



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

Computers & Operations Research 31 (2004) 2263–2278

computers &
operations
research

www.elsevier.com/locate/dsw

A distributed geographic information system for the daily car pooling problem[☆]

Roberto Wolfler Calvo^{a,*}, Fabio de Luigi^b, Palle Haastrup^c, Vittorio Maniezzo^d

^a*Laboratoire d'Optimisation des systèmes Industriels (LOSI), Université de Technologie de Troyes, 12, rue Marie Curie - BP 2060 - 10010 Troyes, France*

^b*Department of Computer Science, University of Ferrara, Italy*

^c*Joint Research Center (JRC), European Commission, Ispra, Italy*

^d*Department of Computer Science, University of Bologna, Italy*

Abstract

Following the difficulty of public transport to adequately cover all passenger transportation needs, different innovative mobility services are emerging. Among those are car pooling services, which are based on the idea that sets of car owners having the same travel destination share their vehicles. Until now these systems have had a limited use due to lack of an efficient information processing and communication support. In this study an integrated system for the organization of a car pooling service is presented, using several current Information and Communication Technologies (ICT's) technologies: web, GIS and SMS. The core of the system is an optimization module which solves heuristically the specific routing problem. The system has been tested in a real-life case study.

© 2003 Elsevier Ltd. All rights reserved.

Keywords: Carpool; ICT technologies; Routing and scheduling; GIS

1. Introduction

The increased human mobility combined with high use of private cars increases the load on the environment and raises issues about the quality of life. The use of private cars lends to high levels of air pollution in cities, parking problems, noise pollution, congestion and the resulting low transfer velocity (and thus inefficiency in the use of public resources). Public transport service is often incapable of effectively servicing non-urban areas where cost-effective transportation systems cannot

[☆] Neither the European Commission nor any person acting on behalf of the Commission is responsible for the use which might be made of the following information.

* Corresponding author.

E-mail address: roberto.wolfler@utt.fr (R. Wolfler Calvo).

be set up. Moreover, the needs for transport of people and goods are increasing both in time and space, which further stresses the inadequacies of public transport systems.

A solution to the problem of the increased passenger and freight transport demand could be obtained by increasing both the efficiency and the quality of public transport systems, and by the development of systems which could provide alternative solutions in terms of flexibility and costs between the public and the private ones. This is the rationale behind so-called Innovative Transport Systems (ITS) [1], which are characterized by the exploitation of innovative organizational elements and by a large flexibility in their management (e.g., traffic restrictions and fares can vary according to the time of the day).

New transport alternatives can be classified on the basis of the different aspects of daily mobility: some of them try to improve the average level of car occupancy (*Car Pooling*), others discourage the use of the private car introducing costs (*Park Pricing* and *Road Pricing*) or making public transportation competitive and appealing (*Dial-a-Ride*, *Park-and-Ride* and *Card Car*). Past experience has already shown that the effectiveness and competitiveness of public transport, compared to private transport, can be improved through an integrated process based on data management facilities. In general potential customers of a new transport system face serious problems of knowledge retrieval, which is the reason to devote substantial efforts to provide clients with an easy and powerful tool to find information. Current information systems give the possibility of real time data management and include the possibility to react to unexpected events, as they can be reached from almost anywhere.

This article describes an integrated ICT system which supports the management of a car pooling service in a real-world prototypical setting. Car pooling is a collective transportation system based on a shared use of private cars (vehicles) with the objective to reduce the number of cars on the road. Car pooling is most effective for daily commuters. It is an old idea, but scarcely applied, at least in Europe, for many reasons. First of all, it is necessary to collect a lot of information about service users and to make them available to the service planner: it can be difficult to cluster people in such a way that everybody feels satisfied. Ultimately, it is important that all the participant perceive a clear personal gain in order to continue using the service.

The effectiveness of the proposed system is strictly related to the architecture and the techniques used to manage information. Architectures for information browsing, like the World Wide Web (www), are easy and powerful means to deploy databases through the Internet and thus represent obvious options for setting up services such as the one advocated in this work. The www is useful to provide access to central data storage, to collect remote data and to allow GIS and optimization software to interact. All these parts have an important impact on the implementation of new transport services. Moreover, a central feature of the proposed architecture is the idea of using an optimization algorithm to obtain a matching among clients.

The *Daily Car Pooling Problem* (DCPP) has been modelled as a vehicle routing problem with pickup and deliveries and time windows, and solved with a tailored heuristic. Specifically, it is known that the DCPP is actually a special case of *Dial-a-Ride Problem* (DARP) to which solution techniques from the literature cannot be applied due to the size and peculiarity of the application faced. Thus, a heuristic specific to the DCPP was designed, based on dual costs approximations (regrets) and iterated local optimization. Furthermore the optimization algorithm is able to cope with time-varying road costs. The objective of this research is to prove that, at least in a particular site (the Joint Research Center (JRC) of the European Commission located in Ispra, northern Italy), it could be possible to reduce the number of transport vehicles without changing neither the number

of commuters nor their comfort level significantly. The users of the system are commuters normally using their own cars for travelling between home and a workplace poorly served by public transport.

The paper is structured as follows. Section 2 presents the problem and the relevant literature. Section 3 describes in details the system architecture. Section 4 presents the algorithm used to solve the core combinatorial optimization problem. The computational results are presented in Section 5 and some conclusions close the paper.

2. Car pooling

Car pooling is a collective transportation system based on a shared use of private cars (vehicles), whose objective is to reduce the number of cars in use by grouping people. Car pooling can be operated in two main ways: *Daily Car Pooling Problem* (DCPP) or *Long-term Car Pooling Problem* (LCPP).

In the case of DCPP [2,3], each day a number of users (*servers*) declare their availability for picking up and later bringing back colleagues (*clients*) on that particular day. The problem is to assign clients to servers and to identify the routes to be driven by the servers in order to minimize service costs and a penalty due to unassigned clients, subject to user time window and car capacity constraints.

In the case of LCPP, each user is available both as a server and as a client and the objective is to define crews—or *user pools*—where each user will in turn, on different days, pick up the remaining pool members [4,5]. The objective here becomes that of maximizing pool sizes and minimizing the total distance travelled by all users when acting as servers, again subject to car capacity and time window constraints.

Another important distinction among services is based on the features of the car pooling system, which can be classified in two types: those where there is a www site collecting information about trips (see for example [6–9]), which are open to every web navigator, and those where the users of the system are a restricted group.

A main issue for the first type is to guarantee the reliability of the information. Their interface is often designed mainly to help setting service-related geographical information (customer delivery points, customers pick up points, paths). Such systems will only rarely suggest a matching between clients and servers, but operates as a “Post-it wall”, where everybody can consult or leave information about travel routes.

As for the latter type, the idea is normally to set up a car pooling service among the employees of the same organization. This system is more structured. Moreover, the spontaneous user matching is substituted by a solution found by means of an algorithmic approach. An example of this type of system is reported in [10]. The system uses the www and reacts to unexpected event with email messages. The weak points of the system proposed are the methods used to manage the geographical data and the methods used to cluster the people involved.

This paper suggests an approach similar to [10] and describes a complete system for supporting the operation of a DCPP case as a prototype for a real-life application. The service is supported by a database of potential users (employees of a company) that daily commute from their house to their workplace. A subset of them offer seats in their cars. Moreover, they specify the departure time (when they leave their house) and the mandatory arrival time at the office. The employees that offer seats in their cars are named *servers*. The employees asking for a lift are named *clients*. The set of *servers* and the set of *clients* needs to be redefined once a day.

3. The system

The deployment of a car pooling service should be supported by multiple functionalities. Given the number and the complexity of the operations to be performed, it is necessary to design a multi-module system, with different modules dedicated to different macro functions, such as user interface, communication, optimization, etc.

The described system is designed to support two types of users: the system administrator and the employees. The system administrator must update the static databases (users, maps) and guarantee all the functionalities. All other data are entered, edited and deleted by the users through a distributed GUI.

Data collected and used by the system are of two different types: geographic and alphanumeric. The geographic database contains the maps of the region of interest with a detailed road network, the geocoded client pickup sites and all the geographic information needed to generate a GIS output on the web (see Fig. 3). The alphanumeric data repository, maintained by a relational database, contains information about the employees and a representation of the road network of the area where the car pooling service is active. Other data, input by the users, are strictly related to the daily situation: detailed information about the service, whether a user is a server or a client, the departure time from home and the maximal acceptable arrival time at work, the number of available seats in the cars offered by the servers and the maximum accepted delay, which is the parameter used to specify how far “out of the shortest way” a server is willing to drive in order to pick up colleagues. The users are permitted to consult, modify or delete their own entries in the database at anytime.

The road network and all user-related data are processed by the optimization module in order to define user pools and car paths. The optimization algorithm can be seen as an agent being activated without immediate presence of anybody and on some regular schedule, searching a local database and presenting the results on the web. All these actions are performed on a periodic basis, on the evening before each working day.

One of the most interesting features of the approach described in this paper, with respect to the other systems currently in use, is the set of algorithms used to obtain a matching of clients and servers. These are part of the optimization module of the system. This also includes a modified version of the Dijkstra algorithm, which considers congestions and allows the optimization module to construct tailored solutions which considered congestions, i.e., different travel times at different hours of the day. Due to the size of the instances to solve, a heuristic approach was in order. In designing it, the efficiency was the main parameters, ensuring relatively fast response time also for larger instances. This approach ruled out sophisticated metaheuristic approaches and exact methods and left us with the need to design an effective construction-iterated local optimization procedure.

The matching obtained minimizes the total travel length of all servers going to the workplace and maximizes the number of clients serviced, while meeting the operational constraints. In addition, real time services are supported: for example, the system sends a warning to the employee involved should delays occur.

3.1. System interface

The system supports three different communication channels: web, SMS and emails. The web is the main interface to connect a user to the system (see Fig. 1). The user, by means of a standard

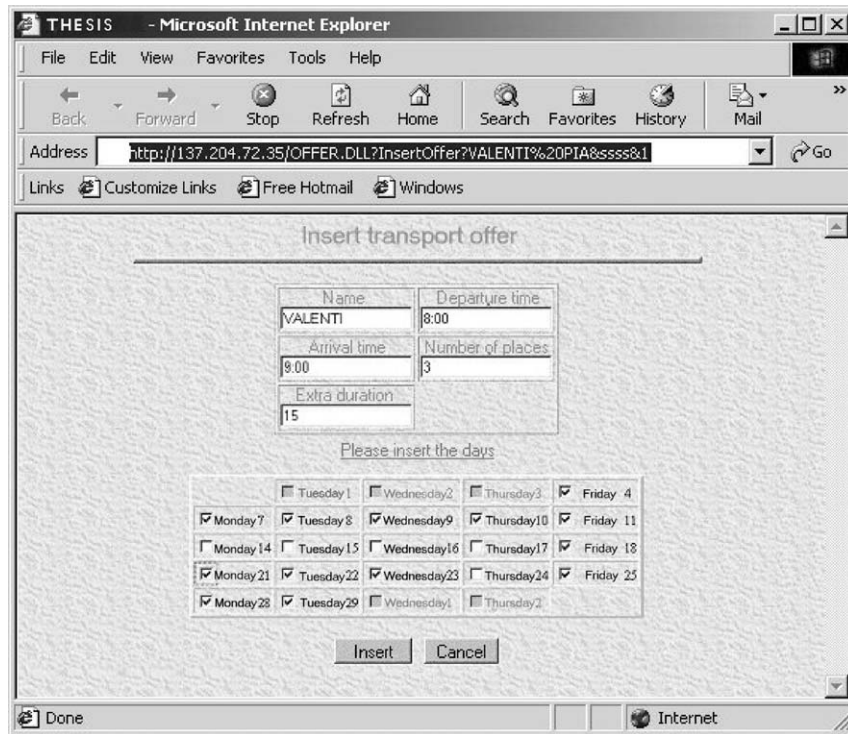


Fig. 1. The web interface to insert the data.

GUI, is allowed to insert transportation requests and offers; the system then generates web pages containing text and maps presenting the results of the optimization algorithm (see Fig. 4). Both email and SMS can also be used for submitting requests and for receiving results or variations of previously proposed results (following real time notifications). The clients know whether their requests could be matched and the detail of their pick-up, while the servers simply receive an SMS notifying the pool formation: travel details are separately sent via email. SMS are also used by the servers (via a service relay) to inform clients of delays.

Among the three means to access and edit the database supported by the system, the easiest to use is the web interface, which is unfortunately also the least accessible since it needs a connected computer. Email can be used by wireless portable devices, and supports push technologies, while SMS are as ubiquitous as are cellular phones.

Operationally, a user sending an email to the system must insert the message string in the subject field, which will be processed by a mail receiver agent and finally sent to the parser. Economic and privacy considerations suggested to send all SMS to a SMS engine, which is interfaced with the system and which transfers the string sent to the parser. Each time the system receives a syntactically correct email or a SMS, it automatically sends an acknowledge SMS message.

While the system generates messages to be sent directly to users, those which the users generate, either email or SMS, must follow a rigidly structured format to allow automatic processing. We implemented a single parser for all these messages. The grammar used had to be tailored to the

application and is described in the following. Moreover, we worked under tight constraints, as especially SMS has to be as short as possible to be actually input by a user while being able to convey all necessary information to specify user constraints. We thus defined a different syntax for the messages generated by clients or servers and sent to the system, and for the response system messages.

Messages generated by the clients:

```
<mess_request>::=R-<client_user_code>-<client_password>-
    <pool_service_date>-<time_departure>-<time_arrival>

<mess_modify_request>::=MR-<client_user_code>-<client_password>-
    <pool_service_date>-<time_departure>-<time_arrival>

<mess_delete_request>::=DR-<client_user_code>-<client_password>-
    <pool_service_date>
```

A client is thus enabled to request a service (header R), to modify a previous request (header MR) or to cancel a previous request (header DR). In each case, all necessary parameters are encoded and sent together.

Messages generated by the servers:

```
<mess_offer>::=
    F-<server_user_code>-<password_server>-
    <pool_service_date>-<time_departure>-<time_arrival>-
    P<num_seats>-E<max_delay>

<mess_modify_offer>::=MF-<server_user_code>-<password_server>-
    <pool_service_date>-<time_departure>-<time_arrival>-
    P<num_seats>-E<max_delay>

<mess_delete_offer>::=DF-<server_user_code>-<password_server>-
    <pool_service_date>

<mess_delay>::=RIT-<server_user_code>-<password_server>-
    <pool_service_date>-MIN:<delay>
```

A server is thus enabled to offer a service (header F), to modify a previous offer (header MF), to cancel a previous offer (header DF) and to announce a real-time delay (header RIT).

Messages originated by the system and sent to the users:

```
<mess_acknowledge_generic>::=ACK-<pool_service_date>
<mess_pool_for_client>::=AP-DATE:<pool_service_date>-TIME:<pickup_time>
<mess_pool_cancelled_to_client>::=CANCELLED-DATE:<pool_service_date>
<mess_delay_to_client>::=ACTRIT-DATE:<pool_service_date>-MIN:<delay>
<mess_pool_for_server>::=AP-DATE:<pool_service_date>-TIME:<pickup_time>-
    CHKEMAIL4DETAILS
<mess_client_cancellation_to_server>::=NOCLIENT-DATE:<pool_service_date>-
    USER:<client_user_code>
```

```
<user_code>, <password>, <time> ::= 4 digits strings  
<date> ::= 6 digits string, format ddmmyy  
<delay> ::= 2 digits string
```

The system sends an acknowledgment to each received message (header ACK), communicates the pools once computed both to clients and to servers (header AP-DATE, different parameters), forwards client and server cancellations (headers NOCLIENT-DATE and CANCELLED-DATE, respectively) and server delays (header ACTRIT-DATE).

The web interface is obviously more user-friendly. Each employee can specify through an ASP page the set of days (possibly empty) when he/she is willing to drive and the set of days when he/she asks to be picked up (see Fig. 1).

The system then computes a matching between servers as clients. The result is a set of routes starting from the server houses, arriving at the workplace and passing through the set of client houses without violating the time windows constraints and the car capacity constraints.

These routes are made available both to clients and to servers. A well known GIS module, ArcView, loads the route information given in alphanumeric format, after the optimisation. Then, it transforms these informations in a set of GIS Views through a set of queries to its database. The last step transforms the views into bitmaps displayed by the web server (see Fig. 3). Moreover, the system alerts clients and servers in real time when any variation of the scheduling happens, by means of SMS.

3.2. System architecture

The system outlined in the previous sections was implemented at the Joint Research Center of the European Commission using standard commercial software and developing extra modules in C++ using Microsoft Visual Studio C++ compiler.

The database used is Microsoft Access and it is accessed through the Internet, in such a way that the user does not need to be aware of the location of the database, it is sufficient that the user is able to consult, add, modify the data as needed.

The access mode to the remote databases makes use of an ISAPI (Internet Server Application Program Interface from Microsoft) interface, which uses the gateway based on Open Data Base Connectivity ODBC and Data Access Objects DAO. ISAPI was developed to solve some typical issues related to the Common Gateway Interface (CGI), and has access to the Dynamic Link Library on the PC platforms, which requires less resources on the server side than the traditional CGI interface. Indeed, the CGI creates a new process for each request made by the users. This operation takes time and memory (to activate a new process and to allocate system resource). The system efficiency decreases rapidly as the queries increase. The ISAPI have the advantage of creating a new thread for each request internal to the process in execution. This operation is faster and uses less memory. The geographical data are stored in shape files and used by an ArcView application (release 3.2). The system administrator can directly access data through MSAccess and ArcView, which are both packages available and running in the same machine where the car pooling application resides. Through Arcview the system manager can start the optimization module.

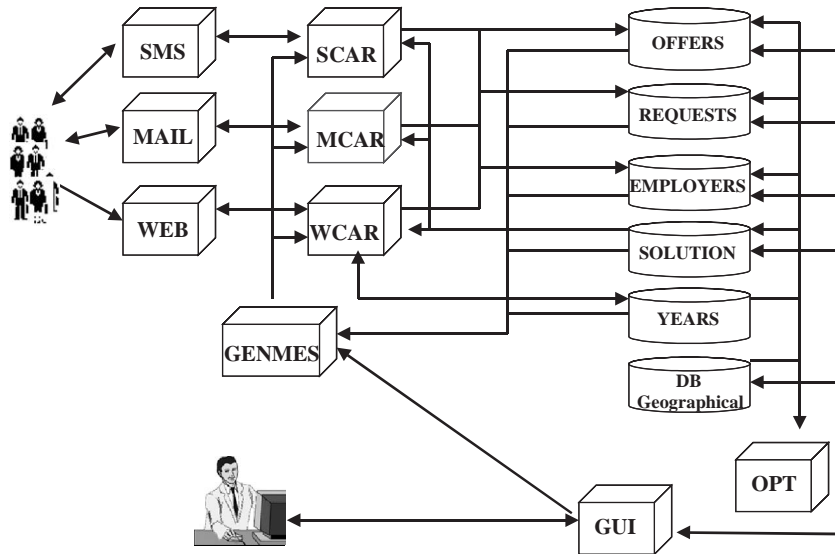


Fig. 2. The architecture of the system.

The architecture of the system developed for this problem is shown in Fig. 2. The system consists of six modules:

- **OPT**: generates a feasible solution using an algorithm which defines the paths for the servers. The algorithm makes use of a heuristic approach and is used to assign the clients to the servers and to define the path for each server. Each path minimizes the travel time, maximizes the number of requests picked up and satisfies the time and capacity constraints.
- **(X)CAR**: this is actually a set of three modules which permit to receive, decrypt and send SMS (SCAR), EMAIL (MCAR) and web pages (WCAR) to the users, respectively. These three modules have a complex internal structure, and they implement different procedures and functionalities. The module *SMS Car Pooling* (SCAR) allows the server to send and receive SMS messages. The module *Mail Car Pooling* (MCAR) supports email communication and uses POP3 and SMTP as protocols. The module (WCAR) is the gateway for the web interaction. All modules filter the customers access, allowing the entitled users to insert new data, to query and to modify the database (for example the departure and the arrival time desired) and to access service data.
- **GENMES**: generates the messages which enable the interaction between the user and the system through the modules (X)CAR for communication with the user.
- **GUI**: a graphical user interface based on ESRI ArcView. It generates a view (a digital map) of the current problem instance and provides all relevant data management. The module GUI connects the module OPT with the module GENMES.

4. The OPT module

The *OPT* module solves the problem of matching clients and servers and defining the paths to be driven by the servers in going to work (and the opposite problem of returning from work). The

solution is actually obtained solving a modified vehicle routing problem with pickup and deliveries time windows (VRPPDTW).

In fact, the DCPD where all servers have identical cars of capacity Q is a special case of the *Dial-a-Ride Problem* (DARP) obtaining the formulation which can be find in [3], which asks to design the routes of m identical vehicles, each of capacity Q , starting from the same depot and servicing n transportation requests. Each request corresponds to a person who is to be picked up at a starting point and dropped off at a destination. As in the case of the DCPD, the objective is to design the routes of the vehicles in order to satisfy as many requests as possible, minimizing the sum of the route costs and the penalties incurred for unserved requests.

So far, no exact algorithms for the DARP with more than one vehicle has been published in the literature, while several heuristic techniques appeared (for a survey, see [11]). DARP itself is a generalization of a number of vehicle routing problems, such as the Vehicle Routing Problem with Pickup and Delivery and Time Windows (VRPPDTW) and the Vehicle Routing Problem with Time Windows (VRPTW). For surveys of methods for these problems, see [12,13].

In fact, the car pooling described can be formalized mathematically as a mixed integer programming problem obtaining the formulation of the VRPPDTW, a problem from which it thus inherits its NP-hard status. The full formulation of DCPD can be found in [3].

The objective function minimizes the sum of two terms: total travel time and a penalty incurred for unserved requests.

The operational constraints are the following. First, it is ensured that each customer is serviced by one vehicle, at most. Then, the number of vehicles travelling to the destination is bounded by the number of offers. Capacity constraints ensuring the feasibility of the loads and a maximum travel time are imposed to each path. A final set of constraints imposes that the departure time from each client's house takes place during the client's acceptable departure time window. Similarly, the arrival time at destination is limited by the shortest among the crews time limits.

4.1. The optimization algorithm

The core of the *OPT* module is the algorithm for solving problem DCPD. As mentioned, DCPD, a problem which can be demonstrated to be an extension of the VRP with unit costs, is NP-hard [3]. Moreover, the size of the instances that needs to be solved exceeds those that can be solved to optimality, thus a heuristic approach for solving DCPD was designed. The module actually implements two algorithms: the first one is a preliminary procedure which defines a graph, the day problem graph (DP graph), which represents the distances between each pair of customers and between each customer and the workplace. Distances are computed as actual kilometer distances on the road network, while the time needed to travel the paths are computed on-line. This is because typical road congestion hours need to be considered, thus the time to drive through an arc depends on when the arc is used. As mentioned, the expansion phase of the Dijkstra code was modified, so that it works with different time matrices, one for each time period corresponding to different congestion conditions.

Necessary data for defining the DP graph are the complete and detailed road network of the region of interest, the sets of clients and servers for that particular day and their geocoding on the road network. The DP graph is then a graph defined as follows. The set of nodes contains a node associated to each client, a node associated to each server and a node associated to the workplace.

The set of arcs contains, for each ordered pair of nodes i, j , an arc (i, j) between them only if it is time-feasible for at least one server to successively visit node i , node j and then the workplace (adaptation of the idea to specific cases where a node corresponds to a server or to the workplace being obvious).

Having computed the DP graph it is possible to construct a heuristic solution for problem DCP. The procedure is a two-phase one, with a construction and a local optimization phase.

4.1.1. The construction heuristic

The algorithm starts with no employee routed and an idle fleet of vehicles. A number of new routes equal to the number of servers is initialized as direct paths from each server to the workplace, then, as long as possible, single clients are inserted into existing routes. It is therefore necessary to determine *which* node to insert and *where* to insert it, which route and which position. As there are many ways to do this, a wide variety of constructive methods can be derived from the literature. Given the problem complexity and the tight CPU time operational constraints, we were interested in greedy algorithms, which expand an incumbent partial solution.

We adapted the construction approach originally proposed by Mulvey and Beck [14] for the capacitated p-median problem. The construction is based on the computation of a *regret* for each client $i \in C$, which are computed as follows. For each client $i \in C$ and server $j \in S$ the *extramileage* $e(ij)$ of the detour of server j for picking up i is computed, that is the difference of the distances for driving directly from j to the workplace and for driving from j to i and then to the workplace. The regret of client i is defined to be the difference between the least and second least extramileages thus computed with reference to client i : $e(ij_2) - e(ij_1)$ where j_1 and j_2 are the servers which have the least and second least extramileage for picking up i , respectively. Regrets are fast approximation of dual assignment information, roughly estimating marginal assignment costs.

The construction algorithm tries to assign each client to its closest server, considering clients in order of decreasing regrets. Thus, when a client has a single feasible server available (infinite regret), it is assigned to it. Clients with the largest regret values are considered first, in order to avoid the large penalty associated with assigning a client to a server which must make a longer detour in order to pick him up. Assignments are conditioned by capacity and time windows constraints, forbidding the assignment of clients to servers which have filled their car capacity or which would violate some time window if picking up the client. The procedure stops either when all the requests are serviced or when it is impossible to assign any unserved client to any server.

4.1.2. Local optimization

Starting from the initial solution produced by the procedure described above, better ones are obtained by means of a local search algorithm. Local search is based on the concept of *neighborhood*, which is the set of all solutions achievable from the current one applying a given family of transformations. A local search procedure looks for an improved solution in the neighborhood of the incumbent solution and if one exists, it replaces the former one; otherwise, the procedure stops.

It is possible to design neighborhoods in many different ways. OPT extracts clients from the paths defined in the construction phase and tries to reinsert them in other paths. Specifically, one random

client is extracted from each path, extramileages are computed for each path and for each unserved client and clients are reinserted as in Section 4.1.1. This tries to overcome local minima derived by the ordering used in the construction phase.

The local search so defined terminates after a predetermined number of extraction–insertion iterations.

4.1.3. Pseudocode

A high level pseudocode of the two phases algorithm follows.

Algorithm HDCPP

Phase 1;

Set $U = \emptyset$ // Set of unserved clients

For each client $i \in \mathcal{C}$

For each server $j \in \mathcal{S}$

 Compute the extramileage $e(ij)$

 Sort the servers by increasing extramileage values for $i: \{j_1, j_2, \dots, j_m\}$

 {Calculate the regret value for client i }

$r_i := e(ij_2) - e(ij_1)$;

Sort the clients by decreasing regrets r_i

For each client $i \in \mathcal{C}$

 Assign it to its least-cost feasible server.

If no server is feasible, **set** $U = U \cup \{i\}$

Phase 2;

While not(*terminating_condition*) **do**

For each server j

 Choose randomly one client i among those associated with server j .

 Remove i from the pool of server j and add i to \mathcal{U}

For each client $i \in \mathcal{U}$

For each server $j \in \mathcal{S}$

Compute the extramileage $e(ij)$ // This step considers the whole remaining path of j

 Sort the servers by increasing extramileage values for $i: \{j_1, j_2, \dots, j_m\}$

$r_i := e(ij_2) - e(ij_1)$; Calculate the regret value for i

 Sort the clients in \mathcal{U} by decreasing regrets r_i

For each client $i \in \mathcal{U}$

 Assign it to its least-cost feasible server, if one exists.

Save the best solution found

End

As mentioned, the terminating condition is set on the number of iterations of the while loop.

The algorithm presented uses a neighborhood based on extramileage, in the local search phase. A different neighborhood has also been used, based on savings. Inside the while loop, the two neighborhoods so obtained were alternated in order to improve the quality of the local minima.

Table 1
Results with client/server distribution 1

Employees	Servers	Clients	CPU (s)	T.Cost	Unservd	T.Time
50	13	37	0	535	14	254
100	25	75	2	1323	16	990
200	50	150	10	1748	22	1386
300	75	225	31	3647	60	2465
400	100	300	57	5106	91	3045
500	125	375	108	6448	105	4222
600	150	450	177	6948	150	3928

5. The real-life case study

The Joint Research Center of the European Commission (JRC) is situated in the northwest of Italy not far from Milan. It covers a big area of 2 km² and its mission is to carry out research useful to the European Commission. The number of employees is of about 2000 people, divided into three main classes: administrative, staff and researchers. They come from all around Europe and they live in the area surrounding the center. The wide geographical area covered by the commuters is of about 100 km². Since this is a sparsely populated area the public transport (i.e. trains, busses, etc.) is definitely insufficient, thus private cars are necessary and used. The European Commission has implemented a private bus service which has some limits. From the employees point of view the time table is rigid and the service covers only a small part of the relevant area. On the other side this service is quite expensive for the commission, and the busses are always underloaded.

5.1. Computational issues

The heuristic HDCPP was tested on a set of real-world problem instances, derived from data provided by the JRC. The instances used are the same as those described in [3]: they are derived from the real-world instance defined over 600 employees by randomly selecting the desired number of clients and servers (see Fig. 4). For each problem dimension two different instances were generated, varying the number of servers and clients. The total number of used instances is 14. The reported computational results were obtained on a Pentium II machine, 266 MHz with 64 Mb of RAM.

Tables 1 and 2 show the following columns:

- *Employees*: number of employees of the instance
- *Servers*: number of servers
- *Clients*: number of clients
- *CPU (s)*: CPU time, in seconds, used by HDCPP
- *T.Cost*: best solution cost
- *Unservd*: number of unserved clients
- *T.Time*: total driving time of the best solution.

The instances reported in Table 1 were defined choosing 25% of employees to act as servers and the remaining as clients; in the second set of instances, Table 2, the number of servers is

Table 2
Results with client/server distribution 2

Employees	Servers	Clients	CPU (s)	T.Cost	Unserved	T.Time
50	17	33	0	545	8	406
100	33	67	1	943	11	681
200	66	134	4	1937	6	1756
300	99	201	10	2894	14	2660
400	132	268	25	4343	30	3057
500	165	335	65	5935	57	4610
600	198	402	46	6809	76	4884

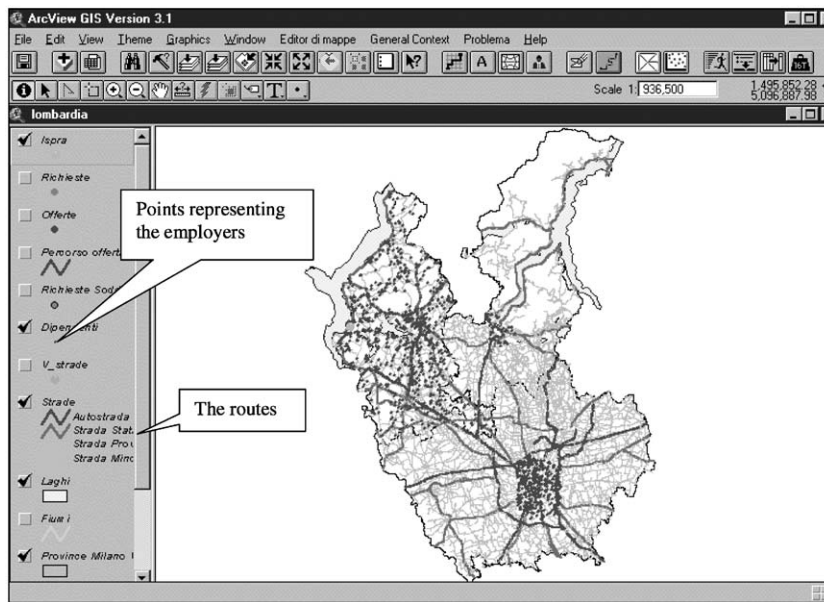


Fig. 3. The set of requests and offers generated.

33% of the total number of employees. A different percentage of servers reflects the impact of different incentive policies that could be implemented by the center. The solution cost is obtained as the total travel time (the sum of the travel times for each vehicle) added to a penalty for each unserved client. The travel time is computed in seconds and the penalty has been set equal to 1000. Analyzing the two tables shows that the CPU time used by HDCPP increases approximately linearly with the problem dimension and the number of unserved requests is a constant proportion of the total number of employees. Moreover, comparing two instances with the same dimension, but with different percentage of servers, one can see that the CPU time increases when fewer servers are available, as expected, while the total travel time decreases, since fewer paths are driven.

The solution and the CPU time necessary to obtain it is also related to the spatial distribution of clients and servers. For example, in Table 1 the total number of routes in the case of 500 employees

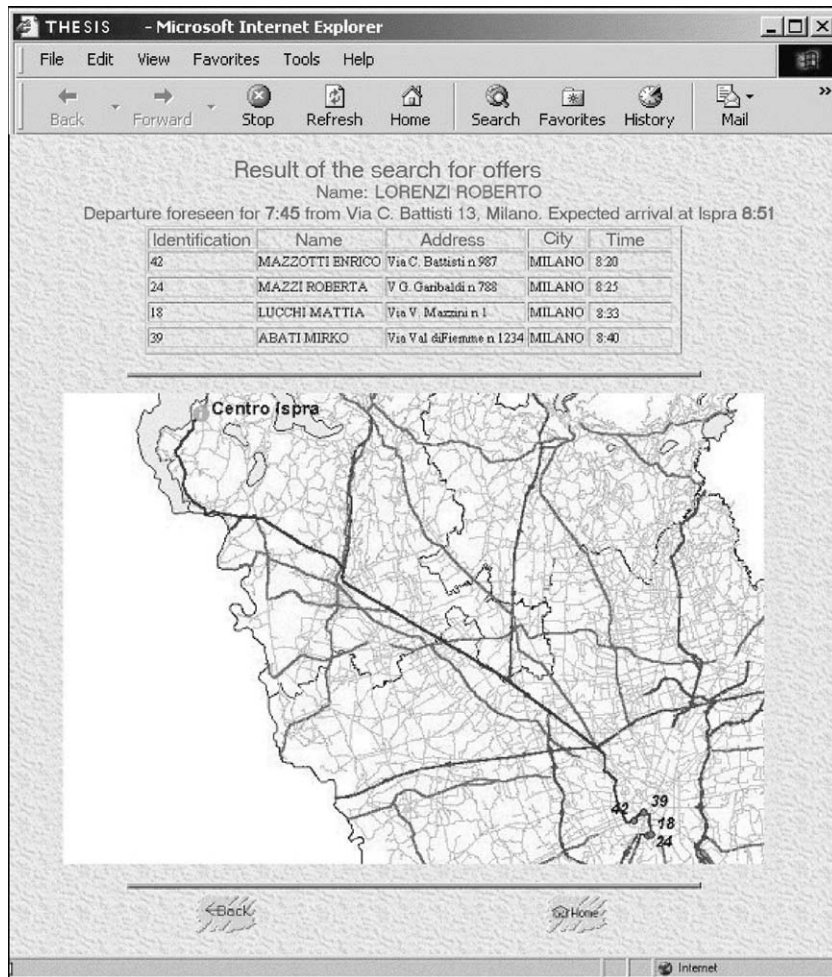


Fig. 4. The path of a single vehicle (server) generated with arcview and displayed through the web.

is greater than the case with 600: this comes from a rarer employee distribution. As a general comment, the trade-off between solution quality and CPU time permits to solve effectively instances up to dimension 400.

6. Discussion and conclusions

A system to manage a car pooling service has been presented, which main feature is the integrated use of state-of-the-art IC Technologies: GIS, Web e SMS and heuristic optimization. The optimization algorithm to build crews and an effective customer alerting system are the key characteristics making this service appealing to the users. Specifically, the optimization module solves large instances of a NP-hard problem related to the Dial-a-Ride problem. Particular attention was devoted

to enabling the optimization algorithm to work on congested road networks, inducing time-varying travel times. Moreover, the information technologies used facilitates updating the database through remote connections for any user, while the GIS is used to build the web pages which show the optimal path. People using the service can verify how far is what they get from what they want and advise the system immediately if anything changes.

The architecture of this system was implemented and presented to a real-life case study to prove its effectiveness. The real-life case study chosen regards the Joint Research Center of European Commission. In this context, it was shown how the system works providing possible advantages for the society, people involved and the organization promoting it. The system developed was tested using models of real-life data, checked and accepted by prospect system administrators. The data of the testing were undertaken by the authors and their collaborators, testing first each part of the system and finally the system as a whole. The data inserted for the test runs were based on the knowledge of the distribution of houses of employees around the JRC site, and preferred routes in the area and typical travel patterns were judged to be reasonably realistic.

This article shows that the technologies used are mature, and that the barriers to practical use are disappearing. The pilot implementation demonstrated that minor issues still exist related to the reliability and interaction between the various technical components, while proving that these obstacles are surmountable. The investigations have also shown that car pooling systems are easier to devise and implement for a situation where one arrival/departure point is common for all users. Finally this study highlighted that the reliability of the overall system actually needs to be very high, in order for the users to be sure that their transports arrive as predicted and that they are informed immediately in case of delay or changes, which is rapidly becoming possible thanks to technological advances.

References

- [1] Colomi A, Cordone R, Laniado E, Wolfler Calvo R. L'innovazione nei trasporti: pianificazione e gestione. In: Pallottino S, Sciomachen A. (Eds.), *Scienze delle decisioni per i Trasporti*, FrancoAngeli, 1999.
- [2] Baldacci R, Maniezzo V, Mingozzi A. An exact algorithm for the car pooling problem. *Proceedings of CASPT-2000, 8th International Conference "Computer-Aided Scheduling of Public Transport"*, 2000.
- [3] Mingozzi A, Baldacci R, Maniezzo V. Lagrangean column generation for the car pooling problem. Technical Report WP-CO0002, University of Bologna, S.I., Cesena, Italy, 2000.
- [4] Hildmann H. An ants metaheuristic to solve car pooling problems. Master's thesis, University of Amsterdam, Faculty of Science, Department of Artificial Intelligence, 2001.
- [5] Maniezzo V, Carbonaro A, Hildmann H. An ants heuristic for the long-term car pooling problem. In: Onwubolu G, Babu BV (Eds.), *New Optimization Techniques in Engineering*, 2003, to appear.
- [6] Telematics for car pooling, 1999, <http://195.65.25.24>.
- [7] The New South Wales Government, 1999, <http://www.rta.nsw.gov.au/index.htm>.
- [8] Land of Sky, 1999, http://www.landofsky.org/planning/lu-trans/transp_opts.htm.
- [9] Valley Metro, 2000, <http://www.valleymetro.maricopa.gov/carpool.html>.
- [10] Dailey DJ, Loseff D, Meyers D. Seattle smart traveler: dynamic ridematching on the world wide web. *Transportation Research Part C* 1999;7:17–32.
- [11] Cordeau J-F, Laporte G. The dial-a-ride problem: variants, modelling issues and algorithms, *4OR-Quarterly journal of the Belgian, French and Italian Operations Research Societies*, Forthcoming.

- [12] Cordeau J-F, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. Vrp with pickup and delivery. In: Toth P, Vigo D (Eds.), *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia: SIAM, 2002. p. 225–42.
- [13] Cordeau J-F, Desaulniers G, Desrosiers J, Solomon MM, Soumis F. Vrp with time windows. In: Toth P, Vigo D (Eds.), *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications. Philadelphia: SIAM, 2002. p. 157–93.
- [14] Mulvey JM, Beck MP, Solving Capacitated Clustering Problems. *European Journal of Operational Research* 1984;18:339–48.