RutaRep: A Computer Package to Design Dispatching Routes in the Meat Industry

J.M. Belenguer, E. Benavent and M.C. Martínez

Universitat de València Estudi General*, Spain

In this paper we present a computer program that has been developed to design the dispatching routes of a medium-sized meat company in Spain. We have modelled the real problem as a variant of the Vehicle Routing Problem with Time Windows and implemented a number of heuristic algorithms based on the most advanced solution techniques for this problem. These algorithms have been embedded in a computer package that is intended to be used as a decision support system for the distribution manager. The program runs under Windows System and is straightforward to use. We also present some computational experience based on real instances provided by the company. This experience shows important improvements, in both global distance and customer service.

Keywords: food distribution; heuristics; tabu search; routing.

Introduction

It is accepted that distribution problems are currently one of the most important problems that both, public and private companies, have to face day to day. This is especially true for companies involved in the delivery of goods, whose income is strongly dependent on how distribution problems are solved. Therefore, it is crucial that computer systems able to perform such a task as efficiently as possible be developed.

In scientific literature these problems are known as Vehicle Routing Problems. During the last twenty years many papers have been devoted to research on optimisation and

* *Correspondence: Dr J.M. Belenguer, Departament d'Estadística i Investigació Operativa, Facultat de Matemátiques, Doctor Moliner, 50, 46100 - Burjassot, Valencia, Spain, e-mail: Jose.Belenguer@uv.es*

approximation algorithms for these problems. Nevertheless, most of the algorithms we find in this literature are not directly applicable to real routing problems which usually incorporate many side constraints.

Real vehicle routing problems may be solved, if not optimally, at least heuristically, using a commercial vehicle routing package. The rapid development of computer technology and the introduction of Geographic Information Systems (GIS) has led to the marketing of lots of distribution packages. However, for small and medium sized companies these packages are usually too expensive and may not exactly meet the characteristics of the company's vehicle routing problem. Furthermore, the more complete these packages are, the longer the time that will be required for their adaptation to the company and for staff training.

In this paper we report the development of a computerised system for distribution planning at Carnes Estellés S.A., a medium sized Spanish meat company. The company's customer-portfolio is currently partitioned into geographical zones, and each zone is assigned to a given vehicle. Daily, the distribution manager organises the orders by zones, and the drivers decide the sequence in which the customers are visited. Moreover, each customer wants to be serviced within a specific period of time and, theoretically, such a period must be respected; however, for majority of the customers the periods are usually very similar, so lateness in service are inevitable.

During recent years the company's volume of business has increased substantially. This has lead the managers to realise the necessity for change in the way in which the routes to deliver their fresh and elaborated meat products are being organised. As a consequence, the company asked us to develop a computer system able to design the routes as automatically as possible. The system should try to improve the delivery service by decreasing the average lateness and the operating costs of the vehicles, wherever possible. The program should also be used to answer several strategic questions, such as whether or not to maintain the current system of delivering by zone, to optimise the size and composition of the vehicle fleet, etc.

Here, we describe the development of our routing system which has been specifically designed to efficiently satisfy the company's needs, being also cheaper and simpler to use than other commercial packages.

The paper has been organised as follows. Firstly, we present the distribution problem faced by the company and its relation with other known routing problems. Secondly, we describe the capabilities of our package and the user interface (designed for use by the distribution manager). Following this, we describe the internal functioning of the program, placing special emphasis on the optimisation algorithms we have implemented to build the delivery routes, and on the handling of the geographical data that are necessary for the algorithms. Finally, we compare the results obtained by our program with the routes used by the company based on a set of real data provided by the company.

The Distribution Problem at Carnes Estellés S. A.

At present, Carnes Estellés S.A. has a customer-portfolio of approximately five thousand customers. The activity of the company mainly consists of slaughtering different kinds of animals, principally pigs and calves, jointing and delivering them to several kinds of customers: restaurants, butcher's, meat markets, hypermarkets and supermarkets. To a lesser extent, the factory also elaborates different products derived from the meat: sausages, pâté, etc.

The customers are partitioned into 23 geographical zones, and each zone is assigned to a given vehicle. Every day, except Sunday, a number of the customers give an order for some quantity of different kinds of meat and other products. To deliver the day's orders, the company has signed a contract with a number of drivers, each owning a vehicle. The money paid to each driver depends on the type of vehicle and on its delivery zone, but neither on the distance travelled nor on the elapsed time in the delivering process. All the vehicles must be loaded at the factory (which will be also called the depot), deliver the customers' orders assigned to their zones, and return to the factory.

Routes are classified into long and short distance routes. At the moment, long distance routes are quite stable and the company is not, at present, interested in changing them. In fact, their corresponding geographical zones are far away from the depot (for example, one of them consists of the Balearic Islands) and usually require a special vehicle. Therefore, our problem is only concerned with the short distance routes that basically cover the Valencian Community. This area includes, at its centre, the city of Valencia, which is 10 km away from the depot, and an area of about 23300 Km^2 along the Mediterranean coast divided, overall, into 11 delivery zones. The daily average number of orders in each zone is about 20.

Constraints

The total demand (kilos of fresh meat and/or elaborated products) delivered by a vehicle must be less than or equal to the vehicle's capacity. The capacities of the vehicles range from 1000 to 3500 kilos. All the vehicles are loaded at the depot, leave the depot at a known departure time and must return before 16:00 h.

Each customer should be serviced at a time interval (time window) $[a,b]$, where a is the earliest time at which the order can be delivered, and b is the latest time at which the customer desires to be serviced. In other words, if a vehicle arrives at a customer's location before the earliest time, it will have to wait, however, the latest time may, if necessary, not be respected. When an order is delivered after the latest time, we will say that a lateness has occurred. This kind of time window is generally called a soft time window. Time windows are very similar in large groups of customers, for example butchers want to have the meat delivered as soon as possible in the morning. For this reason, it is often impossible to meet all the time windows.

As we have mentioned above, each vehicle currently has a delivery zone assigned and each day it services all the orders in its zone. This is convenient for the drivers because they probably have the best knowledge of their own zone, and also for the distribution manager because, in this way, all the orders are automatically assigned to the drivers. Nevertheless, it might happen that, on a particular day, there were great differences among the workloads of each zone. Then, the possibility of assigning the orders of one zone to the vehicle of another zone would allow the balancing of the vehicles' workload thus achieving better routes in terms of distance and customer service. In order to take advantage of this, while avoiding routes containing a lot of orders outside of the vehicle's zone, the route of each vehicle should contain at least a given percentage of the orders within its regular zone. This percentage will be set, each day, by the distribution manager.

Objectives

Each working day, the program must provide the distribution manager with a set of routes covering all the orders, using only the available vehicles and satisfying all the problem constraints. The solution must be generated within a few minutes and the program should allow the manager to make small changes to the routes in an easy way. Furthermore, after viewing the routes generated by the program, the manager should be able to change data relating to some of the orders and reoptimise the routes.

Since the vehicles belong to the drivers, each year the company signs a contract with each driver which stipulates the price of the distribution task, depending on the zone they have to service and on the vehicle's characteristics. For this reason, the company is not especially interested in minimising the total distance travelled and its main objective is to improve the quality of the delivery service to the customers by minimising the lateness. Nevertheless, the distance travelled by a vehicle is, of course, of primary concern to the driver, who has to pay the operating costs of the vehicle. Thus, the routes should also be of minimum distance, to avoid problems with the drivers, who would otherwise want to renegotiate their contracts. Given that minimising the lateness of service and the distance travelled are competing objectives, the program should be able to generate different solutions, leaving the final decision of choosing which will be used to the distribution manager.

The company managers are also very interested in studying the distribution system carried out at present by the company, and in evaluating the impact of some strategic changes that could be made, such as changing the number and composition of the vehicle fleet and making changes to the current distribution zones, etc.

The computer package: ***RutaRep***

In order to solve the problem just described, we have developed the program *RutaRep* that has been coded in Visual C++ and runs on a personal computer under a Windows 95/98 system. It uses the most normal resources of the Windows environment to facilitate the interface with the user: mouse actions, pull-down menus, message boxes, grids, etc.

Once the program RutaRep is opened, the data files containing the neccesary information about the orders and the vehicles should be read. Then, the program offers the following options:

- editing of the data of the orders and/or vehicles,
- building of the routes,
- showing and modifying the routes generated by the program
- printing the routes.

Reading the data

The program uses three input data files: the *customers* file, the *vehicles* file and the *orders* file. The *customers* file contains details regarding all the customers of the company that place orders regularly. This file includes, for each customer: a code number, the delivery zone, the customer name, postal address and delivery time window. This file is only read by the program when any change is produced in the customer-portfolio. The distances between any pair of customers and between the depot and any customer are computed automatically by the program using only their postal addresses; no further input data, such as customer co-ordinates or inter-customer distances, are necessary. This is a great advantage because the information required to the company is just that which any company would usually have regarding its customers. The way in which distances and travel times are computed by the program is explained below.

The *vehicles* file includes, for each vehicle: its delivery zone, its capacity and its estimated departure time. The *orders* file contains the orders made by customers on a given day, and includes, for each order, its total weight and the code of the customer making the order.

Editing of the data

The distribution manager may face different unexpected situations during the process of route design: a vehicle being out of order, the orders of a given zone exceeding the capacity of the corresponding vehicle, the stock of some product not being enough to satisfy the demand, etc. Thus, it must be possible to take into account changes to the data of one or more of the orders and/or vehicles. The program allows the user to modify the capacity and/or the departure time of the vehicles, along with the weight, the zone and the time window of the orders. It is also possible to eliminate one or more of the orders.

Building of the routes

When the user chooses to build the routes, the program prompts a window (see Figure 1) that allows the control of the objective to be prioritised while building the routes:

- minimisation of the total distance travelled by the vehicles,
- minimisation of the total lateness incurred when servicing the orders, or
- balancing of the total distance and total lateness.

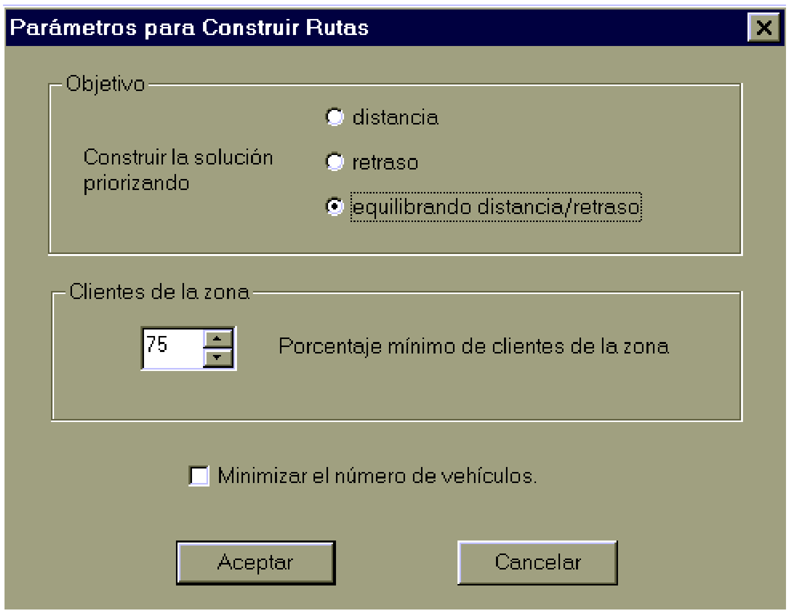

Figure 1

This window is also used to set the minimum percentage of orders within any delivery zone that must be serviced by the vehicle assigned to that zone. This percentage is respected by the program while building the routes, thus avoiding the generation of routes that are very different to the ones usually covered by the drivers.

Finally, the user can check a box that makes the program look for a solution using the minimum number of vehicles.

Obviously, the user may want to see different solutions generated by the program with different options before deciding the routes that will finally be followed by the drivers. In order to facilitate their comparison, the main window displays, at every moment, a summary report of all the solutions generated by the program up to that moment, including, for each one, the distance travelled, the lateness and the number of orders for every route in the solution, as well as the options used to generate it.

Displaying/modifying the routes

The program allows the displaying of each route in detail, showing the sequence in which the orders should be serviced, and including for each customer: the customer name, address, time window, estimated arrival time, and the lateness or waiting time if any. This information can also be printed as a way-bill for the drivers or saved as a text file.

An important feature is that the user can interact with the program modifying some generated routes to meet certain restrictions and to face situations that cannot be taken into account by the program. This allows the user to take advantage of his experience and knowledge of the distribution task.

The program also allows the user to specify a complete solution that can be evaluated by the program. The user should facilitate, for each zone, a list of customer codes in the order in which the customers have to be serviced. From this information the program calculates the arrival times at each customer, the total lateness and the total distance travelled, etc., so the user solution can be easily compared with the ones provided by the program.

It is also possible to manually eliminate a set of vehicles and/or to change their capacities and try to build a solution using the remaining vehicles, or just to entrust to the program to find the minimum number of vehicles required for the building of a reasonable set of routes. All these possibilities can be used individually or in combination, thus providing a powerful simulation tool that allows the managers to evaluate important strategic decisions such as the size of the fleet and the number and design of the delivery zones, etc.

Computing travelling distances and times

Geographical Data Handling

At present, in Spain, and specifically in the area where the customer-portfolio of Carnes Estellés, S.A. is located, it is very difficult to use a *Geographic Information System* (GIS) whose data fit well to the characteristics of the problem we have to deal with. Even when such digital geographical information exists, it is very expensive and it is not completely certain that it would fully satisfy our needs: for instance, some medium-sized towns near the city of Valencia are not coded, the traffic directions of the streets are usually not available, etc. After making a cost/benefit analysis, we decided to use only geographical data which is freely supplied by public institutions, and to build ourselves a code that computes the geographical co-ordinates and the distances among the customers from their respective postal addresses. The geographical data obtained by our code is not as accurate as that which could be obtained from a commercial GIS, but we have found that our approximation is good enough for the problem we are dealing with. Let us briefly describe our method.

We had the following geographical information at our disposal:

1. A list of the towns and villages of the Valencian Community, including their geographical co-ordinates. In this list, the city of Valencia has 8 different geographical points associated with it, corresponding to different roads out of the city. This information was provided by COPUT, an Official Community Office.
2. A matrix with the shortest road distances between any pair of points (towns or villages) included in the previous list (also provided by COPUT).
3. A street directory associated with a street map of Valencia where the city is divided into 18 x 35 rectangles. The directory includes, for each street, the set of rectangles traversed by it and the number of postal addresses in the street. To this directory, we have added the co-ordinates of the centres of these rectangles, so we know, for each street, the approximate co-ordinates of its two endpoints. Therefore, we may estimate the co-ordinates of any customer from the street and number included in his/her address.

For all the towns, except for the city of Valencia, we assign the same co-ordinates to all the customers located in the same town. This is justified by their small size, and also because the number of orders from the same town in a single day is usually small.

Computing distances between customers

To compute the distance between any pair of customers we distinguish several cases depending on where the customers are located: both in Valencia city, one in Valencia and the other outside, or both outside.

- The distance between two customers in Valencia city is set equal to 1.5 times the Euclidean distance between their locations, which is computed from their respective co-ordinates.
- The distance between two customers located outside Valencia is set equal to the road distance between the towns where the customers are located. This distance is available from the above mentioned distance matrix.
- The distance between one customer from Valencia city and the other from outside is set equal to the sum of the distance inside the city from the customer located at Valencia to the appropriate road out of the city, plus the road distance from this point to the town where the other customer is located.

In some cases the computed distance between two customers may result equal to zero, either because they are located within the same rectangle in the city, or because they are located in the same town. In these cases we heuristically fix the distance equal to 0.5 Km.

Computing travel times between customers

Travel times are computed directly from distances. Obviously, travelling through urban areas will result slower than travelling on open roads. Furthermore, the average speed over a short distance is likely to be much lower than the average speed over a longer one. Thus, we have used a convex piecewise linear function that allows us to assign different speeds depending on the magnitude of the distances between customers. We have split the interval $[0, \infty]$ into 17 subintervals of non-equal ranges and assigned to each one a velocity that has been estimated from real data provided by the drivers.

Optimisation algorithms

In the context of Vehicle Routing, the problem we have to deal with is a special Vehicle Routing Problem with Soft Time Windows, see Taillard et al. (1997). The special features of our application are:

- The vehicles have different capacities and different departure times.
- Customers are assigned to zones and each zone is assigned to one vehicle. Nevertheless, a certain degree of flexibility is allowed, i.e. some orders may be serviced by a vehicle from another zone, but a minimum percentage of orders of each zone has to be serviced by the vehicle of that zone.
- Different competing objectives are addressed: minimising the distance and maximising the customer satisfaction (that is, minimising service lateness).

Our program contains several optimisation procedures to build a solution; here, a solution refers to a set of routes that meets all the constraints. The solution is built by first calling a constructive algorithm with the appropriate settings and then calling an improvement procedure. The methods are mainly based on the insertion procedures of Solomon (1987) and Potvin & Rousseau (1993), and on the tabu search heuristic proposed by Taillard et al. (1997).

The reader may find interesting surveys on Vehicle Routing Problems with Time Windows (VRPTW) in Desrochers et al. (1988), Solomon and Desrosiers (1988), and Desrosiers et al. (1995). Finally, recent papers on metaheuristics algorithms for the VRPTW are reported in Gendreau et al. (1997). Some applications of optimisation techniques on real-world vehicle routing problems with many side constraints, including time windows, can be found in Semet & Taillard (1993) and Rochat & Semet (1994).

Modelling the cost function

In order to take into account the two competing objectives, we define the cost of a solution as a weighted combination of the distance travelled by the vehicles and the lateness incurred in the service. We have considered a weight W that may take different values depending on the decision of the user as to whether to give priority to the minimisation of the total distance (W = 0.01), the total lateness (W = 1) or to balance both objectives (W = 0.1). Let D_k be the distance (in Kilometres) travelled by route k and let R_k be the total lateness (in minutes) of route k, i.e., the sum of the lateness incurred in the service of its orders. Then, the cost of a solution is:

$$\sum_k D_k + W \sum_k R_k$$

where the summations are for all the routes contained in the solution. Note that, when an objective is prioritised, the other one is still considered in the cost, although with a lesser relative weight.

We have implemented two constructive heuristics and an improvement procedure.

Constructive heuristic per zones

This constructive procedure constructs a route for each vehicle that includes all the orders within its zone. Let us first assume that all the vehicles are available so every zone has its own vehicle, we will explain below how this method has been adapted to the case where some vehicles are not available. This procedure is based on the Solomon's insertion algorithm I1.

For each zone, a route is initialised either by the order within that zone that is farthest from the depot or by the one with earliest latest time b. Then, at every iteration, a new unrouted order is inserted in the current route until all the orders of the zone have been inserted.

The order to be inserted in each iteration is determined as follows. Firstly, for each unrouted order u, we compute the "cost" of inserting u in the route between orders i and j as:

$$c_1(i,u,j) = \alpha_1 \Delta_d + \alpha_2 \Delta_t + W \Delta_r$$

where Δ_d is the increment of distance in the route, Δ_t is the increment in the arrival time at order j, and Δ_r is the increment of lateness in the whole route caused by the insertion. The values of parameters α_1 and α_2 will be specified later. Then, we compute $c_1(u)$ as the minimum of $c_1(i,u,j)$ among all the consecutive orders i and j in the route. Finally, the order u to be inserted is the one that maximises:

$$c_2(u) = \lambda d_{0u} - c_1(u)$$

among all the unrouted orders of the zone, where λ is another parameter and d_{0u} denotes the distance between u and the depot. Note that, if $\lambda > 0$, this type of insertion rule implies that the orders farthest from the depot and with lesser insertion cost will be inserted first.

Different values for the parameters $(\alpha_1, \alpha_2, \lambda)$ lead to different criteria for selecting the next order to be inserted and the best insertion place. The following values have been tried: (1,0,1), (1,0,2), (1,1,1) and (0,1,2). This, in combination with the two different ways of initialising the route, leads to eight different routes for the zone and the one with lesser cost, as defined above, is selected as the route for the corresponding zone.

This procedure can be adapted as follows to the case where some vehicles are not available. In the first step, the routes for the zones whose vehicles are available are built as in the above method, including all the orders of the corresponding zone. In the second step, the unrouted orders (belonging to the zones corresponding to the missing vehicles) are iteratively inserted in any of the existing routes with enough residual capacity. At each iteration in this second step, the order u to be inserted is the one that maximises $c'_2(u) = \lambda d_{0u} - c'_1(u)$, where $c'_1(u)$ is the minimum insertion cost of u in any of the existing routes at any position.

Constructive heuristic without taking zones into account

This algorithm can be used to estimate the minimum number of vehicles that could make the distribution task without violating the capacity constraints of the vehicles and without exceeding a given amount of total lateness in servicing all the orders. This procedure constructs the routes without taking into account the assignment of the orders to zones, so any order may be included in the route of any vehicle.

The procedure begins by building a solution using all the vehicles of the fleet, then sequentially, the lowest capacity vehicle is removed and a new solution is built if possible; the procedure stops when a feasible solution cannot be found. The method we use to try to build a solution at each step is a parallel insertion method based on Potvin & Rousseau (1993) that is described in some detail below.

First, the routes are initialised with an order, and then, the routes are developed in parallel by inserting an unrouted order iteratively in one of the routes, until all the orders have been routed or an unrouted order cannot be inserted in any route without violating the constraints, in which case the algorithm fails to provide a feasible solution.

Two strategies are tried in order to initialise the routes:

(a) the first route is initialised with the earliest latest time order, the second route is initialised with the second lowest latest time order, etc.,

(b) the first route is initialised with the order farthest from the depot, the second route is initialised with the order that maximises the minimum of the distances to the depot and to the already routed order, and so on.

Once all the routes have been initialised with one order, the algorithm continues by sequentially inserting an unrouted order in one of the partially formed routes. The unrouted order to be inserted at each iteration is selected as follows: for each unrouted order u and each route r, the minimum cost of inserting u in route r (as defined in the preceding constructive algorithm) is computed, let us denote this cost by $c_{1_r}(u)$. Let r^* be the route for which the minimum insertion cost $c_{1_r}(u)$ is reached, and

$$\hat{c}(u) = \sum_{r \neq r^*} (c_{1_r}(u) - c_{1_{r^*}}(u)).$$

The order to be inserted is the one that maximises $\hat{c}(u)$, and it is obviously inserted in the best insertion place and route. Note that $\hat{c}(u)$ sums, for each route r, the difference between the cost of inserting u in route r and that of inserting it in the best route, thus, $\hat{c}(u)$ can be seen as a measure of what can be lost later (in the following iterations) if order u is not immediately inserted in its best route.

The same values for the parameters as in the first constructive heuristic have been used. This, in combination with the two different ways of initialising the route, provides eight different solutions from which the one with lowest cost is selected as the output of the algorithm.

Improvement algorithm based on tabu search

Once a feasible solution is obtained by any of the above described constructive heuristics, we apply an improvement algorithm based on tabu search. This procedure is inspired by the work of Taillard et al. (1997) and it can be described as follows. First, we decompose the initial solution into disjoint subsets of 3 or 4 routes, then we apply a tabu search algorithm on each subset of routes and finally, the improved routes are merged to form a new solution. The decomposition is made in such a way that each subset contains routes that cover geographically adjacent areas. Given that we know the geographical co-ordinates of each order, we compute the gravity centre of each route and, using a sweep procedure centred on the depot, the domain is partitioned into sectors each one containing approximately 3 or 4 gravity centres. The whole procedure is applied three times, each time starting at a different gravity centre, thus obtaining different partitions of the set of routes.

Roughly speaking, our tabu search algorithm is an iterative procedure where at each iteration a non-tabu move (partial modification) is made on the current solution, thus producing a new current solution. In order to avoid the possibility of generating solutions already visited in early iterations of the algorithm, certain moves are declared tabu, that is, they are forbidden. The possible moves considered at each iteration are:

- Intra-route moves: a section of a route, consisting of L consecutive orders is moved to another position in the same route ($1\leq$ L ≤4).
- Inter-route moves: a section of L consecutive orders of a route is interchanged with a section of L' consecutive orders of another route ($1\leq$ L, L' ≤4).

A move is considered as feasible if the new routes that would result if it were applied to the current solution do not violate any constraint of the problem. If a move would produce a solution with the same cost as another solution already generated by the algorithm, it is declared tabu. That is, we assume that two solutions with the same cost are equal. Given that the cost includes the distance and the weighted lateness, it is, in fact, very unlikely to find two different solutions with the same cost.

All the possible intra-route moves within each route, and all possible inter-route moves between each pair of routes, are generated and evaluated and the non-tabu move producing the largest cost improvement is made, thus obtaining a new current solution. The output of the algorithm is the best solution among all those that have been generated.

In the program that was finally supplied to the company, the number of iterations of the tabu search procedure, when applied to every subset of routes, was fixed to 10. This implies that a total number of about $10K$ iterations of the tabu search are made, where K is the number of vehicles. It could be argued that this is a small number, but should to be mentioned that the number of possible moves that are analysed in each iteration is very large, and a single move may suppose a substantial change in the solution. Furthermore, there are reasons for not wasting too much time looking for very small improvements in the solution. Namely, the estimations of distances and travel times may not be very accurate. For instance, the travel times may have large variations (due to different traffic conditions, different drivers, etc.) so, a small improvement in theory may not be appreciated at all in practice.

Computational experience

In order to test the quality of the routes generated by program *RutaRep*, we have compared them with the ones that were actually followed by the drivers (referred to as the company routes) in 7 working days. The data corresponds to 7 zones from the city of Valencia and its surroundings; Table 1 shows, for each day, the number of orders and their total weight. The total capacity of the 7 vehicles involved is 12900 Kg. Note that the total weight of the orders is about 50% of the total capacity of the vehicles, nevertheless, the distribution of the orders among zones is not uniform and some vehicles are almost fully loaded on some days.

day	orders	kilos
5/17/99	94	6685.93
5/18/99	114	6040.83
5/19/99	122	6683.04
5/20/99	124	5445.73
5/21/99	148	5481.17
6/14/00	116	6068.95
7/6/00	123	6180.47

Table 1

We first show the results obtained with the balanced objective of minimising simultaneously the total distance and lateness. Two different levels of percentage satisfaction of the zones (minimum percentage of orders that have to be serviced within the vehicle's zone) were tried: 100% and 75%. Figure 2 compares the total distance of all the routes for the different solutions: company solution, and *RutaRep* solution with 100% and 75% zone satisfaction respectively. On average, the *RutaRep* solution saves 8.96% in distance with the option 100% and 14.77% with option 75% if compared with the company solution.

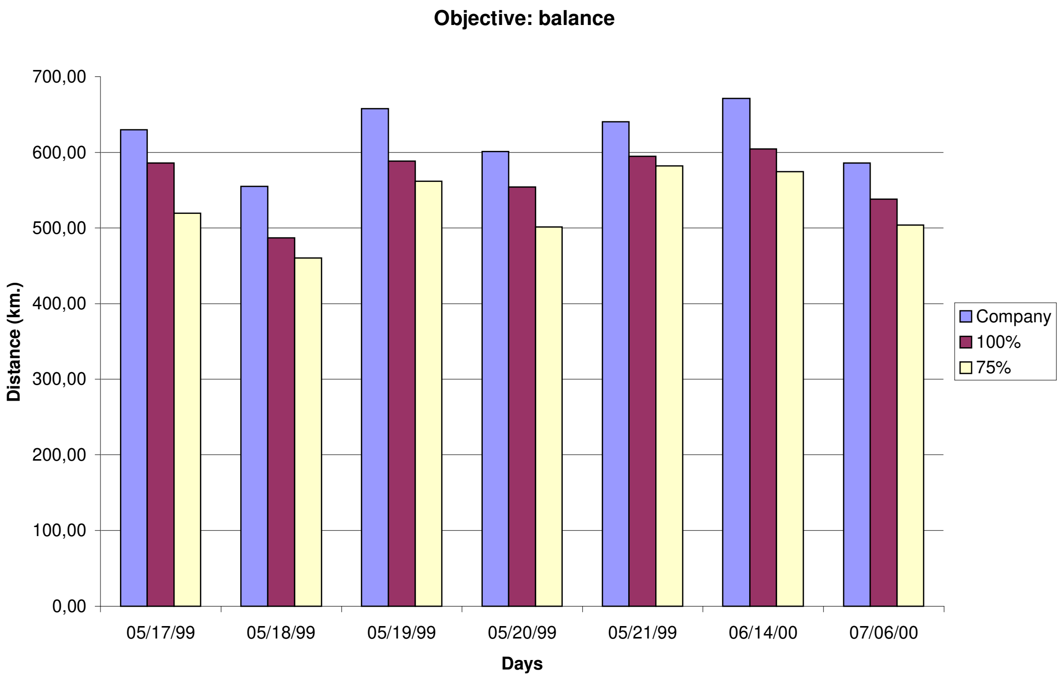

Figure 2

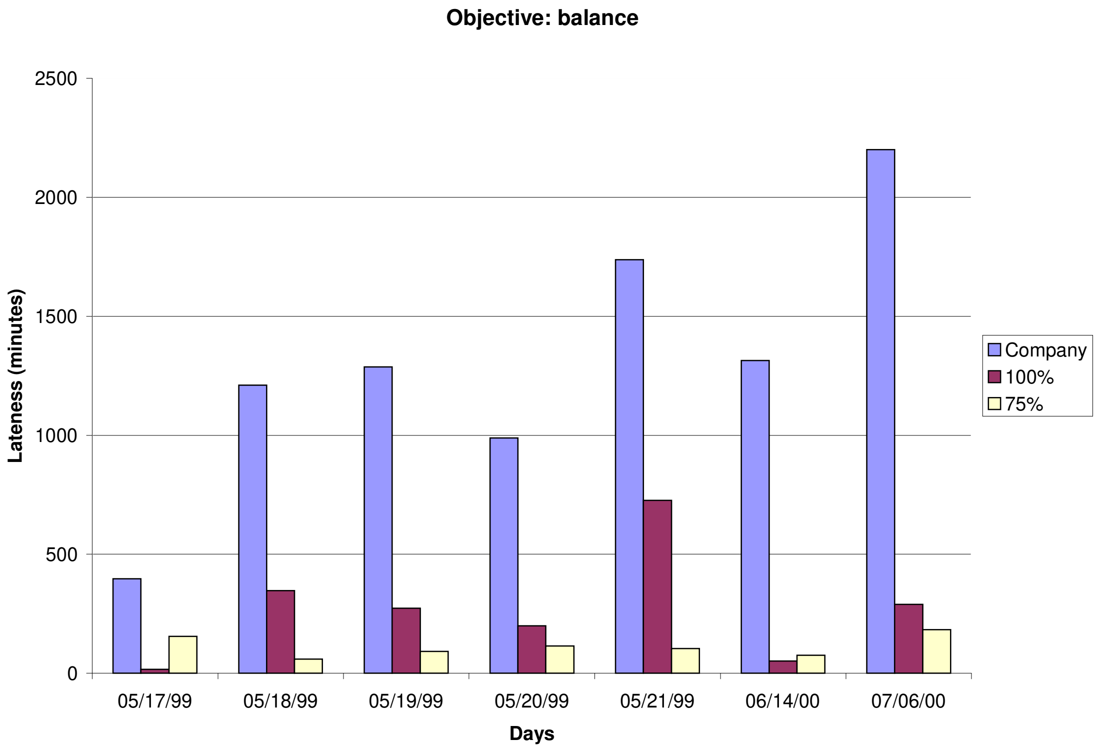

Figure 3

A much larger decrease is observed in the total lateness shown in Figure 3. The *RutaRep* solutions incur a total lateness that is 80.99% smaller than the company solution if the 100% option is used and 88.19% with the 75% option. The conclusion is that the drivers are interested in minimising the distance travelled and they do not mind that a high level of lateness was incurred. Furthermore, it is extremely difficult to manually build routes that satisfy the time windows while maintaining the distance travelled inside reasonable limits.

Table 2 summarises the results obtained with *RutaRep* using the other two objectives (minimising the distance and minimising the lateness) with the options 100% and 75% for zone satisfaction (the total distance and lateness are shown for each solution). Recall that when the objective is to minimise the distance, *RutaRep* still takes into account the lateness, although to a lesser extent, and vice versa. In fact, it can be observed that, even with the objective of minimising lateness, the total distance is slightly less than that of the company solution and a dramatic decrease in lateness is achieved.

	Day	05/17/99	05/18/99	05/19/99	05/20/99	05/21/99	06/14/00	07/06/00
Company		629.67 397	555.24 1211	657.79 1287	601.06 989	640.46 1738	671.14 1314	585.55 2201
RutaRep **(minimise distance)**	100%	551.71 1152	471.38 957	548.34 2642	528.49 1451	556.46 1709	579.35 925	518.71 1106
	75%	501.36 610	441.82 514	519.83 2637	468.31 2456	505.61 1696	554.37 877	478.68 1056
RutaRep **(minimise lateness)**	100%	585.58 12	501.31 251	618.49 206	598.86 0	650.96 345	610.83 45	569.16 133
	75%	547.20 11	474.63 1	581.88 69	544.67 0	613.32 13	635.87 0	553.69 48

Table 2: Distance and lateness of the solutions with different options

We have tried the program with lower values for the percentage of zone satisfaction (50% and 0%), and the solutions obtained are not substantially better than the ones obtained with the 75% option. This shows that it is advisable to maintain the current politic of assigning a vehicle to a zone but certain flexibility is recommended to allow the balancing of the daily duties and to improve customer satisfaction. Furthermore, we have tried increasing the number of iterations made by the program in the tabu search phase, but the improvements that were achieved are not important and could be negligible in practise.

Finally, we have tried the option of reducing the number of vehicles available to service the same set of orders. Table 4 shows the total distance and lateness of the solutions obtained by *RutaRep* on each data set with a decreasing number of vehicles: 7, 6, ..., 3 (the eliminated vehicles are always those of lower capacity). The last column in the table shows the solutions obtained when *RutaRep* is used to automatically minimise the number of vehicles (in this case, the number of vehicles is shown in brackets). Note that the solutions are not too worse if only one or two vehicles are eliminated.

	# vehicles					
day	**7**	**6**	**5**	**4**	**3**	**min_veh**
05/17/99	512.28 146	510.50 108	484.09 91	491.44 819	505.20 3216	510.34 1135 (4)
05/18/99	429.82 38	454.32 167	426.89 574	435.21 2377	347.88 7846	456.78 2748 (4)
05/19/99	558.95 79	551.47 169	566.54 695	545.85 3861	488.30 12611	588.86 656 (5)
05/20/99	500.56 114	493.88 134	437.45 958	492.73 2336	466.64 8098	516.84 2554 (4)
05/21/99	554.98 121	549.88 197	553.24 1716	507.22 5758	522.83 16369	573.75 1914 (5)
06/14/00	559.83 73	554.86 92	530.79 727	517.76 1900	625.47 5809	553.94 1564 (4)
07/06/00	502.83 183	493.62 211	490.90 1421	494.03 2956	484.69 8293	490.33 2349 (4)

Table 4: Distance and lateness of the solutions when the number of vehicles decreases

RutaRep was run on a PC at 330 Mhz. and the time taken in finding a solution was never more than a few seconds. The option which is most time consuming is that of minimising the number of vehicles, but even in this case, the time taken was less than one minute.

Conclusions

We have developed the computer package *RutaRep* to automatically generate delivery routes. *RutaRep* is very user friendly because it is designed specifically to meet the needs of the company and it is implemented in a windows environment. Thus, its use can be learned in one short session without the need for a complicated manual.

The solutions provided by *RutaRep* substantially improve the ones currently used by the company, especially for customer service as it dramatically decreases the lateness.

RutaRep has also shown that it is possible to decrease the number of vehicles with only a slight increase in lateness. This demonstrates that the current fleet would be able to service a larger number of customer orders. At the present time, the company is experiencing an important growth in their order book and is carrying out the study for the necessary interface between their computer system and our program to use it as a decision support system for building the delivery routes.

References

Taillard, E., Ph. Badeau, M. Gendreau, F. Guertin and J-Y. Potvin (1997). "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows". Transportation Science, 31-2, 170-186.

Solomon, M.M. (1987). "Algorithms for the Vehicle Routing and Scheduling Problems with time Window Constraints". Operations Research, 35-2, 254-265.

Potvin, J-Y. and Rousseau, J-M. (1993). "A parallel route building algorithm for the vehicle routing and scheduling problem with time windows". European Journal of Operational Research 66, 331-340.

Desrochers, M., Lenstra, J.K., Savelsberg, M.W.P. and Soumis, F. (1988). "Vehicle routing with time windows: optimization and approximation". B.L. Golden and A.A. Assad eds. in *Vehicle Routing: Methods and Studies*. North-Holland, Amsterdam, 65-84.

Solomon, M.M. and Desrosiers, J. (1988). "Time window constrained routing and scheduling problems". Transportation Science 22-1, 1-13.

Desrosiers, J., Dumas, Y., Solomon, M.M. and Soumis, F. (1995). "Time constrained routing and scheduling". M.O. Ball et al. eds. in *Network Routing*, Handbooks in Operations Research and Management Science, Volume 8, North-Holland, Amsterdam, 35-139.

Gendreau, M., Laporte, G. and Potvin, J-Y. (1997). "Vehicle Routing: Modern Heuristics", in *Local Search in Combinatorial Optimization,* Aarts and Lenstra eds. John Wiley and Sons Ltd., 311-336.

Semet, F. and Taillard, E. (1993). "Solving real-life vehicle routing problems efficiently using tabu search". Annals of Operations Research 41, 469-488.

Rochat, Y. & Semet, F. (1994). "A tabu search approach for delivering pet food and flour in Switzerland". Journal of the Operational Research Society, 45, 1233-1246.