREA 19.75 21 35 11
KEYSEPARATION .75 .75
KEYBOX .6 .6
KEYSHADE URBANISTIC.KEY
KEYAREA 19.75 12.85 35 11
KEYSEPARATION .75 .75
KEYBOX .6 .6
KEYSHADE PARCELLARI.KEY
&messages &on
&return

/* EXTREMS.AML ********************************************************
&dalines 25
&setvar .inici [getchoice A1 A2 A3 A4 A5 A6 B1 B2 B3 B4 B5 B6 ~
                          C1 C2 C3 C4 C5 C6 D1 D2 D3 D4 D5 D6 ~
                          FP TG TL BI NX CS AR A H P R U ~
                          -prompt 'Entra inici del cami:    ']
&dalines 25
&setvar .desti [getchoice A1 A2 A3 A4 A5 A6 B1 B2 B3 B4 B5 B6 ~
                          C1 C2 C3 C4 C5 C6 D1 D2 D3 D4 D5 D6 ~
                          FP TG TL BI NX CS AR A H P R U ~
                          -prompt 'Entra destinacio:        ']
&if %.inici% eq 'zz' or %.desti% eq 'zz' &then &popup novertexs.txt
&else &do /* do 1
  &if %.inici% eq %.desti% &then &popup iguals.txt
  &else &menu extrems.menu
&end /* end do 1
&return

/* CALCUL.AML *********************************************************
&if %.inici% eq 'zz' or %.desti% eq 'zz' &then &popup novertexs.txt
&else &do /* do 1
  &if %.inici% eq %.desti% &then &popup iguals.txt
  &else &do /* do 2
    &run correspon.aml
    &run optim.aml
  &end /* end do 2
&end /* end do 1
&return

/* CORRESPON.AML ******************************************************
&if %.inici% eq 'A1' &then &setvar .root 243
&if %.inici% eq 'A2' &then &setvar .root 245
&if %.inici% eq 'A3' &then &setvar .root 247
&if %.inici% eq 'A4' &then &setvar .root 249
&if %.inici% eq 'A5' &then &setvar .root 251
```

```
&if %.inici% eq 'A6' &then &setvar .root 253
&if %.inici% eq 'B1' &then &setvar .root 234
&if %.inici% eq 'B2' &then &setvar .root 207
&if %.inici% eq 'B3' &then &setvar .root 211
&if %.inici% eq 'B4' &then &setvar .root 213
&if %.inici% eq 'B5' &then &setvar .root 215
&if %.inici% eq 'B6' &then &setvar .root 226
&if %.inici% eq 'C1' &then &setvar .root 368
&if %.inici% eq 'C2' &then &setvar .root 369
&if %.inici% eq 'C3' &then &setvar .root 232
&if %.inici% eq 'C4' &then &setvar .root 230
&if %.inici% eq 'C5' &then &setvar .root 228
&if %.inici% eq 'C6' &then &setvar .root 282
&if %.inici% eq 'D1' &then &setvar .root 203
&if %.inici% eq 'D2' &then &setvar .root 201
&if %.inici% eq 'D3' &then &setvar .root 199
&if %.inici% eq 'D4' &then &setvar .root 197
&if %.inici% eq 'D5' &then &setvar .root 195
&if %.inici% eq 'D6' &then &setvar .root 193
&if %.inici% eq 'FP' &then &setvar .root 187
&if %.inici% eq 'TL' &then &setvar .root 191
&if %.inici% eq 'BI' &then &setvar .root 364
&if %.inici% eq 'NX' &then &setvar .root 239
&if %.inici% eq 'TG' &then &setvar .root 189
&if %.inici% eq 'CS' &then &setvar .root 358
&if %.inici% eq 'AR' &then &setvar .root 414
&if %.inici% eq 'A' &then &setvar .root 330
&if %.inici% eq 'R' &then &setvar .root 287
&if %.inici% eq 'P' &then &setvar .root 284
&if %.inici% eq 'H' &then &setvar .root 341
&if %.inici% eq 'U' &then &setvar .root 336
&if %.desti% eq 'A1' &then &setvar .target 243
&if %.desti% eq 'A2' &then &setvar .target 245
&if %.desti% eq 'A3' &then &setvar .target 247
&if %.desti% eq 'A4' &then &setvar .target 249
&if %.desti% eq 'A5' &then &setvar .target 251
&if %.desti% eq 'A6' &then &setvar .target 253
&if %.desti% eq 'B1' &then &setvar .target 234
&if %.desti% eq 'B2' &then &setvar .target 207
&if %.desti% eq 'B3' &then &setvar .target 211
&if %.desti% eq 'B4' &then &setvar .target 213
&if %.desti% eq 'B5' &then &setvar .target 215
&if %.desti% eq 'B6' &then &setvar .target 226
&if %.desti% eq 'C1' &then &setvar .target 368
&if %.desti% eq 'C2' &then &setvar .target 369
&if %.desti% eq 'C3' &then &setvar .target 232
&if %.desti% eq 'C4' &then &setvar .target 230
&if %.desti% eq 'C5' &then &setvar .target 228
&if %.desti% eq 'C6' &then &setvar .target 282
&if %.desti% eq 'D1' &then &setvar .target 203
&if %.desti% eq 'D2' &then &setvar .target 201
&if %.desti% eq 'D3' &then &setvar .target 199
&if %.desti% eq 'D4' &then &setvar .target 197
&if %.desti% eq 'D5' &then &setvar .target 195
&if %.desti% eq 'D6' &then &setvar .target 193
&if %.desti% eq 'FP' &then &setvar .target 187
&if %.desti% eq 'TG' &then &setvar .target 189
&if %.desti% eq 'TL' &then &setvar .target 191
&if %.desti% eq 'BI' &then &setvar .target 364
&if %.desti% eq 'NX' &then &setvar .target 239
&if %.desti% eq 'CS' &then &setvar .target 358
&if %.desti% eq 'AR' &then &setvar .target 414
```

```
&if %.desti% eq 'A' &then &setvar .target 330
&if %.desti% eq 'R' &then &setvar .target 287
&if %.desti% eq 'P' &then &setvar .target 284
&if %.desti% eq 'H' &then &setvar .target 341
&if %.desti% eq 'U' &then &setvar .target 336

/* OPTIM.AML ***************************************************
&setvar nofares [delete vertexs.tmp]
&setvar file = [open vertexs.tmp openstatus -write]
&if %openstatus% ne 0 &then &run nofile.txt
&else &do /* 1
  &if [write %file% %.root%] <> 0 &then &run nofile.txt
  &else &do /* do 2
    &if [write %file% %.target%] <> 0 &then &run nofile.txt
    &else &do /* do 3
      &if [close %file%] <> 0 &then &run nofile.txt
      &else &do /* do 4
        &system router
        &setvar nofares [delete vertexs.tmp]
        &run resultat.aml
      &end /* end do 4
    &end /* end do 3
  &end /* end do 2
&end /* end do 1

/* RESULTAT.AML ************************************************
&setvar file [open resultat.tmp openstatus -read]
&if %openstatus% <> 0 &then &run noreadfile.txt
&else &do /* do 1
  &setvar .longitud [read %file% readstatus]
  &if %readstatus% <> 0 &then &run noreadfile.txt
  &else &do /* do 2
    &setvar aresta [read %file% readstatus]
    &if %readstatus% <> 0 &then &run noreadfile.txt
    &else &do /* do 3
      &messages &off
      &do &while %readstatus% = 0
        aselect camins arcs
        reselect camins arcs camins-id = %aresta%
        calculate camins arcs pertany = 1
        aselect camins arcs
        &setvar aresta [read %file% readstatus]
      &end /* end do-while
      &if [close %file%] ne 0 &then &setvar nofares [delete %file%]
      &setvar .temps = %.longitud% / 2
      &setvar .temps = %.temps% / 60
      &run mostra.aml
      aselect camins arcs
      calculate camins arcs pertany = 0
      &setvar .inici 'zz'
      &setvar .desti 'zz'
      &run dibuixa.aml
      &messages &on
    &end /* end do 3
  &end /* end do 2
&end /* end do 1

/* MOSTRA.AML **************************************************
&run dibuixa.aml
&messages &popup
&type
&type Aquest es el cami optim entre %.inici% i %.desti%
```

```
&type
&type        La seva longitud es %.longitud% metres
&type
&type        El temps estimat en recorre'l es de %.temps% minuts
&type
&type   <Prem Quit quan desitgis acabar la visualitzacio>
&type
&messages &off
&return
```

6.1.2 MENU files

```
/* ROUTER.MENU *****************************************************
1 Sample router
'Camins          '
  'Nou cami             '  &run extrems.aml
  '                     '
  'Informacio edificis '  &popup edificis.txt
'Area de treball '        &run dibuixa.aml
'Netejar         '        CLEAR
'Sortir          '        QUIT

/* EXTREMS.MENU ****************************************************
7 Menu per mostrar o no el cami

El cami es calculara entre els seguents nodes:

  Inici del cami:%program1

  Final del cami:%program2

   %acceptar %cancelar

%program1 display .inici 2 value
%program2 display .desti 2 value
%acceptar button Acceptar &run calcul %.inici% %.desti%; &return
%cancelar button Cancel.lar &setvar .inici 'zz'; &setvar .desti 'zz'; &return
```

6.1.3 Necessary files - DAT, TXT, KEY Y TMP

```
/* GRAF.DAT ********************************************************
295,187,24.61913,1
2,295,54.09769,2
395,2,59.42771,3
396,395,21.33399,4
400,388,88.34045,5
295,3,67.74267,6
4,3,109.22703,7
.
.
.
328,287,19.65264,282
172,328,73.35757,283
177,415,69.34691,288
415,414,11.94517,287
```

```
415,333,67.00604,284
328,333,42.40464,285
333,335,150.18319,286

/* AUTOR.TXT ******************************************************
PROJECTE D'INFORMATICA DE GESTIO

Alumne:
  Julio Cesar Rodriguez Leon de Santos

Director/Ponent:
  Lluis Perez Vidal

Departament:
  L.S.I. - Seccio Grafics

Data de presentacio: 4 de juliol de 1996

/* EDIFICIS.TXT ***************************************************
Clau Descripcio

A1   Aulari 1
A2   Aulari 2
A3   Aulari 3
A4   Aulari 4
A5   Aulari 5
A6   Aulari 6
B0   Edifici departamental
B1   Edifici departamental
B2   Biblioteca de Camins
B3   CUP
B4   CUP
B5   Casa de l'Estudiant
B6   Facultat d'Informatica de Barcelona
C1   Edifici departamental
C2   Edifici departamental
C3   Edifici departamental
C4   Edifici departamental
C5   Edifici departamental
C6   En construccio
D1   Edifici departamental
D2   Edifici departamental
D3   Edifici departamental
D4   Edifici departamental
D5   Edifici departamental
D6   Edifici departamental
FP   Fundacio Politecnica de Catalunya
NX   Edifici Nexus
BI   Biblioteca de la UPC
TL   Edifici Telecos
TG   Torre Girona
P    Escola Universitaria Politecnica de Barcelona
R    Pavello de Govern
A    E.T.S. d'Arquitectura de Barcelona
AR   Aularis
H    E.T.S. d'Enginyers Industrials de Barcelona
U    Facultat de Matematiques i Estadistica
CS   Centre de Supercomputacio de Catalunya

/* NOVERTEXS.TXT **************************************************
Informacio d'error:
```

```
Atencio: Els vertexs entrats no son valids.
Si us plau, torni a provar-ho.

/* IGUALS.TXT ***************************************************
Informacio d'error:

Atencio: els vertexs inici i desti son el mateix.
Si us plau, torni a provar.ho.

/* NOFILE.TXT ***************************************************
&messages &popup
&type Informacio d'error:
&type
&type Atencio: Error d'acces al fitxer vertexs.tmp.
&type
&messages &on

/* NOREADFILE.TXT ***********************************************
&messages &popup
&type Informacio d'error:
&type
&type Atencio: Error d'acces al fitxer de dades
&type
&messages &on

/* URBANISTIC.KEY ***********************************************
.29
Carrer
.26
Pas de Vianants
.33
Vorera
.27
Mobiliari
.68
Illes d'edificis
.83
Campus Nord

/* PARCELLARI.KEY ***********************************************
.53
Edificis

/* VERTEXS.TMP **************************************************
203
336

/* RESULTAT.TMP *************************************************
739.478271484
17
65
80
149
161
168
177
223
229
236
241
242
245
```

247

6.1.4 C Language

```
// ROUTER.C ******************************************************
#include "general.h"
#include "arcinfo.h"
#include "camins.h"
#include "cjtvertx.h"
#include "cpars.h"
#include "errors.h"
#include "grafs.h"

void Dijkstra(LVERTEXS cami, GRAF g, CPAR *minh, CJTVERTEXS cjt,
                                    VERTEX inici, VERTEX desti);

/*********************************************************************/
/* Funcio que calcula el cami optim entre dos punts g'un graf. Algorisme */
/* de Dijkstra, implementat amb:                                     */
/*      Cua amb Prioritat amb Acces Rapid (CPAR): utilitzada per fer la  */
/*           seleccio del vertex de valor minim. Es fa servir un minheap */
/*           per fer el bucle d'aquesta seleccio mes eficient, mes un    */
/*           vector auxiliar que apunta a la posicio en la qual es troba */
/*           el vertex que cerquem, necessari per no malmetre el cost    */
/*           logaritmic obtingut.                                        */
/*     Conjunt de Vertexs (CJTVERTEXS): vector utilitzat per guardar els */
/*           vertexs del graf que encara no s'han visitat.               */
/*     Graf (GRAF): es un vector amb tots els vertexs de la Base de      */
/*           Dades de la qual hem agafat la informacio, i la seva llista */
/*           de vertexs adjacents a cadascun.                            */
/*     Llista d'arestes (LLISTAARESTES): utilitzada per guardar la       */
/*           informacio que sera retornada a l'aplicacio grafica.        */
/*           Aquesta informacio consta del pes (longitud) total del cami */
/*           optim trobat i la llista de les arestes que el composen.    */
/*********************************************************************/

void Dijkstra(LVERTEXS cami, GRAF g, CPAR *minh, CJTVERTEXS cjt,
                                    VERTEX inici, VERTEX desti)
{
  NODEG *p, *anterior;
  NODEA a;
  PES etiqueta;
  LLISTAV *lv1, *lv2;
  int s;
  ARESTA edge;

  EliminarNodeMinheap(minh, inici);
  cjt[inici]=-1;
  p=g[inici];
  while (p!=NULL) {
        ModificarPesMinheap(minh, p->successor, p->longitud, inici);
    p=p->seg;
  }
  CrearNodeA(inici, 0, &a);
  AfegirLongitudCami(cami, a);
  do {
    a=MinimMinheap(*minh);
    p=g[a.node];
    AfegirLongitudCami(cami, a);
```

```
    lv1=(LLISTAV *)malloc(sizeof(LLISTAV));
        anterior=g[a.node];
        while (anterior->successor!=a.darrer) anterior=anterior->seg;
        edge=anterior->id_aresta;
    lv1->aresta=anterior->id_aresta;
    lv1->seg=cami[a.darrer].lvertexs;
    cami[a.node].lvertexs=lv1;
    cjt[a.node]=-1;
    EsborrarMinimMinheap(minh);
    while (p!=NULL) { s=p->successor;
          if (cjt[p->successor]>0) {
        etiqueta=a.longitud+p->longitud;
        if (etiqueta < ConsultarPesMinheap(*minh, p->successor)) {
          ModificarPesMinheap(minh, p->successor, etiqueta, a.node);
          lv2=(LLISTAV *)malloc(sizeof(LLISTAV));
                  edge=p->id_aresta;
          lv2->aresta=p->id_aresta;
          lv2->seg=cami[a.node].lvertexs;
          cami[p->successor].lvertexs=lv2;
        }
      }
      p=p->seg;
        }
  } while (a.node!=desti);
}

/*************************************************************************/
/* Funcio principal.                                                     */
/*************************************************************************/
main()
{
  NODEG *g[1000];
  NODEG *n1, *n2;
  CJTVERTEXS cv;
  CPAR mh;
  VERTEX node_inici, node_fi, inici, desti, v;
  ARESTA id_aresta;
  PES longitud;
  LVERTEXS cami;
  LLISTAV *lv, *en_curs;
  FILE *fp, *pf;
  int i, dec, sig;
  static char *ch1;
  char ch2[20];

  CrearGraf(g);
  CrearCPAR(&mh);
  CrearCjtVertexs(cv);
  CrearCami(cami);
  fp=fopen("graf.dat", "r");
  if (!fp) error("No es pot obrir el fitxer de dades.");
  else {
        ObtenirDades(fp, &node_inici, &node_fi, &longitud, &id_aresta);
        while (!feof(fp)) {
          n1=(NODEG *)malloc(sizeof(NODEG));
      n1->successor=node_fi;
      n1->id_aresta=id_aresta;
      n1->longitud=longitud;
      AfegirAdjacenciaGraf(&g[node_inici], n1);
      n2=(NODEG *)malloc(sizeof(NODEG));
      n2->successor=node_inici;
      n2->id_aresta=id_aresta;
```

```
      n2->longitud=longitud;
      AfegirAdjacenciaGraf(&g[node_fi], n2);
          ObtenirDades(fp, &node_inici, &node_fi, &longitud, &id_aresta);
    }
        fclose(fp);
    v=PrimerNodeGraf(g);
    while (!DarrerNodeGraf(g, v)) {
      InsertarPesMinheap(&mh, v, MAXINT, -1);
      AfegirVertex(cv, v);
      v=SeguentNodeGraf(g, v);
    }
        AgafarPeticio(&inici, &desti);
        Dijkstra(cami, g, &mh, cv, inici, desti);
        EliminarGraf(g);
    pf=fopen("resultat.tmp", "w");
    if (!pf) error("No es pot crear el fitxer resultat.");
    else {
      ch1=ecvt(cami[desti].longitud, 12, &dec, &sig);
      i=0;
      while (i<dec) {
        ch2[i]=ch1[i];
        i++;
      }
      ch2[i]='.';
      for(;i<strlen(ch1);i++) ch2[i+1]=ch1[i];
      ch2[i+1]=0;
      fputs(ch2, pf);
      en_curs=cami[desti].lvertexs;
      while (en_curs!=NULL) {
        fputc('\n', pf);
                ch1=EnterAString(en_curs->aresta);
        fputs(ch1, pf);
        en_curs=en_curs->seg;
      }
          fputc('\n', pf);
      fclose(pf);
    }
  }
}

// ARCINFO.C ******************************************************
#include "arcinfo.h"
#include <ctype.h>

/*********************************************************************/
/* Funcions que fan la interaccio amb l'aplicacio ARC/INFO.          */
/*********************************************************************/

void ObtenirDades(FILE *fp, VERTEX *vi, VERTEX *vf, PES *lg, ARESTA *a)
{
  char c;
  float f=0.0;
  int div=10;

if (((c=getc(fp))!='\n') && (!feof(fp))) {
/* Agafem el vertex inicial. */
  *vi=a_enter(c);
  while ((c=getc(fp)) != ',') {
    *vi=*vi*10;
    *vi=*vi+a_enter(c);
  }
/* Agafem el vertex final. */
```

```
  *vf=a_enter(getc(fp));
  while ((c=getc(fp)) != ',') {
    *vf=*vf*10;
    *vf=*vf+a_enter(c);
  }

/* Agafem la longitud. */
  *lg=a_enter(getc(fp));
  while ((c=getc(fp)) != '.') {
    *lg=*lg*10;
    *lg=*lg+a_enter(c);
  }
  while (isdigit(c=getc(fp))) {
    f=a_enter(c)%10;
    f=f/div;
    div*=10;
    *lg=*lg+f;
  }

/* Agafem l'aresta. */
  *a=a_enter(getc(fp));
  while ((c=getc(fp))!='\n') {
    *a=*a*10;
    *a=*a+a_enter(c);
  }
 }
}

void AgafarPeticio(VERTEX *vi, VERTEX *vf)
{
  FILE *fp;
  char c;

  fp=fopen("vertexs.tmp", "r");
  if (!fp) error("No es pot obrir el fitxer temporal.");
  else {
    c=getc(fp);
    do {
      *vi=*vi + a_enter(c);
      *vi=*vi * 10;
    } while ((c=getc(fp))!='\n');
    *vi = *vi / 10;
    c=getc(fp);
    do {
      *vf=*vf + a_enter(c);
      *vf=*vf * 10;
    } while ((c=getc(fp))!='\n');
    *vf = *vf / 10;
    fclose(fp);
  }
}

void EmmagatzemarResultat(NODEC optim)
{
  LLISTAV *en_curs;
  FILE *fp;
  int i, dec, sig;
  static char *ch1;
  char ch2[20];

  fp=fopen("resultat.tmp", "w");
  if (!fp) error("No es pot crear el fitxer resultat.");
```

```
  else {
    ch1=ecvt(optim.longitud, 12, &dec, &sig);
    i=0;
    while (i<dec) {
      ch2[i]=ch1[i];
      i++;
    }
    ch2[i]='.';
    for(;i<strlen(ch1);i++) ch2[i+1]=ch1[i];
    ch2[i+1]=0;
    fputs(ch2, fp);
    en_curs=optim.lvertexs;
    while (en_curs!=NULL) {
          fputc(RETORN, fp);
    /*  itoa(en_curs->node, ch1, 10); */
      fputs(ch1, fp);
      en_curs=en_curs->seg;
    }
        fclose(fp);
  }
}

// CAMINS.C ************************************************
#include "camins.h"
#include "errors.h"

/**************************************************************************/
/* Funcions del tipus de dades CAMINS.                                    */
/**************************************************************************/

void CrearCami(LVERTEXS cami)
{
  int i;

  for (i=0;i<1000;i++) {
    cami[i].lvertexs=NULL;
  }
}

void AfegirLongitudCami(LVERTEXS cami, NODEA a)
{
  cami[a.node].longitud=a.longitud;
}

void EliminarCamiVertex(LLISTAV **lv)
{
  LLISTAV *en_curs=*lv;
  LLISTAV *anterior;

  while (en_curs!=NULL) {
    anterior=en_curs;
    en_curs=en_curs->seg;
    free(anterior);
  }
  *lv=NULL;
}

void EsborrarCamins(LVERTEXS cami)
{
  int i;

  for(i=0; i<1000; i++) EliminarCamiVertex(&cami[i].lvertexs);
```

```
}

// CJTVERTX.C ******************************************************
#include "cjtvertx.h"

/*************************************************************************/
/* Funcions del tipus de dades CJTVERTEXS.                               */
/*************************************************************************/

void CrearCjtVertexs(CJTVERTEXS cjt)
{
  int i;

  for (i=0; i<1000; i++) cjt[i]=-1;
}

void AfegirVertex(CJTVERTEXS cjt, VERTEX v)
{
  cjt[v]=v;
}

// CPARS.C *********************************************************
#include "cpars.h"

/*************************************************************************/
/* Funcions del tipus de dades CPAR.                                     */
/*************************************************************************/

/*************************************************************************/
/* Quart nivell del disseny descendent.                                  */
/*************************************************************************/

void CopiarRegistreMinheap(NODEA a, NODEA *b)
{
  b->node=a.node;
  b->longitud=a.longitud;
  b->darrer=a.darrer;
}

/*************************************************************************/
/* Tercer nivell del disseny descendent.                                 */
/*************************************************************************/

int CercarPareMinheap(int index)
{
  if (index<3) return(0);
  else {
    index++;
    if (senar(index)) return (((index-1)/2)-1);
    else return ((index/2)-1);
  }
}

void CrearRegistreMinheap(CPAR *minh, VERTEX v, PES lg, VERTEX darrer)
{
  minh->arbre[minh->SL].node=v;
  minh->arbre[minh->SL].longitud=lg;
  minh->arbre[minh->SL].darrer=darrer;
  minh->acces[v]=minh->SL;
  ++(minh->SL);
}
```

```
void IntercanviarRegistreMinheap(CPAR *minh, int *index, int substituit)
{
  NODEA aux;

  CopiarRegistreMinheap(minh->arbre[*index], &aux);
  CopiarRegistreMinheap(minh->arbre[substituit], &minh->arbre[*index]);
  minh->acces[minh->arbre[*index].node]=*index;
  CopiarRegistreMinheap(aux, &minh->arbre[substituit]);
  minh->acces[minh->arbre[substituit].node]=substituit;
  *index=substituit;
}

int FillMesPetitMinheap(CPAR minh, int pare)
{
  int fill, rightsoon, leftsoon;
  PES pes_pare, pes_fill_dret, pes_fill_esq, pes_fill;

  pes_pare=minh.arbre[pare].longitud;
  if (pare==0) rightsoon=1;
  else rightsoon=(2*pare)+1;
  leftsoon=rightsoon+1;
  pes_fill_dret=minh.arbre[rightsoon].longitud;
  pes_fill_esq=minh.arbre[leftsoon].longitud;
  if (minh.SL <= rightsoon) return (0);
  if (minh.SL-1 == rightsoon) return ((pes_pare<pes_fill_dret)?0:rightsoon);
  if (pes_fill_dret<0) return (0);
  else {
    if (pes_fill_esq<0) return ((pes_pare<pes_fill_dret)?0:rightsoon);
    else {
      pes_fill=min(pes_fill_dret, pes_fill_esq);
      fill=(pes_fill_dret < pes_fill_esq)?rightsoon:leftsoon;
      if (pes_fill < pes_pare) return (fill);
      else return (0);
    }
  }
}

/*************************************************************************/
/* Segon nivell del disseny descendent.                                  */
/*************************************************************************/

void InsertarPesMinheap(CPAR *minh, VERTEX v, PES lg, VERTEX darrer)
{
  int index, pare;

  CrearRegistreMinheap(minh, v, lg, darrer);
  index=minh->SL-1;
  pare=CercarPareMinheap(index);
  while(minh->arbre[index].longitud < minh->arbre[pare].longitud) {
    IntercanviarRegistreMinheap(minh, &index, pare);
    pare=CercarPareMinheap(index);
  }
}

void ModificarPesMinheap(CPAR *minh, VERTEX v, PES lg, VERTEX darrer)
{
  int index1, index2, pare, fill;
  PES longitud;
  NODEA aux;

  index1=minh->acces[v];
  index2=minh->SL-1;
```

```
  CopiarRegistreMinheap(minh->arbre[index1], &aux);
  CopiarRegistreMinheap(minh->arbre[index2], &minh->arbre[index1]);
  minh->acces[minh->arbre[index1].node]=index1;
  --(minh->SL);
  pare=CercarPareMinheap(index1);
  while(minh->arbre[index1].longitud < minh->arbre[pare].longitud) {
    IntercanviarRegistreMinheap(minh, &index1, pare);
    pare=CercarPareMinheap(index1);
  }
  while ((fill=FillMesPetitMinheap(*minh, index1))!=0) {
    IntercanviarRegistreMinheap(minh, &index1, fill);
  }
  InsertarPesMinheap(minh, aux.node, lg, darrer);
}

void EliminarNodeMinheap(CPAR *minh, VERTEX v)
{
  int index1, index2, pare;

  index1=minh->acces[v];
  index2=minh->SL-1;
  CopiarRegistreMinheap(minh->arbre[index2], &minh->arbre[index1]);
  minh->acces[minh->arbre[index1].node]=index1;
  --(minh->SL);
  pare=CercarPareMinheap(index1);
  while(minh->arbre[index1].longitud < minh->arbre[pare].longitud) {
    IntercanviarRegistreMinheap(minh, &index1, pare);
    pare=CercarPareMinheap(index1);
  }
}

void EsborrarMinimMinheap(CPAR *minh)
{
  int index, fill;

  index=minh->SL-1;
  CopiarRegistreMinheap(minh->arbre[index], &minh->arbre[0]);
  --(minh->SL);
  index=0;
  while ((fill=FillMesPetitMinheap(*minh, index))!=0) {
    IntercanviarRegistreMinheap(minh, &index, fill);
  }
}

/*********************************************************************/
/* Nivell zero del disseny descendent.                               */
/*********************************************************************/

void CrearCPAR(CPAR *minh)
{
  int i;

  minh->SL=0;
  for (i=0; i<1000; i++) minh->acces[i]=-1;
}

void CrearNodeA(VERTEX v, PES lg, NODEA *a)
{
  a->node=v;
  a->longitud=lg;
}
```

```
NODEA MinimMinheap(CPAR minh)
{
  return (minh.arbre[0]);
}

PES ConsultarPesMinheap(CPAR minh, VERTEX v)
{
  return (minh.arbre[minh.acces[v]].longitud);
}

// ERROR.C ************************************************
#include "errors.h"

/*********************************************************************/
/* Tractaments dels errors i les excepcions.                         */
/*********************************************************************/

void error(char *tipus)
{
  printf("\nERROR :: %s",tipus);
  exit(1);
}

// FAUXS.C ************************************************
#include "fauxs.h"

char *EnterAString(int i)
{
  char ch[20], res[20];
  int j=0, index;

  do {
    ch[j++]=a_caracter(i%10);
  } while ((i=i/10)!=0);
  ch[j]=0;
  for(index=0; index<strlen(ch); index++, j--) res[index]=ch[j-1];
  res[index]=0;
  return (res);
}

// GRAFS.C ************************************************
#include "grafs.h"
#include "errors.h"

/*********************************************************************/
/* Funcions del tipus de dades GRAF.                                 */
/*********************************************************************/

void CrearGraf(NODEG *g[])
{
  int i;

  for(i=0; i<1000; i++) {
    g[i]=NULL;
  }
}

void EsborrarAdjacents(GRAF g, VERTEX i)
{
  NODEG *en_curs;
  NODEG *anterior;
```

```
  en_curs=g[i];
  while (en_curs!=NULL) {
    anterior=en_curs;
    en_curs=en_curs->seg;
    free(anterior);
  }
  g[i]=NULL;
}

void EliminarGraf(GRAF g)
{
  int i=0;

  while (g[i]!=NULL) {
    EsborrarAdjacents(&g[i], i);
    i++;
  }
}

VERTEX DarrerNodeGraf(GRAF g, VERTEX v)
{
/*  return (g[v]==NULL); */
  return (v==1000);
}

VERTEX SeguentNodeGraf(GRAF g, VERTEX v)
{
  do {
    v++;
  } while ((g[v]==NULL) && (v<1000));
  return (v);
}

VERTEX PrimerNodeGraf(GRAF g)
{
  VERTEX v=0;

  return (SeguentNodeGraf(g, v));
}

void CrearRegistreGraf(NODEG **registre, NODEG *nou)
{
  *registre=nou;
}

void AfegirAdjacenciaGraf(NODEG **lv, NODEG *nou)
{
  NODEG *punter=*lv;
  NODEG *adjacent;
  NODEG *anterior=NULL;

  while (punter!=NULL && punter->longitud<nou->longitud) {
    anterior=punter;
    punter=punter->seg;
  }
  adjacent=(NODEG *)malloc(sizeof(NODEG));
  CrearRegistreGraf(&adjacent, nou);
  if (adjacent==NULL) error("No hi ha prou memoria");
  adjacent->seg=punter;
  if (anterior==NULL) *lv=adjacent;
  else anterior->seg=adjacent;
}
```

```
void LlistarAdjacencies(GRAF g, VERTEX i)
{
  NODEG *en_curs;

  en_curs=g[i];
  while (en_curs != NULL) {
    printf("\nADJACENCIES DEL VERTEX %d",i);
    printf("\tSuccessor::%d",en_curs->successor);
    printf("\tAresta::%d",en_curs->id_aresta);
    printf("\tLongitud::%f",en_curs->longitud);
    en_curs=en_curs->seg;
  }
}

void LlistarGraf(GRAF g)
{
  NODEG *en_curs;
  int i=1;

  while (g[i]!=NULL) {
    en_curs=g[i];
    while (en_curs) {
      printf("\nADJACENCIES DEL VERTEX %d",i);
      printf("\tSuccessor::%d",en_curs->successor);
      printf("\tAresta::%d",en_curs->id_aresta);
      printf("\tLongitud::%f",en_curs->longitud);
      en_curs=en_curs->seg;
    }
    i++;
  }
}

// ARCINFO.H ******************************************************
#ifndef arcinfo_inclos
#define arcinfo_inclos

#include "general.h"

/*************************************************************************/
/* Declaracions de les funcions que fan la interaccio amb l'aplicacio.   */
/*************************************************************************/

void ObtenirDades(FILE *fp, VERTEX *vi, VERTEX *vf, PES *lg, ARESTA *a);
void AgafarPeticio(VERTEX *vi, VERTEX *vf);
void EmmagatzemarResultat(NODEC optim);
void RetornaCamiOptim(LVERTEXS llista);

#endif

// CAMINS.H *******************************************************
#include "general.h"

/*************************************************************************/
/* Declaracions de les funcions del tipus de dades CAMINS.               */
/*************************************************************************/

void CrearCami(LVERTEXS cami);
void EliminarCamiVertex(LLISTAV **lv);
void AfegirLongitudCami(LVERTEXS cami, NODEA a);
```

```
// CJTVERTX.H ***************************************************
#ifndef cjtvertx_inclos
#define cjtvertex_inclos

#include "general.h"

/*************************************************************************/
/* Declaracio de les funcions del tipus de dades CJTVERTEXS.             */
/*************************************************************************/

void CrearCjtVertexs(CJTVERTEXS cjt);
void AfegirVertex(CJTVERTEXS cjt, VERTEX v);

#endif

// CPARS.H ******************************************************
#ifndef cpars_inclos
#define cpars_inclos

#include "general.h"

/*************************************************************************/
/* Declaracions de les funcions del tipus de dades CPAR.                 */
/*************************************************************************/

void CrearCPAR(CPAR *minh);
void EliminarNodeMinheap(CPAR *minh, VERTEX v);
void EsborrarMinimMinheap(CPAR *minh);
void InsertarPesMinheap(CPAR *minh, VERTEX v, PES longitud, VERTEX darrer);
void ModificarPesMinheap(CPAR *minh, VERTEX v, PES longitud, VERTEX darrer);
void CrearNodeA(VERTEX v, PES longitud, NODEA *a);
NODEA MinimMinheap(CPAR minh);
PES ConsultarPesMinheap(CPAR minh, VERTEX v);
void CopiarRegistreMinheap(NODEA a, NODEA *b);
void CrearRegistreMinheap(CPAR *minh, VERTEX v, PES longitud, VERTEX darrer);
void IntercanviarRegistreMinheap(CPAR *minh, int *index, int subst);
int CercarPareMinheap(int index);
int FillMesPetitMinheap(CPAR minh, int pare);

#endif

// ERROR.H ******************************************************
/*************************************************************************/
/* Declaracions dels tractaments dels errors i les excepcions.           */
/*************************************************************************/

void error(char *tipus);

// FAUXS.H ******************************************************
#include "general.h"

char *EnterAString(int i);

// GRAFS.H ******************************************************
#ifndef grafs_inclos
#define grafs_inclos

#include "general.h"

/*************************************************************************/
/* Declaracions de les funcions del tipus de dades GRAF.                 */
/*************************************************************************/
```

```
void CrearGraf(NODEG *g[]);
void EliminarGraf(GRAF g);
VERTEX PrimerNodeGraf(GRAF g);
VERTEX SeguentNodeGraf(GRAF g, VERTEX v);
VERTEX DarrerNodeGraf(GRAF g, VERTEX v);
void AfegirAdjacenciaGraf(NODEG **lv,NODEG *nou);
void LlistarAdjacencies(GRAF g, VERTEX i);
void LlistarGraf(GRAF g);

#endif

// GENERAL.H ******************************************************
#ifndef general_inclos
#define general_inclos

/*#include <conio.h>*/
#include <stdio.h>
#include <stdlib.h>
#include <values.h>
/*#include <process.h>*/
#include "ctants.h"
#include <in_params.h>
#include "fauxs.h"

/*************************************************************************/
/* Estructura de Dades GRAF: Llista d'adjacencia implementada amb vector.*/
/*      Vector de punters a la llista d'adjacencia de cada node -> GRAF  */
/*      Nodes de la llista d'adjacencia -> NODEG                         */
/*************************************************************************/

NODEG {
  VERTEX successor;
  PES longitud;
  ARESTA id_aresta;
  NODEG *seg;
};

typedef NODEG *GRAF[1000];

/*************************************************************************/
/* Estructura de Dades CPAR: implementacio d'una cua amb prioritat       */
/* d'acces rapid, La cua es un vector i un apuntador a la darrera        */
/* posicio lliure, i l'acces rapid es fa mitjançant un segon vector que  */
/* apunta a la posicio que ocupa el node en questio al vector de minheap.*/
/*      Vector d'apuntadors a nodes del Minheap -> VMINHEAP              */
/*      Vector propi de Minheap -> ARBRE                                 */
/*      Informacio que conte cada node del minheap -> NODEA              */
/*      Apuntador a la darrera posicio ocupada del vector -> SL          */
/*************************************************************************/

typedef struct {
  VERTEX node;
  PES longitud;
  VERTEX darrer;
} NODEA;

typedef struct {
  NODEA arbre[1000];
  int SL;
  int acces[1000];
} CPAR;
```

```
/*************************************************************************/
/* Estructura de Dades Conjunt de Vertexs: implementada amb vector.      */
/*      Vector -> CJTVERTEXS                                             */
/*************************************************************************/

typedef VERTEX CJTVERTEXS[1000];

/*************************************************************************/
/* Estructura de Dades Llista d'arestes: implementada amb vector.        */
/*      Vector -> LLISTAARESTES                                          */
/*      Informacio que conte cada posicio del vector -> NODEC            */
/*************************************************************************/

LLISTAV {
  ARESTA aresta;
  LLISTAV *seg;
};

typedef struct {
  LLISTAV *lvertexs;
  PES longitud;
} NODEC;

typedef NODEC LVERTEXS[1000];

#endif
```

6.2 Router 2.0 application sources

Table 3: Router2 application source names.

AML files	MENU files	DAT, TXT and TMP files	C files
ROUTER2.AML	ROUTER2.MENU	CAMINOS2.DAT	ROUTER2.C
DISENA2.AML	DISENA2.MENU	EDIFICIO2.TXT	FUNCION2.C
SALIR2.AML	EXTREMOS2.MENU	CREDITOS2.TXT	GENERAL.H
MENU2.AML	SELECCION2.MENU	EDCOD2.TXT	
LIMPIAR2.AML	PRESENTA2.MENU	INVALIDOS2.TXT	
EXTREMOS2.AML		IGUALES2.TXT	
CALCULO2.AML		NOVERTEXS2.TXT	
OPTIMO2.AML		NOVERTEXSVALIDOS2.TXT	
RESULTADO2.AML		VERTEXS2.TMP	
DIBUJA2.AML		RESULTADO2.TMP	
DIBUJACAPA2.AML			
REDIBUJA2.AML			
ZOOMREDIBUJA2.AML			
DISENARUTA2.AML			
LIMPIARRUTA2.AML			
IMPRIME2.AML			

6.2.1 ArcInfo Macro Language - AML

```
/*****************************************************************************
/* Archivo:  router2.AML
/*
/* Programa principal de la aplicacion de planeamiento de rutas
/*
/* USO:  &RUN router2 <station_file>
/*
/* LLAMADAS: disena2.AML
/*           salir2.AML
/*           menu2.AML (router2.MENU  disena2.MENU)
/*
/* Rotinas:  creacaminos2
/*
/* GLOBALES: .origen
/*           .destino
/*           .zoom
/*           .print
/*           .device
/*
/* LOCALES:  device
/*
/* HISTORICO: 23/11/1999   creado
/*            07/03/2000   editado, actualizado
/*****************************************************************************

/* Recibe argumentos desde el sistema operativo de ArcInfo y verifica si
/* el <station_file> ha sido pasado.

&args device
&setvar .device = %device%
```

```
&if [null %device%] &then &return Sintaxe: &RUN ROUTER2 <station_file>
&else &do

/* Configurar el entorno AML
&AMLPATH aml
&MENUPATH aml

/* Desabilita la salida de mensajes
&messages &off

/* Carga la tabla de datos y selecciona los registros de interes
&call creacaminos2

  /* Inicialización de las variables globales
  &setvar .origen 'zzz'
  &setvar .destino 'zzz'
  &setvar .zoom = .false.
  &setvar .print = .false.
  &setvar .ruta_sel = .false.

  &do i &list urbscompl edupc viascompl ambsok topsok altsok retsok mobsok camsok
     &setvar .%i% = .false.
     &do j &list d
        &setvar .%i%%j% = .false.
     &end
     &do j &list c s
        &setvar .%i%%j% = %i%
     &end
  &end

  &sv .urbscompl = [value .urbscompld]
  &sv .edupc = [value .edupcd]
  &sv .viascompl = [value .viascompld]
  &sv .ambsok = [value .ambsokd]
  &sv .topsok = [value .topsokd]
  &sv .retsok = [value .retsokd]
  &sv .mobsok = [value .mobsokd]
  &sv .altsok = [value .altsokd]
  &sv .camsok = [value .camsokd]
  &sv .whattodo d

  /* Carga el ArcPlot y configura la pantalla de visualización
  ARCPLOT
  DISPLAY %device% SIZE 1028 740
  &station %device%
  PAGESIZE 40 30

  CLIPMAPEXTENT OFF
  MAPEXT URBSCOMPL

  /* Disena la interface de la aplicación en la pantalla de visualización
  &RUN disena2

  /* Tamano y localizacion del menu(s) en la pantalla
  &RUN menu2 create router2 &pulldown &size &frame 450 50 ~
                   &stripe 'Estudio de Planeamiento de Routas a los Campi de la UPC' ~
                   &position &ul &display &ur &pinaction '&RUN salir2'
  &RUN menu2 create disena2 &stripe 'Disena las Capas' &position &lc &thread router2
  &setvar .width = [extract 1 [show &thread &size disena2]]
  &thread &delete &self

  /* Habilita la salida de mensajes
```

```
  &messages &on

&end

/* Finaliza el programa principal
&return

/*****************************************************************************
/* Rotina:   creacaminos2
/* Rotina que carga la tabla de datos y selecciona los registros de interes
/* USO:  &call creacaminos2
/* LOCALES:  nofares
/*           file
/* HISTORICO: 23/11/1999   creado
/*            17/02/2000   editado, actualizado
/*****************************************************************************

/* Indica el nombre de la rotina
&routine creacaminos2

/* Atribuye a la variable nofares el indicador de eliminación del archivo
&setvar nofares [delete caminos2.dat]

/* Aplicación que permite trabajar con tablas
TABLES

/* Selecciona el archivo de datos (alteracion la rutina de Julio Cesar)
SELECT CAMSOK.AAT

/* Carga los registros seleccionados e itens al archivo ASCII
/* (alteracion la rutina de Julio Cesar)
UNLOAD caminos2.dat FNODE# TNODE# LENGTH CAMSOK-ID

/* Finaliza la ejecución de la aplicación TABLES
QUIT

/* Atribuye a la variable "file" el indicador de apertura del archivo
&setvar file [open caminos2.dat openstatus -read]

/* Atribuye a la variable "nofares" el indicador de cierre del archivo
&setvar nofares [close %file%]

/* Finaliza la rotina y regresa a la llamada
&return

/*****************************************************************************
/* Archivo:  disena2.AML
/*
/* Programa que disena la pantalla principal dela aplicación de planeamiento
/* de rutas
/*
/* USO:  &RUN disena2
/*
/* LLAMADA POR: router2.AML
/*              redibuja2.AML
/*              limpiar2.AML
/*
/* GLOBALES: .origen
/*           .destino
/*           .zoom
/*
/* Rotinas:  disenaescala2
```

```
/*           disenaleyenda2
/*
/* HISTORICO: 24/11/1999    creado
/*            03/03/2000    editado, actualizado
/*****************************************************************************

/* Desabilita la salida de mensajes
&messages &off

/* Limpia la pantalla actual
CLEAR

/* Define que las coordinadas especificas por el usuario serán en
/* unidades de pagina
UNITS PAGE

/* Especifica el archivo de conjunto de colores
SHADESET COLORNAMES
LINESET PLOTTER

/* Selecciona el simbolo de linea actual (cambiar color y estilo)
LINESYMBOL 7

/* Dibuja un "box" utilizado las coordinadas especificadas y
/* el simbolo definido por el LINESYMBOL
BOX .5 .5 39.5 29.5

/* Dibuja lineas entre dos puntos utilizado las coordinadas especificadas
LINE .5 2.5 39.5 2.5
LINE .5 27.5 39.5 27.5
LINE 35.5 0.5 35.5 2.5

/* Define caracteristicas del texto que sigue
TEXTFONT 'COURIER'
TEXTSIZE 1
TEXTFIT 'CAMINOS PEATONALES - ZONA UNIVERSITARIA' 10 28 30 28
TEXTSIZE 1
TEXTFIT 'UNIVERSIDAD POLITECNICA DE CATALUNA' 1 1 35 1

&if ^ %.zoom% &then &do
   &call disenaescala2
   &call disenaleyenda2
   MAPEXT URBSCOMPL
   MAPLIMITS 1 3 39 27
   MAPPOSITION LL 1 3
&end

/* Habilita la salida de mensajes
&messages &on

/* Finaliza el programa disena.aml
&return

/*****************************************************************************
/* Rotina:   disenaescala2
/* Rotina que disena el simbolo del norte geografico y especifica la escala
/* USO:  &call disenaescala2
/*
/* HISTORICO: 24/01/2000    creado
/*            03/01/2000    editado, actualizado
/*****************************************************************************
```

```
/* Indica el nombre de la rotina
&routine disenaescala2

/* Inserta dibujo representativo del norte geografico en la pantalla **********
LINESYMBOL 1

/* Especifica los datos que seran usados para dibujar o consultar el mapa
/* ESCALAOK es una "coverage"
MAPEXT ESCALAOK

/* Especifica el area en la pagina grafica donde los datos seran dibujados
MAPLIMITS 35.6 0.6 39.4 2.4

/* Configura la posicion del mapa en la pagina grafica
MAPPOSITION CEN CEN

/* Dibuja los "arcs" usando el corriente LINESYMBOL
ARCS ESCALAOK

/* Adiciona registro al cunjunto de datos seleccionado para el objeto
/* especificado
ASELECT ESCALAOK POLY

/* Selecciona un cunjunto de registro de objeto especificado
RESELECT ESCALAOK POLY DESC-CARACT = 'FLECHA'

/* Atribuye el color especificado a poligonos
POLYGONSHADES ESCALAOK 33

/* Finaliza la rotina y regresa a la llamada
&return


/*****************************************************************************
/* Rotina:   disenaleyenda2
/* Rotina que disena la leyenda del mapa general
/* USO:  &call disenaleyenda2
/*
/* HISTORICO: 25/02/2000    creado
/*            26/02/2000    editado, actualizado
/*****************************************************************************

&routine disenaleyenda2

LINESYMBOL 1

MAPEXT LEYENDA
MAPLIMITS 30 4 39 13
MAPPOSITION LL 30 4

/* Define caracteristicas del texto de la leyenda
TEXTFONT 'COURIER'
TEXTSIZE 0.8
TEXTFIT 'Leyenda:' 30.5 13 38 13

TEXTFONT 'TRIUMVIRATE'
TEXTSIZE 0.5
TEXTFIT 'Edificios UPC' 32.5 12 38 12
TEXTFIT 'Urbanismo' 32.5 10.8 38 10.8
TEXTFIT 'Viabilidad' 32.5 10 38 10
TEXTFIT 'Ambitos' 32.5 9.2 38 9.2
TEXTFIT 'Topografia' 32.5 8.5 38 8.5
```

```
TEXTFIT 'Altimetria' 32.5 7.7 38 7.7
TEXTFIT 'Retulacion' 32.5 6.9 38 6.9
TEXTFIT 'Posibles Caminos' 32.5 6 38 6
TEXTFIT 'Mobiliario Urbano' 32.5 4.8 38 4.8

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 1
POLYGONSHADES LEYENDA 63

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 2
POLYGONSHADES LEYENDA 1

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 3
POLYGONSHADES LEYENDA 70

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 4
POLYGONSHADES LEYENDA 45

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 13
POLYGONSHADES LEYENDA 8

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 15
POLYGONSHADES LEYENDA 70

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 17
POLYGONSHADES LEYENDA 5

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 19
POLYGONSHADES LEYENDA 110

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 9
POLYGONSHADES LEYENDA 83

/* Finaliza la rotina y regresa a la llamada
&return

/*****************************************************************************
/* Archivo:  salir2.AML
/*
/* Programa que cerra la aplicacion ROUTER2
/*
/* USO:  &RUN salir2
/*
/* LLAMADA POR: router2.AML
/*              router2.MENU
/*
/* LLAMADAS: menu2.AML
/*           limpiarruta2.AML
/*
/* GLOBALES: .ruta_sel
/*
/* HISTORICO: 02/02/2000   creado
/*            07/03/2000   editado, actualizado
/*****************************************************************************
```

```
&if %.ruta_sel% &then &RUN limpiarruta2 .true.

&do i &list disena2 seleccion2 extremos2 presenta2
  &RUN menu2 delete %i%
&end
&dv .*
quit
&thread &delete router2

/*****************************************************************************
/* Archivo:  menu2.AML
/*
/* Programa que controla la presentacion y eliminacion de los menus
/*
/* USO:  &RUN menu2 <funcion> <menu> <parametros>
/*
/* LLAMADA POR: router2.MENU
/*              router2.AML
/*              sair2.AML
/*
/* Rotinas:  create
/*       delete
/*
/* LOCALES: operacion
/*          ventana
/*          detalles:rest
/*
/* HISTORICO: 02/02/2000   creado
/*            02/03/2000   editado, actualizado
/*****************************************************************************

&args operacion ventana detalles:rest
&call %operacion%
&return

/*----------------------
&routine create
&if ^ [show &thread &exists %ventana%] &then &do
   &thread &create %ventana% &menu %ventana% [unquote %detalles%]
&end
&return

/*----------------------
&routine delete
&if [show &thread &exists %ventana%] &then
   &thread &delete %ventana%
&return

/*****************************************************************************
/* Archivo:  limpiar2.AML
/*
/* Programa que limpia el area de trabajo
/*
/* USO:  &RUN limpiar2
/*
/* LLAMADA POR: disena2.MENU
/*              router2.MENU
/*
/* LLAMADAS: disena2.MENU
/*
/* HISTORICO: 02/02/2000   creado
```

```
/*            02/03/2000   editado, actualizado
/*****************************************************************************

clear
&do cov &list urbscompl edupc viascompl ambsok topsok altsok retsok mobsok camsok
   &do mode &list d
     &s .%cov%%mode% = .false.
   &end
   &s .%cov% = .false.
&end

clearselect

&RUN disena2
&return

/*****************************************************************************
/* Archivo:  extremos2.AML
/*
/* Programa que verifica si los puntos de origen y destinos han sido bien
/* escojidos
/*
/* USO:  &RUN extremos2.AML
/*
/* LLAMADA POR: selec2.MENU
/*
/* LLAMADAS: invalidos2.TXT
/*           iguales2.TXT
/*           menu2.AML (extremos2.MENU)
/*
/* GLOBALES: .origen
/*           .destino
/*
/* HISTORICO: 09/02/2000    creado
/*            07/03/2000    editado, actualizado
/*****************************************************************************

&if %.origen% = 'zzz' or %.destino% = 'zzz' &then &POPUP invalidos2.txt
&else &do
  &if %.origen% = %.destino% &then &POPUP iguales2.txt
  &else &RUN menu2 create extremos2 &stripe 'Puntos de la Ruta' &position &center
&end

&return

/*****************************************************************************
/* Archivo:  calculo2.AML
/*
/* Programa que calcula la ruta entre los puntos de origen y destinos
/*
/* USO:  &RUN calculo2.AML
/*
/* LLAMADA POR: extremos2.MENU
/*
/* LLAMADAS: invalidos2.TXT
/*           iguales2.TXT
/*           optimo2.AML
/*
/* Rotinas:  asociacion2
/*
/* GLOBALES: .origen
/*           .destino
```

```
/*           .root
/*           .target
/*
/* HISTORICO: 09/02/2000    creado
/*            07/03/2000    editado, actualizado
/*****************************************************************************

&if %.origen% = 'zzz' or %.destino% = 'zzz' &then &POPUP invalidos2.txt
&else &do
  &if %.origen% = %.destino% &then &POPUP iguales2.txt
  &else &do
    &call asociacion2
    &RUN optimo2
  &end
&end
&return

/*****************************************************************************
/* Rotina:   asociacion2
/* Rotina que inserta valores especificos para el uso posterior en una
/* aplicacion en C
/* USO:  &call asociacion2
/* GLOBALES: .origen
/*           .destino
/*           .root
/*           .target
/* HISTORICO: 09/02/2000   creado
/*            07/03/2000   editado, actualizado
/*****************************************************************************

/* Indica el nombre de la rotina
&routine asociacion2

&if %.origen% = 'A1' &then &setvar .root 243
&if %.origen% = 'A2' &then &setvar .root 245
&if %.origen% = 'A3' &then &setvar .root 247
&if %.origen% = 'A4' &then &setvar .root 249
&if %.origen% = 'A5' &then &setvar .root 251
&if %.origen% = 'A6' &then &setvar .root 253
&if %.origen% = 'B1' &then &setvar .root 234
&if %.origen% = 'B2' &then &setvar .root 207
&if %.origen% = 'B3' &then &setvar .root 211
&if %.origen% = 'B4' &then &setvar .root 213
&if %.origen% = 'B5' &then &setvar .root 215
&if %.origen% = 'B6' &then &setvar .root 226
&if %.origen% = 'C1' &then &setvar .root 368
&if %.origen% = 'C2' &then &setvar .root 369
&if %.origen% = 'C3' &then &setvar .root 232
&if %.origen% = 'C4' &then &setvar .root 230
&if %.origen% = 'C5' &then &setvar .root 228
&if %.origen% = 'C6' &then &setvar .root 282
&if %.origen% = 'D1' &then &setvar .root 203
&if %.origen% = 'D2' &then &setvar .root 201
&if %.origen% = 'D3' &then &setvar .root 199
&if %.origen% = 'D4' &then &setvar .root 197
&if %.origen% = 'D5' &then &setvar .root 195
&if %.origen% = 'D6' &then &setvar .root 193
&if %.origen% = 'FP' &then &setvar .root 187
&if %.origen% = 'TL' &then &setvar .root 191
&if %.origen% = 'BGF' &then &setvar .root 364
&if %.origen% = 'NX' &then &setvar .root 239
&if %.origen% = 'TG' &then &setvar .root 189
```

```
&if %.origen% = 'CS' &then &setvar .root 358
&if %.origen% = 'PAR' &then &setvar .root 414
&if %.origen% = 'A' &then &setvar .root 330
&if %.origen% = 'PG' &then &setvar .root 287
&if %.origen% = 'EU' &then &setvar .root 284
&if %.origen% = 'H' &then &setvar .root 341
&if %.origen% = 'U' &then &setvar .root 336
&if %.origen% = 'PE' &then &setvar .root 419
&if %.destino% = 'A1' &then &setvar .target 243
&if %.destino% = 'A2' &then &setvar .target 245
&if %.destino% = 'A3' &then &setvar .target 247
&if %.destino% = 'A4' &then &setvar .target 249
&if %.destino% = 'A5' &then &setvar .target 251
&if %.destino% = 'A6' &then &setvar .target 253
&if %.destino% = 'B1' &then &setvar .target 234
&if %.destino% = 'B2' &then &setvar .target 207
&if %.destino% = 'B3' &then &setvar .target 211
&if %.destino% = 'B4' &then &setvar .target 213
&if %.destino% = 'B5' &then &setvar .target 215
&if %.destino% = 'B6' &then &setvar .target 226
&if %.destino% = 'C1' &then &setvar .target 368
&if %.destino% = 'C2' &then &setvar .target 369
&if %.destino% = 'C3' &then &setvar .target 232
&if %.destino% = 'C4' &then &setvar .target 230
&if %.destino% = 'C5' &then &setvar .target 228
&if %.destino% = 'C6' &then &setvar .target 282
&if %.destino% = 'D1' &then &setvar .target 203
&if %.destino% = 'D2' &then &setvar .target 201
&if %.destino% = 'D3' &then &setvar .target 199
&if %.destino% = 'D4' &then &setvar .target 197
&if %.destino% = 'D5' &then &setvar .target 195
&if %.destino% = 'D6' &then &setvar .target 193
&if %.destino% = 'FP' &then &setvar .target 187
&if %.destino% = 'TG' &then &setvar .target 189
&if %.destino% = 'TL' &then &setvar .target 191
&if %.destino% = 'BGF' &then &setvar .target 364
&if %.destino% = 'NX' &then &setvar .target 239
&if %.destino% = 'CS' &then &setvar .target 358
&if %.destino% = 'PAR' &then &setvar .target 414
&if %.destino% = 'A' &then &setvar .target 330
&if %.destino% = 'PG' &then &setvar .target 287
&if %.destino% = 'EU' &then &setvar .target 284
&if %.destino% = 'H' &then &setvar .target 341
&if %.destino% = 'U' &then &setvar .target 336
&if %.destino% = 'PE' &then &setvar .target 419

/* Finaliza la rotina y regresa a la llamada
&return

/*****************************************************************************
/* Archivo:  optimo2.AML
/*
/* Programa que crea un fichero temporal y llama la aplicación en C para el
/* calculo de la ruta optima
/*
/* USO:  &RUN optimo2.AML
/*
/* LLAMADA POR: seleccion2.MENU
/*
/* LLAMADAS: vertexs2.TMP
/*           novertexs2.TXT
/*           router2.EXE
```

```
/*           resultado2.AML
/*
/* GLOBALES: .root
/*           .target
/*
/* HISTORICO: 02/03/2000    creado
/*            07/03/2000    editado, actualizado
/*****************************************************************************

&setvar fichero_vertexs2 = vertexs2.tmp
&setvar ident_delete = [delete %fichero_vertexs2%]
&setvar ident_append = [open %fichero_vertexs2% openstatus -append]

&if %openstatus% ^= 0 &then &RUN novertexs2.txt
&else &do /* 1

  &if [write %ident_append% %.root%] <> 0 &then &RUN novertexs2.txt
  &else &do /* do 2

    &if [write %ident_append% %.target%] <> 0 &then &RUN novertexs2.txt
    &else &do /* do 3

      &if [close %ident_append%] <> 0 &then &RUN novertexs2.txt
      &else &do /* do 4

        &SYSTEM router2
        &setvar ident_delete = [delete %fichero_vertexs2%]
        &RUN resultado2

      &end /* end do 4

    &end /* end do 3

  &end /* end do 2

&end /* end do 1
&return

/*****************************************************************************
/* Archivo:  resultado2.AML
/*
/* Programa que prepara a presentacion del resultado en pantalla
/*
/* USO:  &RUN resultado2.AML
/*
/* LLAMADA POR: optimo2.AML
/*
/* LLAMADAS: novertexs2.TXT
/*           router2.EXE
/*           resultado2.AML
/*
/* Rotinas:  disenaruta2
/*           presenta2
/*
/* GLOBALES: .longitud
/*           .tiempo
/*           .origen
/*           .destino
/*
/* HISTORICO: 02/03/2000    creado
/*            07/03/2000    editado, actualizado
/*****************************************************************************
```

```
&setvar file [open resultado2.tmp openstatus -read]
&if %openstatus% <> 0 &then &RUN novertexsvalidos2.txt
&else &do /* do 1

  &setvar .longitud [read %file% readstatus]
  &if %readstatus% <> 0 &then &RUN novertexsvalidos2.txt
  &else &do /* do 2

    &setvar aresta [read %file% readstatus]
    &if %readstatus% <> 0 &then &RUN novertexsvalidos2.txt
    &else &do /* do 3

      &messages &off

      &do &while %readstatus% = 0
        aselect camsok arcs
        reselect camsok arcs camsok-id = %aresta%
        calculate camsok arcs pertany = 1
        aselect camsok arcs
        &setvar aresta [read %file% readstatus]
      &end /* end do-while

      &if [close %file%] ne 0 &then &setvar nofares [delete %file%]
      &setvar .tiempo = %.longitud% / 2
      &setvar .tiempo = %.tiempo% / 60
      &setvar .ruta_sel = .true.

      &RUN disenaruta2
      &RUN menu2 create presenta2 &stripe 'Informaciones sobre la ruta calculada' &position &UL

      &messages &on

    &end /* end do 3

  &end /* end do 2

&end /* end do 1
&return

/*****************************************************************************
/* Archivo:  dibuja2.AML
/*
/* Programa que controla el diseno de las capas seleccionadas
/*
/* USO:  &RUN dibuja2 <capa> <activa>
/*
/* LLAMADA POR: disena2.MENU
/*
/* LLAMADAS: redibuja2.AML
/*           dibujacapa2.AML
/*
/* LOCALES: capa
/*          activa
/*
/* HISTORICO: 02/02/2000   creado
/*            02/03/2000   editado, actualizado
/*****************************************************************************

&args capa activa

&if [keyword %capa% urbscompl edupc viascompl ambsok topsok altsok retsok mobsok camsok] < 0 &then
```

```
    &return &inform Invalid OBJECT %capa% (dibuja2.aml).

/* Define cual de las capas será impresa. Valores:  .true. or .false.
/* (p.ej., &setvar .urbscompl = .true.)
&setvar .%capa%%.whattodo% = %activa%

&if ^ %activa% &then
   &RUN redibuja2
&else
   &RUN dibujacapa2 %capa%
&return

/*****************************************************************************
/* Archivo:  dibujacapa2.AML
/*
/* Programa que disena las capas seleccionadas
/*
/* USO:  &RUN dibujacapa2 <capa>
/*
/* LLAMADA POR: disena2.MENU
/*              dibuja2.AML
/*
/* LOCALES: capa
/*
/* HISTORICO: 02/02/2000   creado
/*            04/03/2000   editado, actualizado
/*****************************************************************************

&args capa

&select %capa%
   &when edupc
    &do
         POLYGONSHADES %capa% 63; ARCLINES %capa% 4; TEXTCOLOR 2; LABELTEXT edupc id-testeed; TEXTCOLOR 1
    &end
   &when viascompl
   &do
      ARCLINES %capa% 3
   &end
   &when mobsok
    &do
         POLYGONSHADES %capa% 83
    &end
   &when urbscompl
   &do
      ARCLINES %capa% 1
   &end
   &when ambsok
   &do
      ARCLINES %capa% 7
   &end
   &when topsok
   &do
      ARCLINES %capa% 8
   &end
   &when retsok
   &do
      ARCLINES %capa% 5
   &end
   &when altsok
   &do
      ARCLINES %capa% 63
```

```
   &end
   &when camsok
   &do
      ARCLINES %capa% 2
   &end
   &otherwise
      &type Capa invalida (dibujacapa2.aml)
&end

&return

/*****************************************************************************
/* Archivo:  redibuja2.AML
/*
/* Programa que redisena las capas seleccionadas
/*
/* USO:  &RUN redibuja2
/*
/* LLAMADA POR: disena2.MENU
/*              router2.MENU
/*              dibuja2.AML
/*
/* LLAMADAS: disena2.AML
/*
/* Globales: .ruta_sel
/*           .urbscompld
/*           .edupcd
/*           .viascompld
/*           .ambsokd
/*           .topsokd
/*           .retsokd
/*           .mobsokd
/*           .altsokd
/*           .camsokd
/*           .print
/*
/* HISTORICO: 02/02/2000   creado
/*             04/03/2000   editado, actualizado
/*****************************************************************************

&if ^ %.print% &then &do
   CLEAR
   &RUN disena2
&end

/* Dibuja todas las capas.
&if %.urbscompld% &then &do
   ARCLINES urbscompl 1
&end
&if %.edupcd% &then &do
   POLYGONSHADES edupc 63; ARCLINES edupc 4; TEXTCOLOR 2; LABELTEXT edupc id-testeed; TEXTCOLOR 1
&end
&if %.viascompld% &then &do
   ARCLINES viascompl 3
&end
&if %.ambsokd% &then &do
   ARCLINES ambsok 7
&end
&if %.topsokd% &then &do
   ARCLINES topsok 8
&end
&if %.retsokd% &then &do
```

```
   ARCLINES retsok 5
&end
&if %.mobsokd% &then &do
   POLYGONSHADES mobsok 83
&end
&if %.altsokd% &then &do
   ARCLINES altsok 63
&end
&if %.camsokd% &then &do
   ARCLINES camsok 2
&end
&if %.ruta_sel% &then &do
   ASELECT CAMSOK ARCS
   RESELECT CAMSOK ARCS PERTANY = 1
   ARCLINES CAMSOK 2
&end

&return

/*****************************************************************************
/* Archivo:  zoomredibuja2.AML
/*
/* Programa que redisena las capas seleccionadas
/*
/* USO:  &RUN zoomredibuja2
/*
/* LLAMADA POR: router2.MENU
/*
/* LLAMADAS: disena2.AML
/*           disenaruta2.AML
/*
/* HISTORICO: 08/02/2000   creado
/*            07/03/2000   editado, actualizado
/*****************************************************************************

&setvar .zoom = .true.
&RUN disena2

/* Dibuja todas las capas.
&if %.urbscompld% &then &do
   ARCLINES urbscompl 1
&end
&if %.edupcd% &then &do
   POLYGONSHADES edupc 63; ARCLINES edupc 4; TEXTCOLOR 2; LABELTEXT edupc id-testeed; TEXTCOLOR 1
&end
&if %.viascompld% &then &do
   ARCLINES viascompl 3
&end
&if %.ambsokd% &then &do
   ARCLINES ambsok 7
&end
&if %.topsokd% &then &do
   ARCLINES topsok 8
&end
&if %.retsokd% &then &do
   ARCLINES retsok 5
&end
&if %.mobsokd% &then &do
   POLYGONSHADES mobsok 83
&end
&if %.altsokd% &then &do
   ARCLINES altsok 63
```

```
&end
&if %.camsokd% &then &do
   ARCLINES camsok 2
&end

&if %.ruta_sel% &then &RUN disenaruta2

&return

/*****************************************************************************
/* Archivo:  disenaruta2.AML
/*
/* Programa que actualiza la capa CAMSOK y quita el dibujo de la ultima ruta
/* seleccionada
/*
/* USO:  &RUN disenaruta2
/*
/* LLAMADA POR: resultado2.AML
/*              zoomredibuja2.AML
/*
/* HISTORICO: 07/03/2000    creado
/*            07/03/2000    editado, actualizado
/*****************************************************************************

LINESYMBOL 1
LINECOLOR 2

ASELECT CAMSOK ARCS
RESELECT CAMSOK ARCS PERTANY = 1
ARCS CAMSOK

&return

/*****************************************************************************
/* Archivo:  limpiarruta2.AML
/*
/* Programa que actualiza la capa CAMSOK y quita el dibujo de la ultima ruta
/* seleccionada
/*
/* USO:  &RUN limpiarruta2 <var_boolean>
/*
/* LLAMADA POR: disena2.MENU
/*              salir.AML
/*
/* LLAMADAS:  redibuja2.AML
/*
/* GLOBALES: .origen
/*           .destino
/*           .ruta_sel
/*
/* LOCALES: salir
/*
/* HISTORICO: 07/03/2000    creado
/*            07/03/2000    editado, actualizado
/*****************************************************************************
&args salir
&messages &off

aselect camsok arcs
calculate camsok arcs pertany = 0
&setvar .origen 'zzz'
&setvar .destino 'zzz'
```

```
&setvar .ruta_sel = .false.

&if ^ %salir% &then &RUN redibuja2

&messages &on
&return

/*****************************************************************************
/* Archivo:  imprime2.AML
/*
/* Programa que imprime las capas activadas
/*
/* USO:  &RUN imprime2
/*
/* LLAMADA POR: disena2.MENU
/*
/* LLAMADAS: redibuja2.AML
/*
/* GLOBALES: .print
/*           .device
/*
/* HISTORICO: 12/02/2000    creado
/*            03/03/2000    editado, actualizado
/*****************************************************************************

/* Elimina el directorio temporal donde se disena el mapa con las capas de interes
&setvar eliminado = [delete CAPA_ACT.TMP -directory]
&setvar .print = .true.

MAP CAPA_ACT.TMP
CLEAR
PAGESIZE 12.905 9.8

/* Diseno con bordes y escala*****************************************************
UNITS PAGE
LINESYMBOL 7
BOX .25 .25 12.655 9.55
LINE .25 .75 12.655 .75
LINE .25 9.05 12.655 9.05
LINE 12 0.25 12 .75
TEXTFONT 'COURIER'
TEXTSIZE 0.3
TEXTFIT 'CAMINOS PEATONALES - ZONA UNIVERSITARIA' .50 9.15 12.405 9.15
TEXTSIZE 0.3
TEXTFIT 'UNIVERSIDAD POLITECNICA DE CATALUNA' .50 .40 11.75 .40

LINESYMBOL 1
MAPEXT ESCALAOK
MAPLIMITS 12.05 0.30 12.65 0.70
MAPPOSITION CEN CEN
ARCS ESCALAOK
ASELECT ESCALAOK POLY
RESELECT ESCALAOK POLY DESC-CARACT = 'FLECHA'
POLYGONSHADES ESCALAOK 33
/* Fin de diseno con bordes y escala*****************************************

/* Fin de diseno de la leyenda***********************************************
MAPEXT LEYENDA
MAPLIMITS 9.68 1.30 12.58 4.24
MAPPOSITION LL 9.68 1.30

TEXTFONT 'COURIER'
```

```
TEXTSIZE 0.25
TEXTFIT 'Leyenda:' 9.84 4.24 12.25 4.24

TEXTFONT 'TRIUMVIRATE'
TEXTSIZE 0.16
TEXTFIT 'Edificios UPC' 10.48 3.92 12.25 3.92
TEXTFIT 'Urbanismo' 10.48 3.52 12.25 3.52
TEXTFIT 'Viabilidad' 10.48 3.26 12.25 3.26
TEXTFIT 'Ambitos' 10.48 3 12.25 3
TEXTFIT 'Topografia' 10.48 2.77 12.25 2.77
TEXTFIT 'Altimetria' 10.48 2.51 12.25 2.51
TEXTFIT 'Retulacion' 10.48 2.25 12.25 2.25
TEXTFIT 'Posibles Caminos' 10.48 1.96 12.25 1.96
TEXTFIT 'Mobiliario Urbano' 10.48 1.56 12.25 1.56

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 1
POLYGONSHADES LEYENDA 63

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 2
POLYGONSHADES LEYENDA 1

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 3
POLYGONSHADES LEYENDA 70

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 4
POLYGONSHADES LEYENDA 45

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 13
POLYGONSHADES LEYENDA 8

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 15
POLYGONSHADES LEYENDA 70

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 17
POLYGONSHADES LEYENDA 5

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 19
POLYGONSHADES LEYENDA 110

ASELECT LEYENDA POLY
RESELECT LEYENDA POLY LEYENDA-ID = 9
POLYGONSHADES LEYENDA 83
/* Fin de diseno de la leyenda*********************************************

MAPEXT URBSCOMPL
MAPLIMITS 0.5 0.5 12.4 9.3
MAPPOSITION LL 0.5 0.5

&RUN redibuja2

MAP END
CLEAR
PLOT CAPA_ACT.TMP
```

```
&setvar .print = .false.

/* Area normal ********************************************************************
DISPLAY %.device% SIZE 1028 740
&station %.device%
PAGESIZE 40 30
CLIPMAPEXTENT OFF
MAPEXT URBSCOMPL
&RUN redibuja2

/* Fin de area normal *************************************************************
&return
```

6.2.2 MENU files

```
/* ROUTER2.MENU ********************************************************************
1 Sample router
'Rutas           '
  'Definir ruta    '  &RUN menu2 create seleccion2 &stripe 'Seleccion de la Ruta' ~
                                                    &position &below &thread disena2
  'Sobre edicicios '  &POPUP edificio2.txt
'Area de trabajo '
  'Disena Capas... '    &RUN menu2 create disena2 &stripe 'Disena las Capas' ~
                                                   &position &lc &thread router2
  'Rediseñar       '    &RUN redibuja2.aml
  'Limpiar mapa    '    &RUN limpiar2.aml
  'Zoom de Area    '    &TYPE Entre con dos puntos de una diagonal; MAPEXT *; &RUN zoomredibuja2.aml
  'Area Normal     '    &setvar .zoom = .false.; &RUN redibuja2.aml
'Créditos          '    &POPUP creditos2.txt
'Salida          '      &RUN salir2.aml


/* DISENA2.MENU ********************************************************************
7
/*
   Disenar las Capas

   %x1  ^Urbanismo  %x6  ^Altimetria
   %x2  ^Edificios  %x7  ^Retulacion
   %x3  ^Viabilidad %x8  ^Mobiliario Urbano
   %x4  ^Ambitos    %x9  ^Posibles Caminos
   %x5  ^Topografia
   %butto0 %button1 %button2
   %button3         %button4
%x1 CHECKBOX .urbscompl KEEP RETURN '&RUN dibuja2 urbscompl %.urbscompl%'
%x6 CHECKBOX .altsok KEEP RETURN '&RUN dibuja2 altsok %.altsok%'
%x2 CHECKBOX .edupc KEEP RETURN '&RUN dibuja2 edupc %.edupc%'
%x7 CHECKBOX .retsok KEEP RETURN '&RUN dibuja2 retsok %.retsok%'
%x3 CHECKBOX .viascompl KEEP RETURN '&RUN dibuja2 viascompl %.viascompl%'
%x8 CHECKBOX .mobsok KEEP RETURN '&RUN dibuja2 mobsok %.mobsok%'
%x4 CHECKBOX .ambsok KEEP RETURN '&RUN dibuja2 ambsok %.ambsok%'
%x5 CHECKBOX .topsok KEEP RETURN '&RUN dibuja2 topsok %.topsok%'
%x9 CHECKBOX .camsok KEEP RETURN '&RUN dibuja2 camsok %.camsok%'
%butto0 BUTTON KEEP 'Limpiar' &RUN limpiar2
%button1 BUTTON KEEP 'Redisenar' &RUN redibuja2
%button2 BUTTON KEEP 'Imprimir' &RUN imprime2
%button3 BUTTON KEEP 'Cancelar' &RETURN
%button4 BUTTON KEEP 'Limpiar Ruta' &RUN limpiarruta2 .false.
%FORMOPT NEXTFIELD ADVANCE SETVARIABLES IMMEDIATE
```

```
/* EXTREMOS2.MENU ************************************************************
7

  La ruta sera calculada entre los siguientes puntos:

  Origen de la ruta:  %program1

  Destino de la ruta: %program2

   %aceptar %cancelar

%program1 display .origen 2 value
%program2 display .destino 2 value
%aceptar button Ok &RUN calculo2 %.origen% %.destino%; &return
%cancelar button Cancelar &setvar .origen 'zz'; &setvar .destino 'zz'; &return

/* SELECCION2.MENU ***********************************************************
7

   Selección de la Ruta
   Origen:                              ^Destino:
   %datalist1                           %datalist2









   %button0                               %button1        %button2
%datalist1 INPUT datalist1 26 KEEP SCROLL YES TYPEIN YES ROWS 10 # RETURN '&sv .origen = %datalist1%'
                                   CHOICE -FILE ..\router2\edcod2.txt -DISPLAY 1 -NOSORT
%datalist2 INPUT datalist2 26 KEEP SCROLL YES TYPEIN YES ROWS 10 # RETURN '&sv .destino = %datalist2%'
                                   CHOICE -FILE ..\router2\edcod2.txt -DISPLAY 1 -NOSORT
%button0 BUTTON KEEP HELP 'Presenta la relación de edificios de la UPC' 'Información sobre los edificios'
                          &popup ..\router2\edificio2.txt
%button1 BUTTON KEEP HELP 'Calcular la ruta optima entre los dos puntos seleccionados'
                          'Calcular' &RUN extremos2.AML
%button2 BUTTON KEEP 'Cancelar' &return
%FORMOPT NEXTFIELD ADVANCE SETVARIABLES IMMEDIATE

/* PRESENTA2.MENU ************************************************************
7
   Datos sobre la ruta calculada:

     Esta es la ruta optima entre:
     Origen: %displa2  ^Destino:%displa3
     La distancia es:  %displa0 ^metro(s)
     Tiempo estimado:  %1       ^minuto(s)

                                %butt0
%displa2 DISPLAY .origen 7 VALUE
%displa3 DISPLAY .destino 7 VALUE
%displa0 DISPLAY .longitud 7 VALUE
%1 DISPLAY .tiempo 7 VALUE
%butt0 BUTTON KEEP 'Salir' &return
%FORMOPT NEXTFIELD ADVANCE SETVARIABLES IMMEDIATE
```

6.2.3 Necessary files - DAT, TXT Y TMP

```
/* CAMINOS2.DAT *******************************************************
295,187,24.61913,1
2,295,54.09769,2
395,2,59.42771,3
396,395,21.33399,4
400,388,88.34045,5
295,3,67.74267,6
4,3,109.22703,7
.
.
.
328,287,19.65264,282
172,328,73.35757,283
177,415,69.34691,288
415,414,11.94517,287
415,333,67.00604,284
328,333,42.40464,285
333,335,150.18319,286

/* EDIFICIO2.TXT ******************************************************
Archivo: edificio2.txt

Clave Descripcion

A1    Pavillon de Aulas 1
A2    Pavillon de Aulas 2
A3    Pavillon de Aulas 3
A4    Pavillon de Aulas 4
A5    Pavillon de Aulas 5
A6    Pavillon de Aulas 6
B1    Edificio departamental
B2    Biblioteca de Caminos
B3    CUP
B4    CUP
B5    Casa del Estudiante
B6    Facultad de Informatica de Barcelona
C1    Edificio departamental
C2    Edificio departamental
C3    Edificio departamental
C4    Edificio departamental
C5    Edificio departamental
C6    Edificio departamental
D1    Edificio departamental
D2    Edificio departamental
D3    Edificio departamental
D4    Edificio departamental
D5    Edificio departamental
D6    Edificio departamental
FP    Fundacion Politecnica de Cataluna
NX    Edificio Nexus
BGF   Biblioteca Gabriel Ferrate
TL    Edificio TELECOS
TG    Torre Girona
EU    Escola Universitaria Politecnica de Barcelona
PG    Pavillon de Gobierno
A     E.T.S. de Arquitectura de Barcelona
PAR   Pavellon de Aulas
H     E.T.S. de Ingenieros Industriales de Barcelona
U     Facultad de Matematicas y Estadistica
```

```
PE    Poliesportivo
CS    Centro de Supercomputacion de Cataluna

/* CREDITOS2.TXT ******************************************************
Proyecto de Planeamiento de Rutas

Desarrollado por:
  Hernane Borges de Barros Pereira

Profesor Orientador:
  Lluis Perez Vidal

Organización:
  Universitat Politècnica de Catalunya
  Departament Llenguatges i Sistemes Informàtics - Secció d'Informàtica Gráfica

Versiones:
    Versión 1.0
    Presentada por Julio Cesar Rodriguez Leon de Santos
    Fecha de presentación: 4 de julio de 1996

    Versión 1.5
    Presentada por Miquel Picart
    Fecha de presentación:

    Versión 2.0
    Presentada por Hernane Borges de Barros Pereira
    Fecha de presentación: 15 de mayo de 2000

/* EDCOD2.TXT *********************************************************
A1    A1...Aulari_1
A2    A2...Aulari_2
A3    A3...Aulari_3
A4    A4...Aulari_4
A5    A5...Aulari_5
A6    A6...Aulari_6
B1    B1...Edifici_departamental
B2    B2...Biblioteca_de_Camins
B3    B3...CUP
B4    B4...CUP
B5    B5...Casa_de_l'Estudiant
B6    B6...Facultat_d'Informatica_de_Barcelona
C1    C1...Edifici_departamental
C2    C2...Edifici_departamental
C3    C3...Edifici_departamental
C4    C4...Edifici_departamental
C5    C5...Edifici_departamental
C6    C6...En_construccio
D1    D1...Edifici_departamental
D2    D2...Edifici_departamental
D3    D3...Edifici_departamental
D4    D4...Edifici_departamental
D5    D5...Edifici_departamental
D6    D6...Edifici_departamental
FP    FP...Fundacio_Politecnica_de_Catalunya
NX    NX...Edifici_Nexus
BGF   BGF...Biblioteca Gabriel Ferrate
TL    TL...Edifici_Telecos
TG    TG...Torre_Girona
EU    EU...Escola_Universitaria_Politecnica_de_Barcelona
PG    PG....Pavello_de_Govern
A     A....E.T.S._d'Arquitectura_de_Barcelona
```

```
PAR   PAR...Aularis
H     H....E.T.S._d'Enginyers_Industrials_de_Barcelona
U     U....Facultat_de_Matematiques_i_Estadistica
PE    PE...Poliesportivo
CS    CS...Centre_de_Supercomputacio_de_Catalunya

/* INVALIDOS2.TXT ******************************************************
Error: Los puntos definidos no son validos.
Sugerencia: Por favor, escoja los puntos de origen y distino
que estan puestos en el menu de seleccion.

/* IGUALES2.TXT ********************************************************
Error: Los puntos origen y distino son los mismos.
Sugerencia: Por favor, escoja diferentes puntos de origen y distino.

/* NOVERTEXS2.TXT ******************************************************
&messages &popup
&type Informacion de error:
&type
&type Atencion: Error en el acceso al fichero vertexs2.tmp.
&type
&messages &on

/* NOVERTEXSVALIDOS2.TXT ***********************************************
Informacio de error:

Atencion: Los vertexs suministrados no son validos.
Por favor, vuelva a la seleccion.

/* VERTEXS2.TMP ********************************************************
203
336

/* RESULTADO2.TMP ******************************************************
812.136413574
32
29
34
53
74
77
78
84
83
75
105
113
156
165
179
178
223
229
236
241
242
245
247
```

6.2.4 C Language

```
// ROUTER2.C ******************************************************
#include "general2.h"

void main()
{
    NODEG *n1, *n2, *g[1000];
    VERTEX node_inici, node_fi, v, origen, destino;
    CJTVERTEXS cv;
    ARESTA id_aresta;
    PES longitud, max_longitud;
    LVERTEXS camino;
    CPAR mh;
    FILE *fcaminos, *frota;
    LLISTAV *en_curs;
    TESTE_INT i;
    int dec, sig;
    static char *ch1;
    char ch2[20];

    max_longitud = 10000.0;

    CrearGraf(g);

    CrearCPAR(&mh);

    CrearCjtVertexs(cv);

    CrearCami(camino);

    fcaminos = fopen("caminos2.dat", "r");
    if (!fcaminos) Error("No ha sido posible abrir el archivo para leitura.");
    else {
       do {

           ObtenirDades(fcaminos, &node_inici, &node_fi, &longitud, &id_aresta);

           if (!feof(fcaminos)) {
              n1=(NODEG *)malloc(sizeof(NODEG));
              n1->successor=node_fi;
              n1->id_aresta=id_aresta;
              n1->longitud=longitud;

              AfegirAdjacenciaGraf(&g[node_inici], n1);

              n2=(NODEG *)malloc(sizeof(NODEG));
              n2->successor=node_inici;
              n2->id_aresta=id_aresta;
              n2->longitud=longitud;

              AfegirAdjacenciaGraf(&g[node_fi], n2);

           }

           v=PrimerNodeGraf(g);

       } while (!feof(fcaminos));

       fclose(fcaminos);
```

```
       while (!DarrerNodeGraf(g, v)) {

          InsertarPesMinheap(&mh, v, max_longitud, -1);

          AfegirVertex(cv, v);

          v=SeguentNodeGraf(g, v);

       }

       AgafarPeticio(&origen, &destino);

       Dijkstra(camino, g, &mh, cv, origen, destino);

       EliminarGraf(g);

       frota=fopen("resultado2.tmp", "w");
       if (!frota) Error("No ha sido posible crear el archivo para escrita: resultado2.tmp.");

       else {

          ch1=ecvt(camino[destino].longitud, 12, &dec, &sig);

          i=0;
          while (i<dec) {
             ch2[i]=ch1[i];
             i++;
          }
          ch2[i]='.';

          for(; i<strlen(ch1); i++)
              ch2[i+1]=ch1[i];
          ch2[i+1]=0;

          fputs(ch2, frota);
          en_curs=camino[destino].lvertexs;
          while (en_curs!=NULL) {
             fputc('\n', frota);

             fprintf(frota, "%d", en_curs->aresta);

             en_curs=en_curs->seg;
          }
          fputc('\n', frota);
          fclose(frota);
       }
    }
return;
}

// FUNCION2.C ******************************************************
#include "general2.h"

// Funciones utilizadas en la aplicacion

// Funcion Error() *** Inicio

void Error(char *tipus)
{
  printf("\nERROR: %s",tipus);
  exit(1);
}
```

```
// Funcion Error() *** Fin

// Funcion ObtenirDades() *** Inicio

void ObtenirDades(FILE *fcaminos, int *vi, int *vf, float *lg, int *a)
{
  char c;
  float f=0.0;
  int div=10;

  if (((c=getc(fcaminos))!='\n') && (!feof(fcaminos))) {

    /* Agafem el vertex inicial. */

    *vi=a_enter(c);
    while ((c=getc(fcaminos)) != ',') {
      *vi=*vi*10;
      *vi=*vi+a_enter(c);
    }

    /* Agafem el vertex final. */
    *vf=a_enter(getc(fcaminos));
    while ((c=getc(fcaminos)) != ',') {
       *vf=*vf*10;
       *vf=*vf+a_enter(c);
    }

    /* Agafem la longitud. */
    *lg=a_enter(getc(fcaminos));
    while ((c=getc(fcaminos)) != '.') {
      *lg=*lg*10;
      *lg=*lg+a_enter(c);
    }

    while (isdigit(c=getc(fcaminos))) {
      f=a_enter(c)%10;
      f=f/div;
      div*=10;
      *lg=*lg+f;
    }

    /* Agafem l'aresta. */
    *a=a_enter(getc(fcaminos));
    while ((c=getc(fcaminos))!='\n') {
      *a=*a*10;
      *a=*a+a_enter(c);
    }

  }
}

// Funcion ObtenirDades() ****** Fin

// Funcion CrearGraf() *** Inicio

void CrearGraf(NODEG *g[])
{
  int i;

  for(i=0; i<1000; i++) {
    g[i]=NULL;
```

```
  }
}

// Funcion CrearGraf() *** Fin

// Funcion CrearRegistreGraf() *** Inicio

void CrearRegistreGraf(NODEG **registre, NODEG *nou)
{
  *registre=nou;
}

// Funcion CrearRegistreGraf() *** Fin

// Funcion AfegirAdjacenciaGraf() *** Inicio

void AfegirAdjacenciaGraf(NODEG **lv, NODEG *nou)
{
  NODEG *punter=*lv;
  NODEG *adjacent;
  NODEG *anterior2=NULL;

  while (punter!=NULL && punter->longitud<nou->longitud) {
    anterior2=punter;
    punter=punter->seg;
  }
  adjacent=(NODEG *)malloc(sizeof(NODEG));

  CrearRegistreGraf(&adjacent, nou);

  if (adjacent==NULL) Error("No hay más memoria");
  adjacent->seg=punter;
  if (anterior2==NULL) *lv=adjacent;
  else anterior2->seg=adjacent;
}

// Funcion AfegirAdjacenciaGraf() *** Fin

// Funcion SeguentNodeGraf() *** Inicio

VERTEX SeguentNodeGraf(GRAF g, VERTEX v)
{
  do {
    v++;
  } while ((g[v]==NULL) && (v<1000));
  return (v);
}

// Funcion SeguentNodeGraf() *** Fin

// Funcion PrimerNodeGraf() *** Inicio

VERTEX PrimerNodeGraf(GRAF g)
{
  VERTEX v=0;

  return (SeguentNodeGraf(g, v));
}

// Funcion PrimerNodeGraf() *** Fin

// Funcion DarrerNodeGraf() *** Inicio
```

```
VERTEX DarrerNodeGraf(GRAF g, VERTEX v)
{
  return (v==1000);
}

// Funcion DarrerNodeGraf() *** Fin

// Funcion CrearCPAR() *** Inicio

void CrearCPAR(CPAR *minh)
{
  int i;

  minh->SL=0;
  for (i=0; i<1000; i++) minh->acces[i]=-1;
}

// Funcion CrearCPAR() *** Fin

// Funcion InsertarPesMinheap() *** Inicio

void InsertarPesMinheap(CPAR *minh, VERTEX v, PES lg, VERTEX darrer)
{
  int index, pare;

  CrearRegistreMinheap(minh, v, lg, darrer);

  index=minh->SL-1;

  pare=CercarPareMinheap(index);

  while(minh->arbre[index].longitud < minh->arbre[pare].longitud) {

     IntercanviarRegistreMinheap(minh, &index, pare);

     pare=CercarPareMinheap(index);

  }
}

// Funcion InsertarPesMinheap() *** Fin

// Funcion CrearRegistreMinheap() *** Inicio

void CrearRegistreMinheap(CPAR *minh, VERTEX v, PES lg, VERTEX darrer)
{
  minh->arbre[minh->SL].node=v;
  minh->arbre[minh->SL].longitud=lg;
  minh->arbre[minh->SL].darrer=darrer;
  minh->acces[v]=minh->SL;
  ++(minh->SL);
}

// Funcion CrearRegistreMinheap() *** Fin

// Funcion CercarPareMinheap() *** Inicio

int CercarPareMinheap(int index)
{
  if (index<3) return(0);
  else {
```

```
    index++;
    if (senar(index)) return (((index-1)/2)-1);
    else return ((index/2)-1);
  }
}

// Funcion CercarPareMinheap() *** Fin

// Funcion IntercanviarRegistreMinheap() *** Inicio

void IntercanviarRegistreMinheap(CPAR *minh, int *index, int substituit)
{
  NODEA aux;

  CopiarRegistreMinheap(minh->arbre[*index], &aux);

  CopiarRegistreMinheap(minh->arbre[substituit], &minh->arbre[*index]);

  minh->acces[minh->arbre[*index].node]=*index;

  CopiarRegistreMinheap(aux, &minh->arbre[substituit]);

  minh->acces[minh->arbre[substituit].node]=substituit;
  *index=substituit;
}

// Funcion IntercanviarRegistreMinheap() *** Fin

// Funcion CopiarRegistreMinheap() *** Inicio

void CopiarRegistreMinheap(NODEA a, NODEA *b)
{
  b->node=a.node;
  b->longitud=a.longitud;
  b->darrer=a.darrer;
}

// Funcion CopiarRegistreMinheap() *** Inicio

// Funcion CrearCjtVertexs() *** Inicio

void CrearCjtVertexs(CJTVERTEXS cjt)
{
  int i;

  for (i=0; i<1000; i++) cjt[i]=-1;
}

// Funcion CrearCjtVertexs() *** Fin

// Funcion AfegirVertex() *** Inicio

void AfegirVertex(CJTVERTEXS cjt, VERTEX v)
{
  cjt[v]=v;
}

// Funcion AfegirVertex() *** Fin

// Funcion CrearCami() *** Inicio

void CrearCami(LVERTEXS camino)
```

```
{
  int i;

  for (i=0;i<1000;i++) {
    camino[i].lvertexs=NULL;
  }
}

// Funcion CrearCami() *** Fin

// Funcion AgafarPeticio() *** Inicio

void AgafarPeticio(VERTEX *vi, VERTEX *vf)
{
  FILE *fvertexs;
  char c;

  fvertexs=fopen("vertexs2.tmp", "r");

  if (!fvertexs) Error("No es posible abrir el archivo temporal.");
  else {
    c=getc(fvertexs);
    do {
      *vi=*vi + a_enter(c);
      *vi=*vi * 10;
    } while ((c=getc(fvertexs))!='\n');
    *vi = *vi / 10;
    c=getc(fvertexs);
    do {
      *vf=*vf + a_enter(c);
      *vf=*vf * 10;
    } while ((c=getc(fvertexs))!='\n');
    *vf = *vf / 10;
    fclose(fvertexs);
  }
}

// Funcion AgafarPeticio() *** Fin

// Funcion Dijkstra() *** Inicio

// void Dijkstra(LVERTEXS camino, GRAF g, CPAR *minh, CJTVERTEXS cjt, VERTEX origen, VERTEX destino)
void Dijkstra(LVERTEXS camino, NODEG *g[1000], CPAR *minh, CJTVERTEXS cjt, VERTEX origen, VERTEX destino)
{
  NODEG *p, *anterior=NULL;
  NODEA a;
  PES etiqueta;
  LLISTAV *lv1, *lv2;
  int s, condicion;
  ARESTA edge;

  EliminarNodeMinheap(minh, origen);

  cjt[origen]=-1;
  p=g[origen];

  while (p!=NULL) {

     ModificarPesMinheap(minh, p->successor, p->longitud, origen);

     p=p->seg;
  }
```

```
  CrearNodeA(origen, 0.0, &a);

  AfegirLongitudCami(camino, a);

  do {

     a=MinimMinheap(minh);

     p=g[a.node];

     AfegirLongitudCami(camino, a);

     anterior=(NODEG *)malloc(sizeof(NODEG));
     anterior=g[a.node];
     condicion=anterior->successor!=a.darrer;
     while (condicion && anterior!=NULL ) {
        if ( anterior->seg!=NULL ) {
           anterior=anterior->seg;
           condicion=anterior->successor!=a.darrer;
        }
        else   condicion=0;
     }

     lv1=(LLISTAV *)malloc(sizeof(LLISTAV));

     edge=anterior->id_aresta;
     lv1->aresta=anterior->id_aresta;
     lv1->seg=camino[a.darrer].lvertexs;
     camino[a.node].lvertexs=lv1;
     cjt[a.node]=-1;

     EsborrarMinimMinheap(minh);

     while (p!=NULL) {
        s=p->successor;
        if (cjt[p->successor]>0) {
           int condicion1;
           etiqueta=a.longitud+p->longitud;

           condicion1 = etiqueta < ConsultarPesMinheap(minh, p->successor);

           if (condicion1) {

              ModificarPesMinheap(minh, p->successor, etiqueta, a.node);

              lv2=(LLISTAV *)malloc(sizeof(LLISTAV));
              edge=p->id_aresta;
              lv2->aresta=p->id_aresta;
              lv2->seg=camino[a.node].lvertexs;
              camino[p->successor].lvertexs=lv2;
           }
        }
        p=p->seg;
     }

  AfegirLongitudCami(camino, a);

  } while (a.node!=destino);
}

// Funcion Dijkstra() *** Fin
```

```
// Funcion EliminarNodeMinheap() *** Inicio

void EliminarNodeMinheap(CPAR *minh, VERTEX v)
{
  int index1, index2, pare;

  index1=minh->acces[v];
  index2=minh->SL-1;

  CopiarRegistreMinheap(minh->arbre[index2], &minh->arbre[index1]);

  minh->acces[minh->arbre[index1].node]=index1;
  --(minh->SL);

  pare=CercarPareMinheap(index1);

  while(minh->arbre[index1].longitud < minh->arbre[pare].longitud) {

     IntercanviarRegistreMinheap(minh, &index1, pare);

     pare=CercarPareMinheap(index1);

  }
}

// Funcion EliminarNodeMinheap() *** Fin

// Funcion ModificarPesMinheap() *** Inicio

void ModificarPesMinheap(CPAR *minh, VERTEX v, PES lg, VERTEX darrer)
{
  int index1, index2, pare, fill;
  NODEA aux;

  index1=minh->acces[v];
  index2=minh->SL-1;

  CopiarRegistreMinheap(minh->arbre[index1], &aux);

  CopiarRegistreMinheap(minh->arbre[index2], &minh->arbre[index1]);

  minh->acces[minh->arbre[index1].node]=index1;
  --(minh->SL);

  pare=CercarPareMinheap(index1);

  while(minh->arbre[index1].longitud < minh->arbre[pare].longitud) {

     IntercanviarRegistreMinheap(minh, &index1, pare);

     pare=CercarPareMinheap(index1);

  }

  FillMesPetitMinheap(minh, index1, &fill);

  while( fill != 0 ) {

     IntercanviarRegistreMinheap(minh, &index1, fill);

     FillMesPetitMinheap(minh, index1, &fill);
```

```
  }

  InsertarPesMinheap(minh, aux.node, lg, darrer);

}

// Funcion ModificarPesMinheap() *** Fin

// Funcion CrearNodeA() *** Inicio

void CrearNodeA(VERTEX v, PES lg, NODEA *a)
{
  a->node=v;
  a->longitud=lg;
}

// Funcion CrearNodeA() *** Fin

// Funcion AfegirLongitudCami() *** Inicio

void AfegirLongitudCami(LVERTEXS camino, NODEA a)
{
   camino[a.node].longitud=a.longitud;
}

// Funcion AfegirLongitudCami() *** Fin

// Funcion MinimMinheap() *** Inicio

NODEA MinimMinheap(CPAR *minh)
{
  return (minh->arbre[0]);
}

// Funcion MinimMinheap() *** Fin

// Funcion EsborrarMinimMinheap() *** Inicio

void EsborrarMinimMinheap(CPAR *minh)
{
  int index, fill;

  index=minh->SL-1;

  CopiarRegistreMinheap(minh->arbre[index], &minh->arbre[0]);

  --(minh->SL);
  index=0;

  FillMesPetitMinheap(minh, index, &fill);
  while (fill != 0) {

     IntercanviarRegistreMinheap(minh, &index, fill);

     FillMesPetitMinheap(minh, index, &fill);

  }
}

// Funcion EsborrarMinimMinheap() *** Fin
```

```
// Funcion FillMesPetitMinheap() *** Inicio

void FillMesPetitMinheap(CPAR *minh, int pare, int *nou_fill)
{
  int fill, rightsoon, leftsoon;
  PES pes_pare, pes_fill_dret, pes_fill_esq, pes_fill;

  // printf("pes_pare: %f", pes_pare);   // temporaria

  pes_pare=minh->arbre[pare].longitud;
  if (pare==0) rightsoon=1;
  else rightsoon=(2*pare)+1;

  leftsoon=rightsoon+1;
  pes_fill_dret=minh->arbre[rightsoon].longitud;
  pes_fill_esq=minh->arbre[leftsoon].longitud;
  if (minh->SL <= rightsoon) *nou_fill=0;
  else if (minh->SL-1 == rightsoon) *nou_fill=((pes_pare<pes_fill_dret)?0:rightsoon);
  else if (pes_fill_dret<0) *nou_fill=0;
       else {
          if (pes_fill_esq<0) *nou_fill=((pes_pare<pes_fill_dret)?0:rightsoon);
             else {
                pes_fill=min(pes_fill_dret, pes_fill_esq);
                fill=(pes_fill_dret < pes_fill_esq)?rightsoon:leftsoon;
                if (pes_fill < pes_pare) *nou_fill=fill;
                else *nou_fill=0;
             }
       }
}

// Funcion FillMesPetitMinheap() *** Fin

// Funcion ConsultarPesMinheap() *** Inicio

PES ConsultarPesMinheap(CPAR *minh, VERTEX v)
{
  return (minh->arbre[minh->acces[v]].longitud);
}

// Funcion ConsultarPesMinheap() *** Fin

// Funcion EliminarGraf() *** Inicio

void EliminarGraf(GRAF g)
{
  int i=0;

  while (g[i]!=NULL) {

     EsborrarAdjacents(&g[i], i);

     i++;
  }
}

// Funcion EliminarGraf() *** Fin

// Funcion EsborrarAdjacents() *** Inicio

void EsborrarAdjacents(GRAF g, VERTEX i)
{
  NODEG *en_curs;
```

```
  NODEG *anterior;

  en_curs=g[i];
  while (en_curs!=NULL) {
    anterior=en_curs;
    en_curs=en_curs->seg;
    free(anterior);
  }
  g[i]=NULL;
}

// Funcion EsborrarAdjacents() *** Fin

// GENERAL.H ********************************************************
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <malloc.h>
#include <string.h>

#define NODEG struct nodeg
#define LLISTAV struct llistav

#define senar(x) ((x%2==1)?1:0)
#define a_caracter(x) (x+48)
#define a_enter(ch) (ch-48)

typedef int ARESTA;
typedef int BOOLEA;
typedef float PES;
typedef int VERTEX;
typedef VERTEX CJTVERTEXS[1000];
typedef int TESTE_INT;

/*************************************************************************/
/* Estructura de Datos GRAF: Lista de adyacentes implementada con vector.*/
/*      Vector de punteros a la lista de adyacentes de cada nodo -> GRAF */
/*      Nodos de la lista de adyacentes -> NODEG                         */
/*************************************************************************/

NODEG {
  VERTEX successor;
  PES longitud;
  ARESTA id_aresta;
  NODEG *seg;
};

typedef NODEG *GRAF[1000];

/*************************************************************************/
/* Estructura de Datos CPAR: implementacion de una cola con prioridad    */
/* de acceso rapido, la cola es un vector y un apuntador a la ultima     */
/* posicion libre, y el acceso rapid se hace mediante un segundo vector  */
/* que apunta a la posicion que ocupa el nodo en cuestion al vector de   */
/* minheap.                                                              */
/*      Vector de apuntadores a nodos del Minheap -> VMINHEAP            */
/*      Vector propio de Minheap -> ARBRE                                */
/*      Informacion que contiene cada nodo del minheap -> NODEA          */
/*      Apuntador a la ultima posicion ocupada del vector -> SL          */
/*************************************************************************/
```

```
typedef struct {
  VERTEX node;
  PES longitud;
  VERTEX darrer;
} NODEA;

typedef struct {
  NODEA arbre[1000];
  int SL;
  int acces[1000];
} CPAR;

/*************************************************************************/
/* Estructura de Datos Lista de aristas: implementada con vector.        */
/* Informacion que contiene cada posicion del vector -> NODEC            */
/*************************************************************************/

LLISTAV {
  ARESTA aresta;
  LLISTAV *seg;
};

typedef struct {
  LLISTAV *lvertexs;
  PES longitud;
} NODEC;

typedef NODEC LVERTEXS[1000];

// Prototipos de las funciones utilizadas en la aplicacion *** Inicio

void Error(char *tipus);

void ObtenirDades(FILE *, int *, int *, float *, int *);
void AgafarPeticio(VERTEX *vi, VERTEX *vf);

void CrearGraf(NODEG *g[]);
void AfegirAdjacenciaGraf(NODEG **lv,NODEG *nou);
void EliminarGraf(GRAF g);
void EsborrarAdjacents(GRAF g, VERTEX i);
VERTEX PrimerNodeGraf(GRAF g);
VERTEX SeguentNodeGraf(GRAF g, VERTEX v);
VERTEX DarrerNodeGraf(GRAF g, VERTEX v);

void CrearCPAR(CPAR *minh);
void EliminarNodeMinheap(CPAR *minh, VERTEX v);
void EsborrarMinimMinheap(CPAR *minh);
void InsertarPesMinheap(CPAR *minh, VERTEX v, PES longitud, VERTEX darrer);
void ModificarPesMinheap(CPAR *minh, VERTEX v, PES longitud, VERTEX darrer);
void CrearNodeA(VERTEX v, PES longitud, NODEA *a);
NODEA MinimMinheap(CPAR *minh);
PES ConsultarPesMinheap(CPAR *minh, VERTEX v);
void CopiarRegistreMinheap(NODEA a, NODEA *b);
void CrearRegistreMinheap(CPAR *minh, VERTEX v, PES longitud, VERTEX darrer);
void IntercanviarRegistreMinheap(CPAR *minh, int *index, int subst);
int CercarPareMinheap(int index);
void FillMesPetitMinheap(CPAR *minh, int pare, int *fill);

void CrearCjtVertexs(CJTVERTEXS cjt);
void AfegirVertex(CJTVERTEXS cjt, VERTEX v);

void CrearCami(LVERTEXS camino);
```

```
void AfegirLongitudCami(LVERTEXS camino, NODEA a);

void Dijkstra(LVERTEXS camino, NODEG *g[], CPAR *minh, CJTVERTEXS cjt, VERTEX origen, VERTEX destino);
```

6.3 Digital Cartography

The digital cartography has been acquired of the *Oficina de Atención al Ciudadano* addressed on the Plaza San Miquel in Barcelona. In the Table 4, we show the digital cartography price.

Table 4: The digital cartography price. Sources: Microstation format (.DGN). Price for "cell" (Date: Juny 29, 1999).

Group	Code	Description	Price	Note	Scale
SMAG	3111	Way sheet	950	Geo-referenced	1:500
SMAG	3115	Cadastral sheet (including way)	2.400	Geo-referenced	1:500
SMAG	3116	Urban sheet (including way and cadastral sheets)	3.050	Geo-referenced	1:500
SMAG	3117	Altitudinal sheet	650	-	1:500
SMAG	3120	Urban and altitudinal sheet	3.700	Geo-referenced	1:500

In the following seven pages, we show whole element description provided by *Institut Municipal d'Informàtica - Infomarció de Base i Cartografia.*

Element descriptions (I)

Element descriptions (II)

Element descriptions (III)

Element descriptions (IV)

Element descriptions (V)

Element descriptions (VI)

Element descriptions (VII)

References

Aho, A. V., Hopcroft, J. E. and Ullman, J. D. (1983). *Data structure and algorithms*, Addison-Wesley, Delaware.

Beslon, G., Biennier, F. and Hirsbrunner, B. (1998). Multi-robot path-planning based on implicit cooperation in a robotic swarm, *Autonomous Agents 98*, Minneapolis, pp. 39–46.

Bestavros, A. and Matta, I. (1997). Load profiling for efficient route selection in multi-class networks, *Proceedings of the 1997 International Conference on Network Protocols - ICNP '97*, IEEE, pp. 183–190.

Chen, D. Z. (1996). Developing algorithms and software for geometric path planning problems, *ACM Computing Surveys*.

Decasper, D., Dittia, Z., Parulkar, G. and Plattner, B. (1998). Router plugins: A software architecture for next generation routers, *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, ACM SIGCOMM, Vancouver, pp. 229–240.

Derniame, J. C. and Pair, C. (1971). *Problèmes de Cheminement dans les Graphes*, Dunod Éditeur, Paris.

Dommety, G., Veeraraghavan, M. and Singhal, M. (1997). Route optimization in mobile atm networks, *Proceedings of the third annual ACM/IEEE international conference on Mobile computing and networking*, pp. 43–54.

E.S.R.I. (1992). *Understanding GIS. The ArcInfo Method*, Environmental Systems Research Institute, INC. Rev. 6.

E.S.R.I. (1998). *Help of ArcInfo: ArcDoc Version 7.2.1*, eletronic edn, Environment Systems Research Institute, Inc.

Gerke, M. (1998). Optimal route planning for wheeled mobile robots - an optical approach, *Proceedings of the ...*, IEEE, pp. 2463–2468.

Gerla, M. and Tsai, J. T. C. (1995). Multicluster, mobile, multimedia radio network, *Wireless Networks* **1**: 255–265.

Goto, Y., Matsuzaki, K., Kweon, I. and Obatake, T. (1986). Cmu sidewalk navigation system: A blackboard-based outdoor navigation system using sensor fusion with colored-range images, *IEEE* pp. 105–113.

Gudaitis, M., Lamont, G. and Terzuoli, A. (1995). Multicriteria vehicle route-planning using parallel a search, *Proceedings of the 1995 ACM symposium on Applied computing*, pp. 171–176.

Hwang, Y. K. and Ahuja, N. (1992). Gross motion planning - a survey, *ACM Computing Surveys* **24**(3): 219–291.

Iakovou, E., Douligeris, C., Li, H., IP, C. and Yudhbir, L. (1999). A maritime global route planning model for hazardous materials transportation, *Transportation Science* **33**(1): 34–48.

Kannan, G., Martinez, J. C. and Vorster, M. C. (1997). A framework for incorporating dynamic strategies in earth-moving simulations, *in* D. H. W. S. Andradóttir, K. J. Healy and B. L. Nelson (eds), *Proceedings of the 1997 Winter Simulation Conference*, pp. 1119–1126.

Kingsley, L. C., Klesczewski, K. S. and Smith, J. A. (1998). A logistics model of coast guard buoy tending operations, *in* P. H. M. Abrams and J. Comfort (eds), *Proceedings of the 1998 Winter Simulation Conference*, pp. 753–760.

La Porta, T. F., Sabnani, K. and Gitlin, R. D. (1996). Challenges for nomadic computing mobility management and wireless communications, *Mobile Networks and Applications* **1**(1): 3–16.

Lapalme, G. et al. (1992). A geographic information system for transportation applications, *Communications of the ACM* **35**(1): 80–88.

Laurini, R. and Thompson, D. (1992). *Fundamentals of Spatial Information Systems*, Academic Press Limited, London.

Lee, J. J. and Fishwick, P. A. (1995). Simulation-based real-time decision making for route planning, *in* W. R. L. C. Alexopoulos, K. Kang and D. Goldsman (eds), *Proceedings of the 1995 Winter Simulation Conference*, pp. 1087–1095.

Maffeis, S., Bischofberger, W. and Mätzel, K. U. (1996). A generic multicast transport service to support disconnected operation, *Wireless Networks* **2**: 87–96.

Martin, P. (1999). Train performance and simulation, *in* D. T. S. P. A. Farrington, H. B. Nembhard and G. W. Evans (eds), *Proceedings of the 1999 Winter Simulation Conference*, pp. 1287–1294.

Masson, C., Escassut, R., Barbier, D., Winer, D. and Chevallier, G. (1991). Object oriented lisp implementation of the cheops vlsi floor planning and routing system, *28th Conference on ACM/IEEE design automation conference*, San Francisco, pp. 259–264.

Medhi, D. (1995). Multi-hour, multi-traffic class network design for virtual path-based dynamically reconfigurable wide-area atm networks, *IEEE/ACM Transations on Networking* **3**(6): 809–818.

Pellazar, M. B. (1998). Vehicle route planning with constraints using genetic algorithms, *Proceedings of the ...*, IEEE, pp. 392–399.

Pijls, W. and Kolen, A. (1992). A general framework for shotest path algorithms, *WoPEc* pp. 1–20. http://netec.mcc.ac.uk/WoPEc/data/Papers/dgreureco199775.html.

Prasad, T. V. (1990). An expert system for travel planning in indian railroads, *ACM* pp. 579–587.

Prieto, J. A., Rueda, A., Quintana, J. M. and L., H. J. (1997). A performance-driven placement algorithm with simultaneous place/route optimization for analog ic's, *Proceedings of the 1997 European Design and Test Conference - ED/TC '97*, IEEE, pp. 389–394.

Rabuske, M. A. (1992). *Introdução à Teoria dos grafos*, Editora da UFSC, Florianópolis.

Strothotte, T., Fritz, S., Michel, R., Raab, A., Petrie, H., Johnson, V., Reichert, L. and Schalt, A. (1996). Development of dialogue systems for a mobility aid for blind people: Initial design and usability testing, *ASSETS '96*, Vancouver, pp. 139–144.

Wang, T., Mehdi, Q. H. and Gough, N. E. (1998). An object-oriented environment database or agv path planning, *Proceedings of the Technology of Object-Oriented Languages and Systems-Tools*, IEEE, pp. 86–95.

Wiley, R. B. and Keyser, T. K. (1998). Discrete event simulation experiments and geographic information systems in congestion management planning, *in* J. D.J. Medeiros, E.F. Watson and M.S.Manivannan (eds), *Proceedings of the 1998 Winter Simulation Conference*, pp. 1087–1093.

Wu, Y.-L. and Marek-Sadowska, M. (1995). Orthogonal greedy coupling - a new optimization approach to 2-d fpga routing, *32nd ACM/IEEE Design Automation Conference*.