HAL
open science

A decision support system for vehicle routing based on model inversion and data analysis

Bernat Gacias, Pierre Lopez, Julien Cegarra

To cite this version:

Bernat Gacias, Pierre Lopez, Julien Cegarra. A decision support system for vehicle routing based on model inversion and data analysis. International Conference of Modeling and Simulation (MOSIM'10), May 2010, Hammamet, Tunisia. 10p. hal-00484249

HAL Id: hal-00484249

https://hal.science/hal-00484249v1

Submitted on 18 May 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

8th *International Conference of Modeling and Simulation - MOSIM'10 - May 10-12, 2010 - Hammamet - Tunisia*
"Evaluation and optimization of innovative production systems of goods and services"

A decision support system for vehicle routing based on model inversion and data analysis

Bernat GACIAS, Pierre LOPEZ

CNRS; LAAS
7 avenue du Colonel Roche, F-31077 Toulouse, France
Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; F-31077 Toulouse, France
bernat.gacias@laas.fr, pierre.lopez@laas.fr

Julien CEGARRA

CNRS; CLLE
5 allées Machado, F-31058 Toulouse, France
Université de Toulouse
julien.cegarra@univ-jfc.fr

ABSTRACT: *In this paper, we present a three-phase solving mechanism for the vehicle routing problem. The solving mechanism is part of a decision support system architecture deduced from an interdisciplinary study. We highlight that human factors and dynamic aspects are generally ignored in the classical approaches to solve the problem. In our approach, a link is done between methods of operations research and an ecological interface design coming from cognitive ergonomics. We focus our study in how to manage the constraint relaxation if the problem is not satisfiable. We propose and evaluate model inversion techniques and data classification based methods in order to determine the most suitable constraints to relax in priority.*

KEYWORDS: *Decision support system, vehicle routing problem, constraint programming, model inversion, data classification.*

1 INTRODUCTION

The importance of the optimization of the vehicle routing problem (VRP) has considerably increased during the last two decades. The aggressive competition forces the companies involved in goods and services to guarantee a minimum level of quality of service for the customers.

The *vehicle routing problem* consists in determining the routes of a fleet of vehicles for the transportation of goods or passengers according to some customer demands (Figure 1).

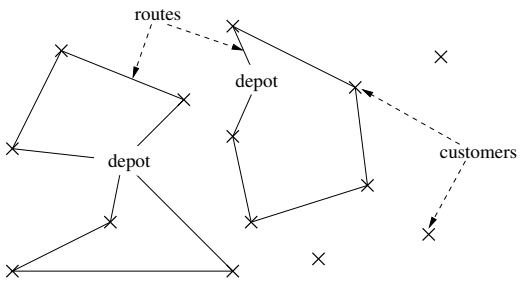

Figure 1: Vehicle routing

Besides the economical and social aspects, the VRP is a very interesting problem that has drawn the researchers attention. A large number of methods to solve very efficiently the various existing variants of the problem have been proposed (Barnhart and Laporte, 2007; Golden, Raghavan and Wasil, 2008; Toth and Vigo, 2001).

Many examples of decision support systems (DSSs) are available in the literature. In Basnet, Foulds and Igbaria (1996) a DSS for the construction of milk tanker routes in New Zealand is proposed. The authors propose to use the sweep algorithm (Gillet and Miller, 1974) mixed with the farthest insertion algorithm (Syslo and N. Deo, 1983) to construct the routes. Similarly, in Ruiz, Maroto and Alcaraz (2004) the authors propose an interactive DSS for a feed compounder company. A two-stage, route enumeration and integer programming optimization, exact approach is proposed to solve the customer-constrained VRP. More recently, Ray (2007) and Santos, Coutinho-Rodrigues and Current (2008) propose spatial decision support systems integrating Geographical Information Systems in the DSS. The first article uses a hybrid-iterative search algorithm to solve the problem while the second paper proposes to use a modified version of the path-scanning (Golden, DeArmon and Baker, 1983). Finally, Mendoza, Medaglia and Velasco (2009) propose a DSS for a real distance-constrained VRP. The authors propose a memetic algorithm using an adapted variant of the *savings* heuristic (Clarke and Wright, 1964) to get the initial population, a local search procedure inspired on the λ interchange method proposed by Osman (1993) to improve the solutions, and two integer-programming clustering models to distribute the cus-

tomers into workdays.

However we consider that the DSSs to solve the VRP have two important limitations. The first one is that human factors are not much considered in the modeling phase of the problem. Now, humans play a major role by carrying out specific operations and making decisions whenever perturbations occur. For example, when supervising the routes, the user of the resolution methods has enough knowledge and know-how to anticipate emergencies, vehicle breakdowns, traffic jams, driver substitutions, and so on (see Cegarra (2008) for a discussion on human contributions).

It is important to allow the human be part of the system. We consider that the robustness of the proposed solutions may increase if the human is allowed to act on constraints. It is usually noted that experienced individuals build schedules that are robust enough to disturbances (Cegarra, 2008). For example, the experienced individuals may assign the operations to the least flexible resources with the intention of preserving the most flexible resources for later unexpected disturbances. Another example is the use of more than one hundred different "human" heuristics to tentatively anticipate problems in order to build robust schedules.

The second limitation is that these models are not ready to deal with the rapid changing situations. In the logistics domain the constraints may even change, in some extreme cases, before the end of the modeling phase. The model still has to remain valid recovering these changes. The DSSs that we find in the literature are not adapted to deal with the changes, for example the proposed solver tools do not allow to consider new types of constraints without a system redesign.

In order to overcome these limitations, we propose in Gacias, Cegarra and Lopez (2009) a generic architecture for a decision support system for the VRPs (see Figure 2). In the article, an interdisciplinary approach with two components is proposed: (1) an ecological interface based on the abstraction hierarchy resulting from a work domain analysis (Rasmussen, Pejtersen and Goodstein, 1994; Vicente, 1999a); (2) solving mechanisms based on Operations Research techniques, in particular Constraint Programming (Dechter, 2003; Rossi, van Beek and Walsh, 2006).

We consider the work domain analysis as a first step for the design of a VRP decision support system. A solving system based on optimization techniques is also part of the DSS. The user could interact with the DSS through an *Ecological Interface* (Vicente, 1999b; Vicente, 2002). The human could participate in the modeling and also in the problem solving (Figure 2). The decision-making process is shared between the human and the proposed specific algorithms.

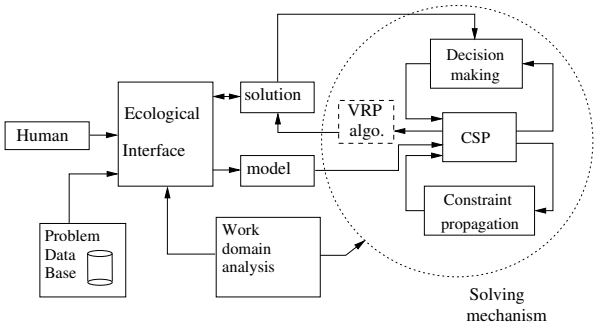

Figure 2: Architecture for a VRP Decision Support System

The Ecological Interface is the interface capable to represent the abstraction hierarchy as an external mental model for the resolution of the problem. The physical and the functional information are displayed for the interface in order to make constraints and complex relationships in the work environment perceptually obvious to the user. This allows more of users' cognitive resources to be devoted to higher cognitive processes such as problem solving and decision making. By reducing mental workload and supporting knowledge-based reasoning, the Ecological Interface aims to improve user performance and overall system reliability for both anticipated and unanticipated events in a complex system.

We focus here on the solving mechanism description. In Section 2 we formally describe the vehicle routing problem and the problem constraints. We present in Section 3 the algorithms used and the interfaces proposed to facilitate the user participation in the decision-making process. The model inversion techniques and the data classification methods are proposed in Section 4. Finally, the Section 5 is devoted to computational results.

2 PROBLEM STATEMENT

The problem considered is the VRP taking into account the constraints deduced from the Work Domain Analysis. In this section, we define the objects of the problem and we give a formal definition of the constraints.

Let us consider the objects of the problem: the customers ($C_i,\ i = 1..nc$), the vehicles ($V_j,\ j = 1..nv$), the drivers ($D_l,\ l = 1..ndr$), the depots ($De_t,\ t = 1..nde$), and the products ($P_k,\ k = 1..np$).

A set of nc customers, each with a demand d_i^k for each product and a time service ts_i, has to be served by a set of nv vehicles. A vehicle j starts the route from a depot t and ends at the same depot.

We describe some characteristics of the problem.

First, the customer demands can be indifferently deliveries ($d_i^k > 0$) or pick-ups ($d_i^k < 0$). Second, each vehicle j has a maximal weight capacity (C_j^w), a maximal volume capacity (C_j^v) and a maximal authorized length (C_j^l). As the same way, each product k has a weight (P_k^w), a volume (P_k^v) and a length (P_k^l). Let us consider the variable $x_i^j = 1$ if customer i is served by vehicle j, and $x_i^j = 0$ otherwise. We can deduce a set of constraints for the deliveries (Equations 1).

$$\sum_i^{nc} \sum_k^{np} x_i^j * \max(d_i^k, 0) * P_k^w \leq C_j^w \quad \forall\ j = 1..nv$$

$$\sum_i^{nc} \sum_k^{np} x_i^j * \max(d_i^k, 0) * P_k^v \leq C_j^v \quad \forall\ j = 1..nv \quad (1)$$

$$\max_{i, d_i^k > 0} (x_i^j * P_k^l) \leq C_j^l \quad \forall\ j = 1..nv$$

and a set of constraints for the pick-ups (Equations 2):

$$\sum_i^{nc} \sum_k^{np} x_i^j * \max(-d_i^k, 0) * P_k^w \leq C_j^w \quad \forall\ j = 1..nv$$

$$\sum_i^{nc} \sum_k^{np} x_i^j * \max(-d_i^k, 0) * P_k^v \leq C_j^w \quad \forall\ j = 1..nv \quad (2)$$

$$\max_{i, d_i^k < 0} (x_i^j * P_k^l) \leq C_j^l \quad \forall\ j = 1..nv$$

If there exist any other vehicle limitation for a product property, the DSS offers the possibility to add new constraints of this type. Third, for each vehicle the distance (V_j^D), the time (V_j^T), and the number of customers (V_j^C) may be limited. Let us define R_j as the route of vehicle j, $D(R_j)$ and $T(R_j)$ are the distance and the time of R_j, respectively. The following new set of constraints is defined:

$$D(R_j) \leq V_j^D \quad \forall\ j = 1..nv$$

$$T(R_j) \leq V_j^T \quad \forall\ j = 1..nv \quad (3)$$

$$\sum_i^{nc} x_i^j \leq V_j^C \quad \forall\ j = 1..nv$$

Fourth, time windows ($TWC_i = [r_i, d_i]$) are considered for each customer. The customer has to be served inside the interval of its time windows. Depot time windows ($TWDe_t = [r_t, d_t]$) and driver ($TWDr_l = [r_l, d_l]$) time windows are also considered. The vehicle departures and arrivals have to occur inside the depot time windows, and drivers cannot work outside their time windows. Fifth, precedence and immediate precedence constraints between customers can be considered. If customer i precedes customer i' ($i \prec i'$), customer i has to be served before customer i', and if customer i immediately precedes customer i', customer i' has to be served after customer i without delay by the same vehicle. Finally, we take into account allocation or not-allocation constraints between the objects of the problem. For example, a product may be allocated to a vehicle due to transportation conditions, or one customer cannot be allocated to a vehicle because of route access characteristics.

3 SOLVING MECHANISM

The solving mechanism is divided in three independent phases: the vehicle selection, the customer allocation and the routes creation. In this section we describe the algorithms and the user interfaces proposed.

For each phase, we propose different control modes to solve the problem. In van Wezel and Cegarra (2007) five control modes are proposed: manual, advisory, dynamic allocation or interactive, supervisory, and automatic. The control mode defines the level of user participation in the problem solving. In the manual control, the human makes all decisions (none algorithmic assistance is provided). In the advisory control, the human makes the decisions and an algorithm checks the decisions feasibility. In the interacting mode or dynamic allocation, the decision-making process is shared between the human and the algorithms. In the supervisory control, the algorithm is executed automatically, then necessarily informs the user who accepts the decisions. Finally, in the automatic control, the algorithm makes all the decisions (the user is completely out of the decision-making process).

3.1 Vehicle Selection

The vehicle selection phase is used to determine the vehicles to use to solve the problem (Figure 3). The available vehicles with the same characteristics are grouped as a same type of vehicle (tv is the number of vehicle types). Let us define the number of vehicles of type τ used to solve the problem (nv_τ) as the decision variables for this phase.

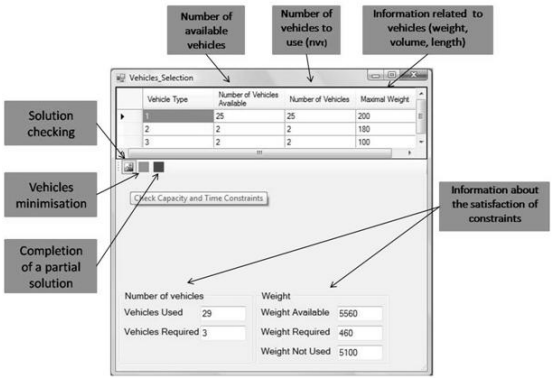

Figure 3: User interface for the vehicle selection

In some cases a chosen solution for the vehicle selection is not feasible. Therefore the DSS informs the user about the margin of constraint satisfaction (see Figure 3). That way, the user has pieces of information about the problem feasibility and the flexibility of the tested solution.

3.1.1 Advisory mode: user solution checking

The solution checking is the advisory control mode of the phase. It verifies whether a solution proposed by the user is a feasible solution. First, we verify that the weight capacity, the volume capacity and the maximum length of the vehicles is sufficient to serve the customer demands (Equations 4). We also check whether a given type of vehicle has to be actually present because of allocation constraints.

$$\sum_{i=1}^{nc}\sum_{k=1}^{np} d_i^k * P_k^w \leq \sum_{\tau=1}^{tv} nv_\tau * C_\tau^w$$

$$\sum_{i=1}^{nc}\sum_{k=1}^{np} d_i^k * P_k^v \leq \sum_{\tau=1}^{tv} nv_\tau * C_\tau^v \qquad (4)$$

$$\max_{i,d_i^k \neq 0} (P_k^l) \leq \max_{nv_\tau > 0} (C_\tau^l)$$

Then, we compare the number of vehicles used in the solution ($\sum_\tau^{tv} nv_\tau$) with a lower bound for the number of vehicles necessary to solve the problem (LB_{nv}). We compute different lower bounds using the information about the customer and the depot time windows (LB_{nv}^{CTW}, LB_{nv}^{DTW}), the distance limitation for vehicle (LB_{nv}^{Dmax}), the time limitation for vehicle (LB_{nv}^{Tmax}) and the maximal authorized number of customers for vehicle (LB_{nv}^{Cmax}). The LB_{nv} is equal to the maximum value of the lower bounds. We verify that $\sum_\tau^{tv} nv_\tau \geq LB_{nv}$.

To compute a lower bound LB_{nv}^{CTW} using customer time windows $TW_i = [r_i, d_i]$, we determine for each couple of customers if they are in conflict. A customer is in conflict with another customer if both cannot be served by the same vehicle. Let us consider $t_{ii'}$ as the time necessary to go from customer i to customer i'. For a couple of customers $< i, i' >$, if $r_i + ts_i + t_{ii'} > d_{i'}$ and $r_{i'} + ts_{i'} + t_{i'i} > d_i$, then the couple $< i, i' >$ is in conflict. LB_{nv}^{CTW} equal to the maximal number of customers in conflict.

For the LB_{nv}^{Dmax} and the LB_{nv}^{Tmax} computation, we use a lower bound for the distance ($LB_{distance} = \sum_i^{nc} \min_{i' \neq i}(d_{ii'})$) and for the time ($LB_{time} = \sum_i^{nc} \min_{i' \neq i}(t_{ii'})$, where $t_{ii'}$ is calculated using routes speed information). Then, $LB_{nv}^{Dmax} = LB_{distance} / \max_j(V_j^D)$ and $LB_{nv}^{Tmax} = LB_{time} / \max_j(V_j^T)$. The bound related to the limitation of the number of customers is equal to $LB_{nv}^{Cmax} = nc / \max_j(V_j^C)$.

The lower bound on the depot time windows (LB_{nv}^{DTW}) is computed using the lower bound for the time (LB_{time}). Then $LB_{nv}^{DTW} = LB_{time} / \max_t(d_t - r_t)$.

If the capacity constraints (Equations 4) are satisfied and the number of vehicles used is greater than LB_{nv} then the solution is accepted. If it is not the case, a constraint relaxation guide based on model inversion techniques is proposed to the user (see Section 4).

3.1.2 Supervisory mode: vehicle number minimization

The vehicle number minimization tool corresponds to the supervisory control mode of the phase. We compute the solutions for the number of vehicles that satisfy the capacity constraints (see Equations 4) and minimize the number of vehicles ($\min \sum_\tau^{tv} nv_\tau$).

We apply an exact method to obtain a solution for the minimal number of vehicles necessary to satisfy the weight or volume constraints. The algorithm selects, among the available vehicles, the vehicle with the biggest capacity except for the last vehicle (the sum of vehicle capacities is enough to satisfy the constraint) where the vehicle selected is the vehicle with the smallest capacity that allows the satisfaction of the constraint.

Finally, we propose an algorithm in order to offer to the user a list of interesting solutions with a minimal number of vehicles (see Algorithm 1). $Is_Possible(solution, extra_capacity, t_{rep}, t_{add})$ is a function that returns true if it is possible to replace a selected vehicle in a solution with a vehicle with smaller capacity, without violating the constraints. t_{rep} is the type of vehicle with the biggest capacity that can be replaced, and t_{add} the type of vehicle to add to the solution. Then we add a new solution to

solution_list. Note that the list of solutions is not a complete enumeration but we obtain a list of interesting solutions for the user.

Algorithm 1

//Insert the solution into the list
$solution_list.push_front(solution)$;
while $Is_Possible(solution, extra_capacity, t_{rep}, t_{add})$ **do**
  //Create new solution deleting one vehicle of
  //type t_{rep} and adding one vehicle of type t_{add}
  $solution \leftarrow Create_Solution(solution, t_{rep}, t_{add})$;
  //Add the new solution into the list
  $solution_list.push_front(solution)$;
  //Update the over capacity of the solution
  $extra_capacity = over_capacity + C^w_{t_{rep}} - C^w_{t_{add}}$;

Again, in case of an infeasible problem because there are not enough vehicles to satisfy the constraints, a constraint relaxation guide is proposed to the user (see Section 4).

3.1.3 *Interactive mode: complete a partial solution*

In the interactive control an algorithm completes a partial solution proposed by the user. The algorithms proposed are the same algorithms presented for the minimization of vehicles, but here we start the minimization process including in the solution the vehicles already selected by the user.

3.2 Customer Allocation

The customer allocation is another subtask of the problem. In the customer allocation phase, we determine for each customer the vehicle to be allocated. For this phase of problem, we take into account all the constraints of the problem (see Section 2). Thus, the solution obtained is a feasible customer allocation. Figure 4 displays the user interface for the customer allocation; the customer and vehicle information is easily accessible and the DSS offers to the user the possibility to make or to modify any decision. The system supports the advisory, the supervisory and the interactive control modes.

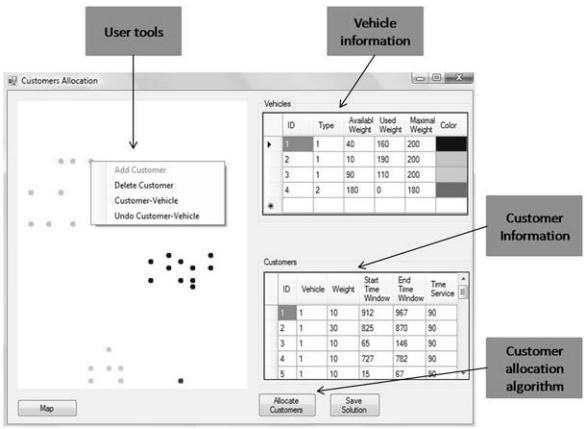

Figure 4: User interface for the customer allocation

3.2.1 *Advisory mode: user customer allocation*

The user proposes an allocation for each customer. After each decision, the system verifies whether it is a feasible decision according to the problem constraints. We use a limited discrepancy search (LDS) algorithm (Harvey and Ginsberg, 1996) to find a feasible solution. If no solution is found, the user has to backtrack to his previous choices or has to modify his last decision.

3.2.2 *Supervisory mode: algorithm customer allocation*

In the supervisory control, a complete solution is proposed by the system. The user can modify the decisions of the algorithm. The sweep algorithm (Gillet and Miller, 1974) principle is used for the customer allocation. First, we start to allocate the customers that have to be allocated to a specific vehicle because of allocation constraints. Then, we allocate each customer (selected following the sweep algorithm) on the non-empty vehicle with the smallest mean distance between the customer and the already allocated customers. If there is not a not-empty available vehicle to allocate the customer, then we allocate the customer on a new vehicle.

3.2.3 *Interactive mode: complete a partial solution*

The user provides the allocation for some of the customers and the algorithm ends the customer allocation from the partial solution. The algorithm used is the same than in the supervisory mode.

3.3 Route Creation

In this phase, the sequence of customers for each vehicle is determined. Like for the customer alloca-

tion all the necessary information are available for the user. Figure 5 shows the proposed user interface. Three control modes are also proposed for this phase of problem.

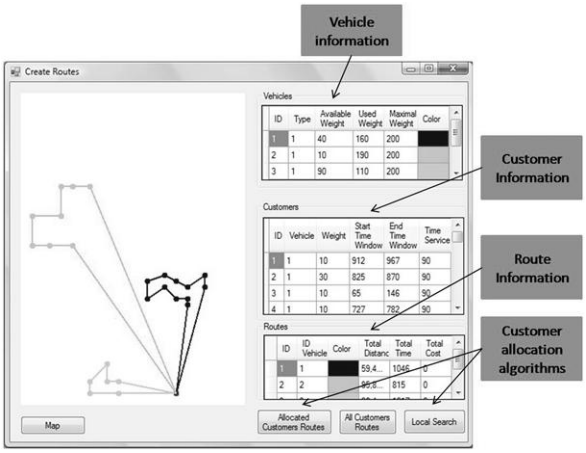

Figure 5: User interface for route creation

3.3.1 Advisory mode: user route creation

The user proposes the sequence of customers for each vehicle. An algorithm checks for the feasibility after each decision, we verify that the sequence is feasible and the not yet sequenced customers can be added to the sequence without violating the problem constraints.

3.3.2 Supervisory mode: algorithm routes creation

Once the customer allocation is done, we propose to use the *savings* algorithm (Clarke and Wright, 1964) to compose the sequence. When a customer cannot be inserted, the LDS algorithm is used to find a solution.

In the case when the customer time windows are very restrictive, a constraint-based heuristic can be used instead of the *savings* algorithm to solve the conflicts. Indeed, Kilby, Prosser and Shaw (2000) demonstrate that goal-based algorithms like savings may become very inefficient when the problem is over-constrained. In our case, if the algorithm is modified according to the constraints nature, the computation time spent by the LDS algorithm can be considerably reduced. We propose a variation of the *regrets* algorithm (Franklin and Sheng-Yuan, 1999). The algorithm selects, as the next customer to sequence, the customer with less positions to be inserted until the conflicting customers are resolved; then the savings algorithm is used.

3.3.3 Interactive mode: complete a partial solution

The user begins the sequence design supported by the checking algorithm and the algorithm completes the solution. For that phase, we propose a local search to improve the solution. The user can select between a local search method to improve the route for one vehicle, preserving the decisions already made (customer allocation, partial sequence, ...), or to use an algorithm for a global optimization.

4 MODEL INVERSION

We propose to use model inversion techniques in order to offer user support for constraint relaxation, when the problem becomes infeasible. The model inversion consists in the exchange of roles between the decision variables and the parameters, endogenous (related to a given decision center) and exogenous (not-related to the decision center). In model inversion, the decision variables become parameters, that restrict the decision space, and the parameters become decision variables, that we can use to deduce inferences.

We describe here a model inversion for the first phase of the resolution only: the vehicle selection. The analysis of the parameters subject to be modified for the vehicle selection phase is quite simple, since the relations between the parameters and the number of constraints considered are limited.

The first step of the method is to identify the parameters that integrate the constraints (the constraints of Equation 4 and the lower bounds presented in Section 3.1). We propose to divide the parameters in several groups. Each parameter is related to its physical object group (customers, vehicles, drivers, depots, products). When a constraint or a group of constraints is violated, the DSS asks the user what are the group of parameters that are candidate to be modified in order to obtain a feasible problem. That way, we reduce the number of parameters to inverse and the number of operations to compute.

For example, if we consider the capacity constraints (see Equation 4), we find parameters related to customer group, the demand of each customer ($\sum_k^{np} d_i^k$), and parameters being part of the vehicle group, the maximal capacity for each type of vehicle (C_j^w, C_j^v, C_j^l).

Once the parameters are defined, we propose an inversion model procedure for each constraint. The procedure is called when a constraint is violated. It computes for each parameter the limit value to obtain a feasible problem, or propose, when we deal with an extended set of values, an ordered list of constraints to relax. The proposed choices try to minimize the

impact of the relaxation and try to propose the most suitable modifications.

Each model inversion procedure is divided into two phases. At the first stage, the procedure identifies the parameters susceptible to be modified in order to obtain a feasible problem. The second phase uses the data analysis to propose the list of the most suitable parameters to modify.

The parameters identification for each constraint are not presented in this paper. In this section, we describe the generic data classification methods with different constraint adapted criteria that are used to propose a list of options to the user. The methods have to be adapted to each constraint in order to propose the best adapted choice to the user.

The first criterion used to class the customers is a geographic criterion. We propose the k-means (Forgy, 1965) which aims to partition nc customers into K clusters ($P = P_1 \cup P_2 \cup .. \cup P_K$) as homogeneous as possible in relation to a similarity measure defined for each couple of customers. The similarity measure is the distance between the customers and the number of clusters is the number of vehicles of the solution ($\sum_{\tau}^{nt} nv_{\tau}$). The k-means algorithm is an iterative algorithm (see Algorithm 2). $Select_Means(k, customers)$ creates K means, we propose to uniformly spread the initial means along the longest axis of the problem taking the centroid of the customers as the center point. Then for each iteration, we compute for each customer the distance from the means, we allocate the customer to the cluster with the nearest mean, and finally we update the new means (centroid of each cluster). The algorithm stops when we observe no more changes between the clusters of two successive iterations.

Algorithm 2 k-means algorithm for geographic customer classification

```
begin
    //Select K means
    M ← Select_Means(customers);
    it ← 0;
    while (P^it != P^(it-1)) do
        //compute distances between customers and means
        Compute_Distances(M, customers);
        //allocate each customer to the nearest mean
        Allocate_Customers(P, M, customers);
        //update the means of each cluster P
        M ← Update_Means(P);
        it ← it + 1;
end
```

The customers being part of the same cluster have high probability to belong to the same route. For each customer, we compute a mean distance (dm_i) between the customer and the other customers of its cluster. We use the mean distance as indicator to decide whether a customer is candidate to be suppressed.

We propose a variant of the algorithm in order to adapt the clusters to the problem constraints. We do not accept customers in conflict on the same cluster when a capacity constraint is violated. And, in the case a time constraint is violated, then the demands of the customers allocated to a cluster cannot exceed the maximal capacity of the available vehicles. In both cases, the customer with the longest distance to the mean of the cluster is allocated to another cluster.

The second geographical criterion to consider is the distance between the customer and the nearest depot. The customer with a largest distance is also proposed as a candidate to remove from the problem.

We have two geographic-based customer classifications, now we propose a classification based on temporal information. The clustering approach is based on the Dynamic Cluster Algorithm (DCA) introduced by (Diday, 1971). DCA is an extension of k-means algorithm. It needs to define an allocation function (a dissimilarity measure), because the similarities can no longer be shown as an Euclidean distance, and also a way to represent the classes (the centroid does not exist anymore). We propose a dissimilarity measure based on customer time windows. The parameter measures the degree of centering between two time windows. Figure 6 displays the main relations between two time windows and the value of the dissimilarity measure for each configuration.

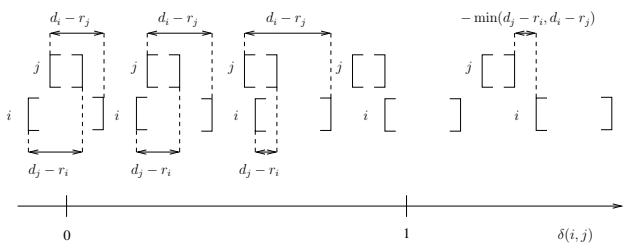

Figure 6: Main configurations between two time windows

$$\delta(i,j) = \begin{cases} 1 - \frac{\min(d_j - r_i, d_i - r_j)}{\max(d_j - r_i, d_i - r_j)} \\ \qquad if \min(d_i, d_j) \geq \max(r_i, r_j), \\ 1 - \frac{\min(d_j - r_i, d_i - r_j)}{\frac{1}{n}\sum_{i=1}^{n}(d_i - r_i)} \quad otherwise. \end{cases}$$

We also need an allocation and a representation function. The core of a cluster P_k is the customer C_k which its release date r_k is the nearest to the middle of the time interval $[\min_{i \in P_k} r_i, \max_{i \in P_k} r_i]$. At

each iteration, a customer $C_{i'}$ is allocated to a core C_k with $\delta(C_k, C_{i'}) = \min_{k=1..K} \delta(C_k, C_{i'})$.

To initialize the cores we propose Algorithm 3. The cores are uniformly distributed all along the release times scale.

Algorithm 3 Core Initialisation for the temporal-based classification

begin
- $time_horizon \leftarrow \max_{i=1..nc} r_i - \min_{i=1..nc} r_i$;
- $step \leftarrow time_horizon/K$;
- $t_1 \leftarrow \min_{i=1..nc} r_i + step/2$;
- $C_1 \leftarrow$ customer i with r_i nearest to t_1;
- $j \leftarrow 1$;
- **while** $(k \leq K)$ **do**
  - $t_k \leftarrow t_k + step$;
  - $C_k \leftarrow$ customer i with r_i nearest to t_k and C_i is not already a center;

end

The dissimilarity measure groups the customers with time windows interaction. For each cluster, we compute a parameter, the critical index CI_k, as the ratio between the number of customers of the cluster and the number of vehicles. It gives an idea of how critical the cluster is, a big value for the CI_k means that, because of customer time windows, we may have some troubles to serve the customers of the cluster. The CI_k is also a parameter that we consider to point the customers to suppress.

We observe that the time travel between the customers, parameter that can have a huge influence, is not considered for the time-based classification. To cover this lack, we propose to do a time-based classification and target a set of customers using the critical index (CI_k), and then, use the proposed k-means algorithm that allows the classification of the customers geographically. We suppose here that time and distance are directly proportional for the problem, otherwise the choices proposed for the method will not be a good candidate. For that reason, instead of the mean distance (dm_i) between the customers of each cluster a second temporal-based criterion is also proposed.

As we have mentioned, the customer classification methods have to be adapted to the constraint that is being violated in order to make the best suitable choice. For example, if a capacity constraint is violated, we use the geographic classification to cluster the customers and, in the case of one cluster is overload, we are going to propose to reduce the capacity of one customer that belongs to the overload cluster.

Each of the candidates proposed by the different criteria are given to the user. In order to help the user to select one of them, a statistical analysis is proposed. The analysis give an idea of how relevant is the criterion. We assume that the decision variable X for each criterion (mean distances between customers of a cluster, customers distance to the nearest depot, number of conflicts for each customer, and mean value of the dissimilarity measure) follows a normal distribution $N(\mu, \sigma)$ that can be approximated for $N(\bar{X}, S)$ where $\bar{X}$ and $S = \sqrt{\frac{(X-\bar{X})^2}{n-1}}$ are the mean and the mean deviation of the sample, respectively. We convert each of the normal distributions to the standard normal $N(0, 1)$, also called the *z-distribution*, using the function $z = \frac{X-\bar{X}}{S}$. The *z-distribution* allows us to compare the candidates of the different criteria. We can evaluate how relevant is the criterion using the z value of the candidate. For example, the values of the z nearest to zero indicate that all the customers have similar values for this particular criterion, so it is not very relevant. The user can use the z value of each candidate in order to decide what is the best one to remove from the problem.

5 COMPUTATIONAL RESULTS

In this section, we evaluate the criteria proposed in Section 4. We propose to optimally solve small-size instances (9 customers) of the capacited vehicle routing problem with customer time windows (CVRPTW). A complete enumeration of the feasible solutions is done. We compare the results obtained taking out of the problem the customer for which the various criteria give the priority: the *GDC* (mean distance between customers of the same cluster after the geographical clustering), the *GDD* (largest distance with the nearest depot), and the *TDC* (mean distance between customers of the same cluster after the temporal clustering), with the optimal solution. The selected customer *SC* is the customer with a z value farther than zero, usually the more relevant criterion. However, for small instances we cannot completely trust on the statistical analysis results because the number of elements is very poor.

To generate the instances we have randomly eliminate customers of the CVRPTW small-size instances of Solomon (Solomon, 1983). Depending on customer locations, the Solomon's instances are classified in three groups: clustered customers (C), random customers (R), and mixed customers (RC).

Table 1 displays, in the first column, the number of times (*NbOptDist*) the optimal solution for the distance minimization is reached when the problem is solved without the customer selected by the criterion. The second column (*AvgDev*) represents the average deviation from the optimal solution for the instances where the optimal solution is reached when the eliminated customer is not the customer selected

by the criterion. *AvgPos* indicates the average position of the solution when we class the solutions in a non-decreasing order of the objective function. For example, if the solution reached when the problem is solved without the customer selected by the criterion is the second best solution, then its position is 2. Finally, the last column (*NbSol*) represents the number of instances the decision of the criterion is the decision which a bigger number of feasible solutions. We can see this parameter as a mesure of the flexibility provided by the decision.

56 instances $nc = 9$	*NbOptDist*	*AvgDev*	*AvgPos*	*NbSol*
GDC-C	13 (17)	8.73 %	2.7	6
GDC-R	9 (23)	5.99 %	3.5	17
GDC-RC	9 (16)	5.01 %	3.2	5
TDC-C	5 (17)	14.98 %	4.5	7
TDC-R	3 (23)	11.46 %	5.4	7
TDC-RC	4 (16)	9.30 %	5.3	7
GDD-C	8 (17)	15.76 %	4.3	8
GDD-R	5 (23)	10.97 %	4.7	11
GDD-RC	3 (16)	9.12 %	4.5	4
SC	32	6.83 %	3.54	30

Table 1: Results for the distance optimization

We can deduce that for small instances the *GDC* is the most efficient criterion: it reaches the optimal solution over a half of the instances. The *AvgPos* is around 3 for the *GDC* criterion that means that when the criterion decision is not the optimal decision the proposition of the criterion keeps being a good proposition.

We also observe that the statistical analysis is efficient, the best candidate is proposed over 50% of the instances. However, we cannot take relevant information from the analysis, since the size of the samples is not big enough. We think that the performance of the analysis may increase for the medium-size and large-size problems.

6 CONCLUSIONS AND FURTHER WORK

In this paper, we have proposed a three-phase solving mechanism for a generic vehicle routing problem. The methods and the algorithms used to solve each phase of the problem have been presented. We have also proposed different solving control modes in order to let the user be part of the decision making system. The proposed decision support system minimizes the danger of user skill loss or user complacency because the system demands for interaction and it offers to the user the opportunity to use his abilities and knowledge to solve the problem.

We have proposed and tested, for the vehicle selection phase, model inversion techniques and data classification algorithms to support the user to relax some constraints when the problem becomes infeasible. The next step is to propose a procedure to identify the parameters that can be modified and to adapt the proposed classification methods for each kind of constraint. Finally, we plan to extend the work to the other two phases of the resolution. In that case, the model inversion mechanisms have to be more sophisticated for two reasons: the number of constraints and the number of relations between the parameters are much more important, and we have to manage a set of the decisions already made.

REFERENCES

Barnhart, C. and Laporte, G. (eds) (2007). *Handbooks in operational research and management science: Transportation*, Vol. 14, North-Holland.

Basnet, C., Foulds, L. and Igbaria, M. (1996). Fleetmanager: a microcomputer-based decision support system for vehicle routing, *Decision Support systems* **16**(3): 195–207.

Cegarra, J. (2008). A cognitive typology of scheduling situations: A contribution to laboratory and field studies, *Theoretical Issues in Ergonomics Science* **9**(3): 201–222.

Clarke, G. and Wright, J. V. (1964). Scheduling of vehicles from a central depot to a number of delivery points, *Operations Research* **12**(4): 568–581.

Dechter, R. (2003). *Constraint Processing*, Morgan Kaufmann, San Francisco, USA.

Diday, E. (1971). Une nouvelle méthode en classification automatique et reconnaissance des formes : la méthode des nuées dynamiques, *Revue de Statistique Appliquée* **19**(2): 19–33.

Forgy, E. W. (1965). Cluster analysis of multivariate data: efficiency versus interpretability of classifications, *Biometrics* (3): 768.

Franklin, Liu, F.-H. and Sheng-Yuan, S. (1999). A route-neighborhood-based metaheuristic for vehicle routing problem with time windows, *European Journal of Operational Research* **118**(3): 485–504.

Gacias, B., Cegarra, J. and Lopez, P. (2009). An interdisciplinary method for a generic vehicle routing problem decision support system, *International Conference on Industrial Engineering and Systems Management, IESM 2009, Montreal (Canada)* .

Gillet, B. E. and Miller, L. R. (1974). A heuristic algorithm for the vehicle dispatch problem, *Operations Research* **22**(2): 340–349.

Golden, B. L., DeArmon, J. and Baker, E. K. (1983). Computational experiments with algorithms for

a class of routing problems, *Computers and Operations Research* **10**(1): 47–59.

Golden, B., Raghavan, S. and Wasil, E. (eds) (2008). *The Vehicle Routing Problem: Latest Advances and New Challenges*, Vol. 43, Springer.

Harvey, W. D. and Ginsberg, M. L. (1996). Limited discrepancy search, *Proceedings of 14th IJCAI.*

Kilby, P., Prosser, P. and Shaw, P. (2000). A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints, *Constraints* **5**(4): 389–414.

Mendoza, J. E., Medaglia, A. L. and Velasco, N. (2009). An evolutionary-based decision support system for vehicle routing: The case of a public utility, *Decision Support Systems* **46**(3): 730–742.

Osman, I. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, *Annals of Operations Research* **41**(4): 421–451.

Rasmussen, J., Pejtersen, A. M. and Goodstein, L. P. (1994). *Cognitive Systems Engineering*, Wiley, New York.

Ray, J. J. (2007). A web-based spatial decision support system optimizes routes oversize/overweight vehicles in Delaware, *Decision Support Systems* **43**: 1171–1185.

Rossi, F., van Beek, P. and Walsh, T. (2006). *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*, Elsevier Science Inc., New York, NY, USA.

Ruiz, R., Maroto, C. and Alcaraz, J. (2004). A decision support system for a real vehicle routing problem, *European Journal of Operational research* **153**(3): 593–606.

Santos, L., Coutinho-Rodrigues, J. and Current, J. R. (2008). Implementing a multi-vehicle multi-route spatial decision support system for efficient trash collection in Portugal, *Transportation Research* **42** (Part A): 922–934.

Solomon, M. (1983). *Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms*, PhD thesis, University of Pennsylvania, USA.

Syslo, M. M. and N. Deo, J. S. K. (eds) (1983). *Discrete Optimization Algorithms*, Prentice-Hall.

Toth, P. and Vigo, D. (eds) (2001). *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.

van Wezel, W. and Cegarra, J. (2007). Developing user-oriented scheduling algorithms, *14th International Annual EurOMA Conference, Ankara, Turkey.*

Vicente, K. J. (1999a). *Cognitive Work Analysis: Toward Safe, Productive, and Healthy Computer-Based Work*, Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA.

Vicente, K. J. (1999b). Ecological interface design: Supporting operator adaptation, continuous learning, distributed, collaborative work, *Proceedings of the Human Centered Processes Conference* pp. 93–97.

Vicente, K. J. (2002). Ecological interface design: Progress and challenges, *Human Factors* **4**: 62–78.