

*This submission is an extended version of a paper presented at the 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems, Valenciennes, France, August-September 2010*

# Scheduler-oriented algorithms to improve human-machine cooperation in transportation scheduling support systems

Bernat Gacias<sup>1</sup>, Julien Cegarra<sup>2</sup>, and Pierre Lopez<sup>3,4</sup>

<sup>1</sup> Université Paris Est; CERMICS; 6-8 avenue Blaise Pascal, Cité Descartes, 77455 Marne-la-Vallée, Cedex 2, France

<sup>2</sup> CLLE; Université de Toulouse; Centre Universitaire, Place de Verdun, F-81012 Albi, France

<sup>3</sup> CNRS ; LAAS ; 7 avenue du Colonel Roche, F-31077 Toulouse, France

<sup>4</sup> Université de Toulouse ; UPS, INSA, INP, ISAE, UT1, UTM ; LAAS ; F-31077 Toulouse, France

## Abstract

A decision support system designed to enhance human-machine interaction in transportation scheduling is proposed. We aim to integrate human factors and ergonomics from the beginning of the design phase and to propose a system fitted with enough flexibility to be able to deal with the characteristics of a dynamic context such as transportation scheduling. In this interdisciplinary approach, a link is done between problem solving methods (operations research techniques and data classification algorithms) and human-machine interaction (solving control modes). A set of scheduler-oriented algorithms favouring human-machine cooperation for problem solving is proposed. Some of these algorithms have been efficiently tested on instances of the literature. Finally, an original framework aiming to assist scheduler in constraint relaxation when the problem becomes infeasible is proposed and evaluated.

**Keywords:** Decision support system, vehicle routing problem, work domain analysis, control modes, user-oriented algorithms, model inversion.

## 1 Introduction

Nowadays, many firms emphasize the need to support routing performance because of an increasingly competitive environment. For this reason, routing problems have been a predominant application area for decision support systems ever since they were first introduced, as evidenced by bibliometric analyses (Eom and Lee, 1990; Eom et al., 1998; Eom and Kim, 2006). The starting point for these support systems is usually to tackle the hard combinatorial problem arising from the numerous constraints that have to be taken into account, then to allocate any remaining components to the human planner. Sanderson noted back in 1989 that these support tools are not especially dominant in practice, and very little has changed since then. Vehicle routing is often performed by a single experienced individual or, more rarely, by a small group of individuals (see Cegarra (2008)). In field studies, humans appear to be crucial for taking the large set of constraints into account and adapting to changes in the domain (Sanderson, 1989; Jackson et al., 2004). At the same time, the integration of transportation technologies (GPS, EDI, GIS) is gradually changing the nature of these decision-making processes, in that vehicle routing systems now combine human planners and technologies.

In this paper, we begin by discussing constraint processing and the importance of modelling the work domain in order to assess planners' constraint processing. After that, the work domain analysis of the vehicle routing problem and the decision support system architecture is introduced at the end of Section 2.

Typical support systems focus on supporting the “mechanical” process of generating the schedule. Algorithms therefore function like a black box: the human has to provide the necessary information and has to adapt the outcome if the route contains errors, but during the actual generation process, s/he plays no role whatsoever. In Section 3 we present how to take into account the cooperative aspects between human and computer, which are both participating in the decision-making process.

Support needs above all to consider how human perform tasks and their performance. In the last section a model inversion framework allowing to support the human in relaxing constraints is presented.

## 2 Background

There have been a number of attempts to define the vehicle routing task structure on the basis of hierarchical task analysis (Rahimi and Dessouky, 2001) and cognitive task analysis (Wong and Blandford, 2002). However, when Cegarra and van Wezel (2010) compared the amount of information produced by these two methods, as well as by work domain analysis, they found that the latter was far more exhaustive in identifying constraints, not least because it provides a generic view of constraints and does not focus on usual or known tasks.

Work domain analysis (WDA) was developed by Rasmussen and colleagues (Rasmussen et al., 1994; Vicente, 1999). Instead of focusing directly on the tasks being considered by the decision-maker, it looks at the constraints imposed on behaviour by the environment. For an interface designer, this sometimes requires a change of point of view, as Vicente (2000) (p.63) pointed out: “A task can be defined as the set of actions that can or should be performed by one or more actors to achieve a particular goal. In contrast a work domain is the system being controlled, independent of any particular worker, automation, event, task, goal, or interface”. As previously stressed, for the purposes of assessing and supporting decision-making, completeness in identifying domain constraints is highly desirable in a vehicle routing system.

WDA is usually performed using an abstraction hierarchy which depicts the constraint space. The higher levels of the abstraction hierarchy describe functional information about the domain, whereas the lower levels describe physical information. Moreover, WDA usually looks at five levels of abstraction (Naikar et al., 2005): functional purposes, abstract functions, generalized functions, physical functions, and physical forms. In addition to this breakdown into physical and functional aspects, a part-whole distinction can also be made, taking several levels of details into account: system, subsystem and components. However, this part-whole distinction is not always made, as we will see below.

Initially, WDA was applied to “causal” systems guided by physical laws, as in nuclear power stations (Itoh et al., 1995), conventional power stations (Burns, 2000) and cement milling plants (van Paassen, 1995). In “causal” systems, the objective reality is imposed on the human operator (Vicente, 1999), as opposed to “intentional” systems, where the operator is the main agent of the domain and there are fewer references to an external environment. This is the case of routing problems, in which it is difficult to enumerate the domain constraints because they result from conventions, organizational objectives, formal or informal rules and operators’ goal. Vicente (1999) stressed that WDA should be performed independently of usual or known tasks, the aim being to provide an exhaustive breakdown that is resistant to changes in the situation. To this end, we sought to enhance the identification of domain constraints by extending the scope of the domain. More specifically, instead of focusing on currently known or usual cases in one particular situation (which inevitably leads to the inclusion of more details about the current situation), we set out to identify constraints from different situations documented in the literature. Each variant of the VRP would allow us to increase the completeness of the constraint space by con-

sidering constraints that might potentially help to identify planners' degrees of freedom. Our search for VRPs in scientific databases yielded more than ten thousand articles. However, only a few of them suggested genuinely new variants of the generic VRP (see Toth and Vigo (2001)). Instead, researchers tended to develop algorithms to solve known variants or design algorithms for multiple variants (e.g., Pisinger and Ropke (2007)).

In the main, the variants extend the domain by providing constraints related to a temporal perspective, including customers' time windows (VRPTW) and vehicles with limited capacity (CVRP), and more complex variants, such as trucks and trailers (TTRP), pick-up and delivery (VRPPD), or ones related to multiple depots (MDVRP), which are summarized in Table 1. As previously indicated, the WDA abstraction hierarchy organizes the planners' problem space in terms of different concepts that planners can then use for reasoning within a work system. We also consider that the classic five levels of the abstraction hierarchy are needed to describe the constraint space. Figure 1 summarizes the VRP domain according to these different levels.

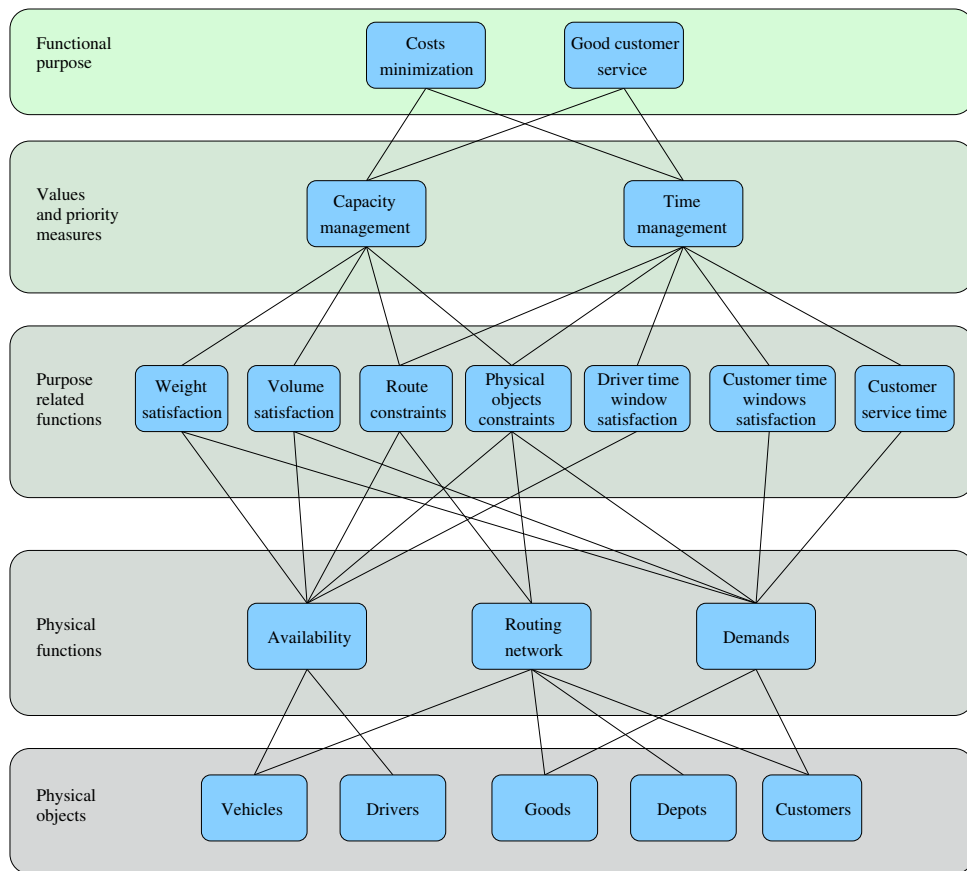


Figure 1: Work domain analysis of the generic vehicle routing problem

Details of the entities corresponding at each abstraction level are presented in Gacias (2010). The decomposition remains very generic according to studies and problem descriptions found in the literature. Nevertheless, most of the vehicle routing problems are covered and only some particular variants may required other components not considered on our analysis (see Gacias

(2010) for a scope analysis).

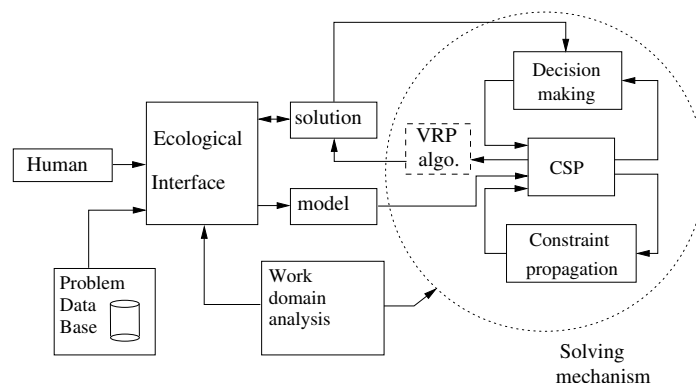


Figure 2: Decision support system architecture

A support system architecture aiming to cover some of the major issues concerning human-machine cooperation is proposed. An important clue often pointed by researchers is the advisability of sharing a unique reference system between the human and the machine. To this purpose, the work domain analysis has been considered crucial in order to identify the information needed to solve the problem. In our support system, as shown in Figure 2, the human and the solving mechanism manipulate both the same entities (physical objects and technical constraints) identified in the analysis. The system is composed by a set of human-machine interfaces seeking to efficiently assist the scheduler to solve problem tasks (see Gacias et al. (2010b) and Cegarra et al. (2011) for a detailed description of these interfaces). The information displayed has been identified in the work domain analysis. The idea is to connect these interfaces and the human-system interaction tools to the user-oriented algorithms presented in the next sections in order to favour human-machine cooperation for problem solving.

Variant	Name	Description	Objectives, constraints under consideration	Real-life example
VRP	Vehicle routing problem	A set of vehicles is to be routed from a single depot to multiple customers.	The objective is to minimize the number of vehicles and total travel distance. Workload balancing sometimes appears as a secondary objective of the problem.	See more specific cases below.
<b>Customers-related variants</b>				
DVRP	Dynamic VRP	A VRP where a subset of parameters of the problem (new customers, etc.) evolve in real time.	The objective is to minimize deviations from the original schedule (stability criterion) and minimize total travel distance and the number of vehicles.	Travelling repairman, emergency services, taxicab services. (e.g., Rahimi and Dessouky (2001))
VRPTW	VRP with time windows	A time window is associated with each customer (the interval at the depot is named the scheduling horizon).	The objective is to minimize the size of the vehicle fleet and of the total travel distance and waiting time needed to supply all customers at the required times.	Waste collection (e.g., Kim et al. (2006)).
VRPPD	VRP with pick-up and delivery	The demands of the customers are indifferently pick-ups or deliveries. Some variants of the problem are: <ul style="list-style-type: none"> <li>• Simultaneous PD: pick-up and delivery for each customer must be carried out simultaneously.</li> <li>• Mixed PD: customers either have a pick-up or a delivery demand.</li> </ul>	The objective is to minimize the size of the vehicle fleet and total travel distance, with the restriction that the vehicle must have sufficient capacity for transporting the commodities to be delivered and those picked up from customers and taken back to the depot. One of the most widely used strategies for solving the problem is delivery first and pick-up second.	Beverage industry (where filled bottles are delivered while the empty ones are collected), on-demand transportation (in reality, customer time windows are also considered).
ARP	Arc routing problem	The customers' demands are on arcs.	The objective is to minimize the number of vehicles and total travel distance.	Trash collection (e.g., Santos et al. (2008))
<b>Vehicle-related variants</b>				
CVRP	Capacitated VRP	The vehicles are identical and based at a single depot, and only the capacity restrictions for the vehicles are imposed.	The objective is to minimize the number of vehicles and total travel distance, with the restriction that vehicle capacity must be respected.	Milk collection from supplier farms (e.g., Basnet et al. (1996)).
DVRP	Distance-constrained VRP	The length (or duration or cost or number of customers) of each vehicle's route is bounded by a prescribed amount.	The objective is to minimize the size of the vehicle fleet and total travel distance.	The travelling auditor (e.g., Mendoza et al. (2009))
HVRP	Heterogeneous fleet VRP	There exist a number of heterogeneous vehicle types (capacity, fixed cost, variable travel cost).	The objective is to minimize the total cost of the routes. The best vehicle fleet composition has to be determined.	Feed compound delivery (e.g., Ruiz et al. (2004)).
TTRP	Truck and trailer RP	The fleet is made up of trucks and trailers. Some customers can only be served by a single truck but others can be served either by a single truck or by a truck pulling a trailer.	The objective is to minimize the total distance travelled, or the total cost incurred by the fleet. The truck's uncoupling and re-coupling of its trailer is authorized in some locations.	
<b>Depot-related variants</b>				
MDVRP	Multiple depot VRP	If the customers are clustered around depots, then the distribution problem should be modelled as a set of independent VRPs. However, if the customers and the depots are intermingled, then a multi-depot VRP should be solved.	The objective is to minimize the size of the vehicle fleet and total travel time, and the total demand of commodities must be met from several depots.	Gas and oil station delivery, ready-mixed concrete distribution (e.g., Matsatsinis (2004)).

Table 1: Variants of the vehicle routing problem

### 3 Decision support system algorithms

The algorithms integrated in the solving mechanism are presented in this section. The section starts with a formal description of the vehicle routing problem and of the problem constraints. The control modes for problem solving are then presented. Finally, we describe the proposed algorithms.

#### 3.1 Formal problem statement

The vehicle routing problem has already been formally defined (see for example Toth and Vigo (2001) for a collection of linear programs proposed in the literature for different variants of the problem). Our problem is the vehicle routing problem taking into account the constraints identified during the work domain analysis (see Section 2).

A set  $\{C_i\}_{i=1..nc}$  of customers has to be served by a set  $\{V_j\}_{j=1..nv}$  of vehicles. A vehicle starts its route from a depot in  $\{De_t\}_{t=1..nde}$  and ends at the same depot. To each vehicle is assigned a driver in  $\{D_l\}_{l=1..ndr}$ . Each customer demands an amount  $d_i^k$  for each product in  $\{P_k\}_{k=1..np}$  and a time service  $ts_i$ .

We describe here some characteristics of problem constraints. First, the customer demands can be indifferently deliveries ( $d_i^k > 0$ ) or pick-ups ( $d_i^k < 0$ ). Second, each vehicle  $j$  is capacity constrained: 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  $k$  is defined by a weight ( $P_k^w$ ), a volume ( $P_k^v$ ) and a length ( $P_k^l$ ).

Let us define the variable  $x_i^j$  to indicate if a customer  $i$  is served by the vehicle  $j$ .

$$x_i^j = \begin{cases} 1 & \text{if customer } i \text{ is served by vehicle } j \\ 0 & \text{otherwise} \end{cases}$$

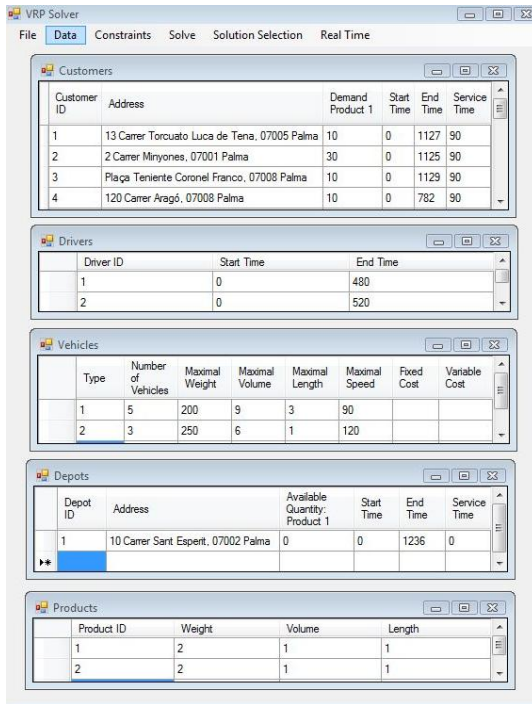
Two different sets of constraints can be then defined, one for the deliveries the other one for the pick-ups (depending on the sign of the demand), as follows:

$$\begin{aligned} \sum_i^{nc} \sum_k^{np} x_i^j \times |d_i^k| \times P_k^w &\leq C_j^w \quad \forall j = 1..nv \\ \sum_i^{nc} \sum_k^{np} x_i^j \times |d_i^k| \times P_k^v &\leq C_j^v \quad \forall j = 1..nv \\ \max_i(x_i^j \times P_k^l) &\leq C_j^l \quad \forall j = 1..nv \end{aligned} \quad (1)$$

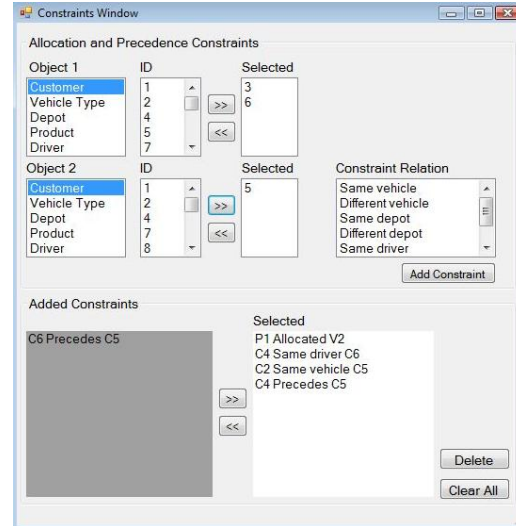
For each vehicle/route a limit for travelled distance ( $V_j^D$ ), time ( $V_j^T$ ), and for the number of customers ( $V_j^C$ ) may also be defined. Let us define  $R_j$  as the route of vehicle  $j$ ,  $D(R_j)$  and  $T(R_j)$  are the travelled distance and the duration of  $R_j$ , respectively. The following new set of constraints is then defined:

$$\begin{aligned} D(R_j) &\leq V_j^D \quad \forall j = 1..nv \\ T(R_j) &\leq V_j^T \quad \forall j = 1..nv \\ \sum_i^{nc} x_i^j &\leq V_j^C \quad \forall j = 1..nv \end{aligned} \quad (2)$$

Then, time windows ( $TWC_i = [r_i, d_i]$ ) are considered for each customer. Each customer has to be served inside the interval of its time windows. Besides, depot time windows ( $TWDe_t =$



(a) User interface for objects information



(b) User interface for physical objects constraints

Figure 3: **User interfaces for problem modelling.** *The scheduler specifies the problem components identified by the work domain analysis: physical objects (vehicles, drivers, goods, depot, and customers), their appearance characteristics (capacity availability and capacity required), their location characteristics (routing network), the temporal constraints (driver and customer time windows and customer service times), and the problem constraints between the objects (allocation and precedence constraints).*

$[r_t, d_t]$ ) and driver ( $TWDr_l = [r_l, d_l]$ ) time windows are also considered. The vehicle departures and arrivals have to take place inside depot time windows. Drivers cannot work outside their time windows. Precedence and immediate precedence constraints between customers can be also defined. 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 immediately after customer  $i$  by the same vehicle. Finally, allocation constraints between physical objects (depot-vehicle, customer vehicle, product-vehicle, etc.) are also considered.

Two different interfaces are proposed for problem modelling (see Figure 3). These interfaces are designed from the physical levels (purpose-related functions, physical functions, and physical objects) of the abstraction hierarchy.

### 3.2 User-oriented algorithms: control modes

The role of the human in problem solving has traditionally been a conflict issue between operations research and human factors researchers. Indeed, the latest claimed the advisability of give scheduler a more important role on problem solving process. Function allocation to humans and algorithms in planning and scheduling has been recently raised in van Wezel et al. (2010).

The authors propose principles for function allocation based on the analysis of problem subtasks, human capabilities, and the characteristics of the available algorithms to solve the problem.

In that context, a solving mechanism with different control modes to perform the problem tasks is proposed. Problem solving has been divided into three subtasks: vehicle selection, customer allocation, and route selection. This task division is justified in Gacias (2010) after the interview of two different companies transportation planners. In van Wezel et al. (2010), the authors propose five control modes: manual, advisory, interactive, supervisory, and automatic. The control mode specifies the degree of user participation in problem solving process. In the manual control mode, all decisions are made by the human (none algorithmic assistance is provided). In the advisory control mode, human makes the decisions and an algorithm checks the decisions feasibility. In the interacting control mode, the decision-making process is shared between the human and the algorithms. In the supervisory control, the algorithm is first executed, then necessarily informs the user, deciding to accept or reject the algorithm decisions. Finally, in the automatic control mode, the decisions are completely made by the algorithm (the user is then completely out of the decision-making process).

For problem solving, we think that problem tasks require both scheduler and algorithms participation. Indeed, the complex nature of problem constraints, some of them cannot even be considered on problem modelling phase, demands scheduler participation in order to ensure their satisfaction. On the other hand, problem complexity and the huge amount of computing requirements demands the articulation of transportation technologies (GPS, EDI, GIS) with efficient algorithms in order to obtain satisfactory solutions in a reasonable time. In a real-life context, the full-manual control mode is unreasonable in front of too many items to manage. Indeed, it involves exceeding the cognitive capacities of humans, especially for decision-making under stress. Moreover, a dynamic environment implies painful re-computing. On the other hand, the consideration of the automatic control mode falls out of the scope of our paper focusing on user-oriented algorithms. In that context, we propose a three-phase solving mechanism with different control modes (advisory, interactive, and supervisory) facilitating scheduler participation in decision-making process and integrating efficient algorithms for problem solving and constraint satisfaction.

### 3.2.1 Vehicle selection algorithms

The goal of vehicle selection algorithms is to assist scheduler to select the vehicles serving customers. Let us define  $y_j$  as the variable to define if a vehicle  $j$  is used to serve customers.

$$y_j = \begin{cases} 1 & \text{if vehicle } j \text{ is used to serve customers} \\ 0 & \text{otherwise} \end{cases}$$

#### Advisory control mode: user solution checking

The advisory control mode consists in a user solution checking. It verifies whether a solution proposed by the user is a feasible solution. To this purpose, global capacity constraints (weight, volume, and length) are first verified (Equations 3, 4, and 5). Then, allocation constraints involving vehicles such as vehicle-customer or vehicle-product are checked in order to ensure the presence of a given class of vehicle necessary to satisfy customer demands. Finally, the number of vehicles proposed by the user to serve customers ( $\sum_1^{nv} y_j$ ) is compared with a lower bound of minimal number of vehicles required to solve the problem ( $LB_{nv}$ ). The lower bound is computed from different problem information. Customer and depot time windows and route constraints such as maximal route time, distance or number of customers are therefore considered to compute  $LB_{nv}$ . Further details about lower bound computation are described in Gacias (2010).

$$\sum_{i=1}^{nc} \sum_{k=1}^{np} d_i^k \times P_k^p \leq \sum_{j=1}^{nv} y_j \times C_j^p \quad (3)$$

$$\sum_{i=1}^{nc} \sum_{k=1}^{np} d_i^k \times P_k^v \leq \sum_{j=1}^{nv} y_j \times C_j^v \quad (4)$$

$$\max_{i, d_i^k \neq 0} (P_k^l) \leq \max_{y_j > 0} (C_j^l) \quad (5)$$

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

### Supervisory and interactive control mode: vehicle number minimization

Both control modes focus on vehicle number minimization. In that context, several solutions satisfying capacity and allocation constraints and minimizing the number of vehicles are proposed to the scheduler. Besides, the lower bound  $LB_{nv}$  has to be also respected by these solutions.

The problem can be defined as a linear program as follows:

$$\min \sum_{j=1}^{nv} y_j \quad (6)$$

subject to

$$\sum_{i=1}^{nc} \sum_{k=1}^{np} d_i^k \times P_k^w \leq \sum_{j=1}^{nv} y_j \times C_j^w \quad (7)$$

$$\sum_{i=1}^{nc} \sum_{k=1}^{np} d_i^k \times P_k^v \leq \sum_{j=1}^{nv} y_j \times C_j^v \quad (8)$$

$$\sum_j y_j \geq LB_{nv} \quad (9)$$

$$y_j \in \{0, 1\} \quad (10)$$

An exact and polynomial algorithm to solve the problem when either one of the capacity constraints (weight or volume) are relaxed is proposed. Algorithm 1 illustrates volume constraint relaxation. First, a solution satisfying the weight constraint and minimizing the number of vehicles is proposed. Then, the satisfaction of the lower bound is checked. The algorithm starts selecting the vehicle with the biggest capacity among the available vehicles except for the last vehicle (when the sum of vehicle capacities is enough to satisfy the constraint) where the vehicle with the smallest capacity satisfying the constraint is selected.

No polynomial algorithm exists when all constraints (weight and volume constraints) have to be considered. Actually, the problem is known to be NP-hard (Garey and Johnson, 1979). In that case, the idea is to solve both relaxations separately in order to find a solution satisfying all capacity constraints. Note that when no feasible solution is found after solving both relaxations (on volume and weight constraints) using Algorithm 1, the integer linear problem (6–10) has then to be solved.

Finally, a list of interesting solutions with a minimal number of vehicles is proposed to the scheduler. This set of feasible solutions are found by trying to replace in the solution each

---

**Algorithm 1:** Number of vehicles minimization

---

Step 1: Compute the weight to transport,  $weight \leftarrow \sum_i^{nc} \sum_k^{np} d_i^k \times P_k^w$

**if** capacity weight is bigger than weight to transport,  $\sum_j^{nv} C_j^w \geq weight$ , **then**

- while** ( $weight > 0$ ) **do**
  - Step 2: Select the vehicle  $V_s$  with a maximal weight capacity
  - if**  $C_{V_s}^w < weight$  **then**
    - Step 3: Add vehicle  $V_s$  to the solution
  - else**
    - Step 4: Select vehicle  $V_s$  with a minimal weight capacity but greater than  $weight$
    - Step 5: Add vehicle  $V_s$  to the solution
    - Step 6: Update the weight,  $weight \leftarrow weight - C_{V_s}^w$
- else**
  - Not feasible problem
- if** number of vehicles of the solution is less than  $BI_{nv}$  **then**
  - Step 7: Select a vehicle  $V_s$
  - Step 8: Add vehicle  $V_s$  to the solution

---

vehicle of the original solution by a new vehicle with smaller capacity. Of course, the solution after replacing vehicles has to satisfy the capacity constraints of the problem. The list of solutions proposed is not a complete enumeration; still, a list of interesting solutions is provided to the planner.

The interactive control mode follows the same principle. The vehicle number minimization begins instead from a partial solution proposed by the scheduler. The algorithms propose a set of solutions taking into account the decisions already made by the scheduler.

Again, in case of infeasibility because not enough vehicles are available to satisfy the problem constraints, a constraint relaxation assistance is then proposed (see Section 4).

### 3.2.2 Customer allocation algorithms

The goal here is to determine for each customer of the problem an allocation on a vehicle. The system offers the possibility to manually allocate the customers to vehicles (advisory control mode); in this case the feasibility of each decision is checked by an algorithm. An algorithm for customer allocation is also integrated in solving mechanism. The goal of the algorithm is to propose a complete solution for supervisory control mode and to complete a partial solution proposed by the scheduler for interactive control mode.

#### Advisory control mode: scheduler decision checking

An algorithm checking allocation feasibility is proposed. The idea is to verify after each allocation decision that it exists a feasible solution. The algorithm is divided in two parts: a constraint satisfaction checking and a feasible solution searching. A feasibility test using energetic reasoning (Lopez and Esquirol, 1996) principles is proposed for the first part. In a scheduling context, Gacias et al. (2010a) have proposed an algorithm integrating setup times in the feasibility test. Indeed, travel times between customers in vehicle routing problem can be seen as sequence-dependent setup times. A limited discrepancy search (LDS) algorithm (Harvey and Ginsberg, 1995) is used to find a feasible solution. If no solution is found, the scheduler has to backtrack to her/his previous choices or has to modify her/his last decision. Further details of

the algorithm are described in Gacias (2010).

### Supervisory and interactive control mode: customer allocation algorithm

In the supervisory control mode, a complete solution is proposed by the system. The scheduler can however modify the decisions of the algorithm. The sweep algorithm (Gillet and Miller, 1974) principle is used for customer allocation. First, customers that have to be allocated to a specific vehicle because of allocation constraints are allocated. Then, each customer (selected from the sweep algorithm order) is allocated 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 a customer, then the customer is allocated to a new vehicle.

The interactive control mode uses the same algorithm to complete a solution partially proposed by the scheduler. The scheduler specifies here the allocation for some of the customers and the algorithm ends the customer allocation from the partial solution.

### 3.2.3 Route selection algorithms

The goal of route selection is to determine for each route a sequence of customers. Besides the control modes proposed here to solve the problem, the system also integrates several algorithms for solution optimization.

Figure 4 shows the proposed user interfaces for route selection.

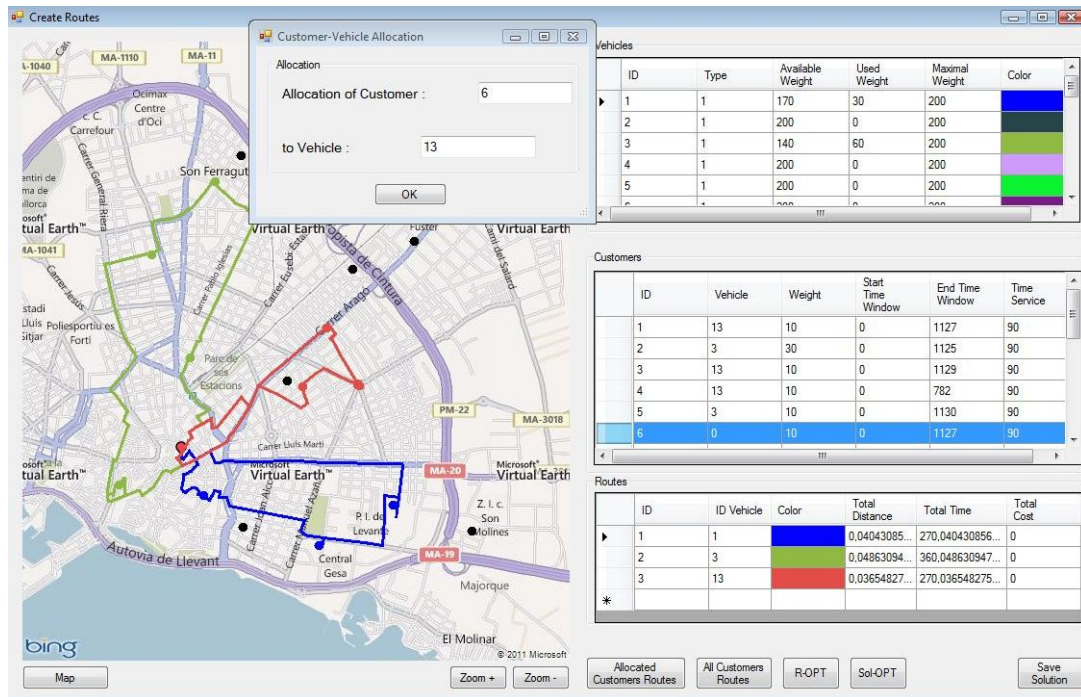


Figure 4: **User interfaces for route selection.** One of the interfaces is a geographical representation allowing the scheduler to visualize routes on a map and displaying relevant information about the problem constraints. User interaction tools are also available facilitating the scheduler participation on problem solving.

### **Advisory control mode: scheduler decision checking**

The scheduler can manually propose a sequence for each vehicle through the human-machine interaction tools. Once again, an algorithm tests the feasibility of the scheduler decisions. First, different dominance rules are tested to verify that the proposed sequence is not inconsistent. After that, if the sequence is correct, a tree search algorithm based on limited discrepancy search is launched to find a feasible sequence with all the customers allocated to the vehicle respecting the subsequence proposed by the scheduler. If no solution is found, the scheduler has to backtrack to her/his previous choices or has to modify her/his last decision.

### **Supervisory and interactive control mode: customer allocation algorithm**

The idea here is to propose a feasible complete solution for the problem. To this purpose, different algorithms are proposed: the first algorithm is based on classical customer insertion techniques traditionally used in the literature and the other two algorithms are proposed for solution optimization. The goal is to propose first a feasible solution with the first algorithm (a solution with a structure respecting schedulers' criteria). The scheduler may then use the optimization algorithms to try to improve the quality of the solution. The use of metaheuristics and constraint programming has already been proposed efficiently to solve the vehicle routing problem (Pesant and Gendreau, 1999; Caseau et al., 1999; Backer et al., 2000).

Once customer allocation is performed, the algorithm uses the principle of *savings* and *regrets* algorithms proposed by Clarke and Wright (1964) and Liu and Shen (1999), respectively. Each customer is inserted in the position of the sequence that minimizes route length. If a customer cannot be inserted in a sequence, a tree search based on LDS is launched in order to find a feasible sequence. The drawback of the savings algorithm is to find a feasible solution when the problem is over-constrained (Kilby et al., 2000). It is worth remarking the importance of having an efficient insertion algorithm, thus each time a customer cannot be inserted, a feasible sequence has to be found and the tree search may be time consuming. We then propose to integrate in the algorithm the regrets criterion favouring first the sequence of the most conflicting customers. The proposed algorithm then sequences the customer following the savings criterion except when it exists a customer with only two or less feasible positions into the sequence; in that case the conflicting customer is sequenced first.

Two solutions optimization algorithms are also proposed. The goal of one of them is to independently optimize the routes of the solution. The second algorithm is a local search algorithm for complete solution optimization. The execution time for each algorithm is specified by the scheduler.

The first algorithm is a Climbing Discrepancy Search (local search method using a limited discrepancy search for neighbourhood exploration)(Milano and Roli, 2002). The goal of the algorithm is to optimize the customer sequence for one route. Customer allocation cannot be therefore modified during the optimization process. The main advantage of the approach is that problem solution may be improved by optimizing only some routes and keeping the same structure for the remaining part. Indeed, in the process of solving problem the scheduler may define a solution structure according to her/his criteria; it is then advisable to propose optimization algorithms disturbing as less as possible the solution structure.

The second algorithm for problem solution optimization is described in Algorithm 2. The idea is to integrate the algorithm for route optimization in a classical local search scheme. For each iteration, a customer reallocation based on geographical criteria is first proposed and then the modified routes are optimized using the CDS algorithm. We define now the reallocation operators used in the algorithm. We find some interesting operators in the literature such as the *Relocation* and the *Exchange* proposed by Savelsbergh (1992) or the CROSS-exchange defined by Taillard et al. (1997). Our goal is to find as soon as possible the best customer allocations. The idea is to focus on geographical criteria to define our neighbourhoods. The first operator

uses the distance between the customers and the gravity centre of the routes to define the new allocations. The distance between a customer and the gravity centre of its route is compared with the distance between the customer and the centroid of the rest of routes. The idea is to determine the customers ( $C_i$ ) nearer of the centroid of another route ( $G_{R_j}$ ) than its own route centroid ( $G_{R_q}$ ). If it is the case, the customer  $C_i$  is then selected to be allocated to route  $R_j$ . The second operator compares for each customer the distance between the customer and the other customers of the problem with the distance between the customer and its own route centroid. If it exists a customer  $C_i$  of route  $R_q$  nearest of the customer  $C_{i'}$  of route  $R_j$  than its own route centroid  $G_{R_q}$ , the customer  $C_i$  is then selected to be allocated to route  $R_j$ . Each new allocation can be defined as  $\{C_i, R_q, R_j\}$ , where  $C_i$  specifies the customer that will be reallocated from route  $R_q$  to route  $R_j$ . We propose then to explore the next neighbourhoods:

- (1) Each new reallocation. The neighbourhood consists in allocating customer  $C_i$  on the vehicle of the route  $R_j$ .
- (2) Each allocation exchange between two customers. If it exists an allocation defined as  $\{C_i, R_q, R_j\}$  and another allocation with this form  $\{C_{i'}, R_j, R_q\}$ , then allocations of both customers are modified at the same time. The customer  $C_i$  is allocated to the vehicle of route  $R_j$  and the customer  $C_{i'}$  is allocated to the vehicle of route  $R_q$ .

It is worth to note our neighbourhoods are sub-neighbourhoods of Relocation and Exchange operators proposed by (Savelsbergh, 1992).

---

**Algorithm 2:** Solution optimization algorithm

---

Step 1: An initial solution  $Sol$  is proposed by scheduler or solving mechanism algorithms

**foreach** route of  $Sol$  **do**

Step 2: Optimize routes independently using the CDS algorithm (authorised discrepancies are limited)

**if** a better solution  $BetterSol$  is found **then**

└ Step 3: Update the solution,  $Sol \leftarrow BetterSol$

**while** execution time defined by the scheduler is not achieved **do**

Step 4: Determine all reallocations  $\{C_i, R_q, R_j\}$  defined for each operator

**foreach** new reallocation defined by neighbourhood (1) and(2) **do**

Step 5: Optimize route  $R_q$  using the CDS algorithm

Step 6: Optimize route  $R_j$  using the CDS algorithm

**if** a better solution  $BetterSol$  is found **then**

└ Step 7: Update solution and go back to Step 4,

└  $Sol \leftarrow BetterSol$

---

### 3.3 Computational results

The efficiency of the solution optimization algorithm (Algorithm 2) is evaluated here. The algorithm was coded in C++ and run on a 2.8 GHz personal computer with 3.8 Go of RAM under the Linux Debian operating system (Version 5.0.7). The algorithm was tested for the well-known instances of the literature (25, 50, and 100 customers) proposed by Solomon (1983) for the VRPTW.

Table 2 shows the results of the comparison between our algorithm (called LS+CDS) and the best-known solutions for distance minimization. The best-known solutions are the optimal solutions for most of the small instances (25 and 50 customers). Computation time is limited

to 120 seconds for small instances (25 and 50 customers) and it is increased to 300 seconds for the instances with 100 customers. The first column *NV* specifies the average number of vehicles, *DIST* defines the average travelled distance, *MeanDev* is the mean deviation to the best-known solutions, *NBest* is the number of instances where our algorithm reaches the best-known solution and finally *TimeBest* indicates the average time to find the best solution.

Instances	Best-Known		LS+CDS				
	<i>NV</i>	<i>DIST</i>	<i>NV</i>	<i>DIST</i>	<i>MeanDev</i>	<i>NBest</i>	<i>TimeBest</i>
<i>C1-25</i>	3	190.59	3	190.59	0.00 %	9 (9)	0.56
<i>R1-25</i>	4.92	463.37	5	474.58	3.70 %	4 (12)	0.58
<i>RC1-25</i>	3.25	350.24	3.25	354.99	9.24 %	7 (8)	0.15
<i>C1-50</i>	5	361.69	5	361.69	0.00 %	9 (9)	15.38
<i>R1-50</i>	7.75	766.13	8.5	805.2	5.24 %	0 (12)	15.03
<i>RC1-50</i>	6.5	730.31	6.87	768.86	7.19 %	3 (8)	2.21
<i>C1-100</i>	10	826.7	10	873.98	7.36 %	4 (9)	49.56
<i>R1-100</i>	13.25	1173.61	14.50	1305.32	11.87 %	0 (12)	184.10
<i>RC1-100</i>	11.12	1341.33	13.37	1410.14	5.39 %	0 (8)	42.67

Table 2: Comparison of solution optimization algorithm with the best-known results

These results show that the algorithm is very efficient for small-size instances. Indeed, in most cases good or even optimal solutions are reached very quickly for the instances with 25 and 50 customers. Besides, the mean deviation with the best-known solutions is also acceptable for all instances. It is clear that the performance of the algorithm decreases with instances size. However, the goal of the algorithm is to propose a good performance to optimize a given solution (sometimes defined by the scheduler). Our algorithm is then hardly dependent of the initial solution contrarily to the most efficient metaheuristics of the literature. Anyway, it is always interesting for robustness to keep as much as possible the structure of the initial solution, specially if the scheduler participates in the design of the solution.

## 4 Model inversion: A new approach for constraint relaxation

One of the weak aspects of decision support systems is the lack of mechanisms proposed to deal with infeasible problems. Obviously, an infeasible problem is unsolvable, that means it does not exist a solution satisfying all the problem constraints. It is then necessary to relax some constraints in order to find a solution. It is important to stress the advisability of involving the scheduler in constraints relaxation process. Indeed, in planning and scheduling context, the scheduler has a wide knowledge of problem constraints and, more importantly, s/he also has the ability to negotiate constraints relaxation with other problem actors (customers, drivers, etc.). In the literature few approaches have been proposed to deal with infeasible problems. On the one hand, human-machine interaction experts propose to focus the research on the way the problem constraints are presented in order to facilitate constraint relaxation process to the scheduler (Higgins, 1996, 2001). On the other hand, Jussien (2001) proposes, in a fully automated context, algorithmic mechanisms based on the concept of “explanation” to determine the constraints to relax to get a feasible problem.

In the next section, an original framework is proposed aiming to support scheduler by offering a list of interesting possibilities for constraint relaxation making the problem feasible.

## 4.1 Model inversion principles

The study is focused on how to manage constraint relaxation when the problem is not satisfiable. The idea is to propose model inversion mechanisms in order to identify the constraints to relax and how these constraints have to be modified. The model inversion consists in the exchange of roles between the decision variables and the problem fixed parameters. In model inversion, decision variables become parameters restricting decision space and parameters become decision variables that can be used to deduce inferences. The main idea is then to determine how problem parameters have to be relaxed in order to get a feasible problem.

Figure 5 shows the proposed user interface for vehicle selection implementing model inversion mechanisms.

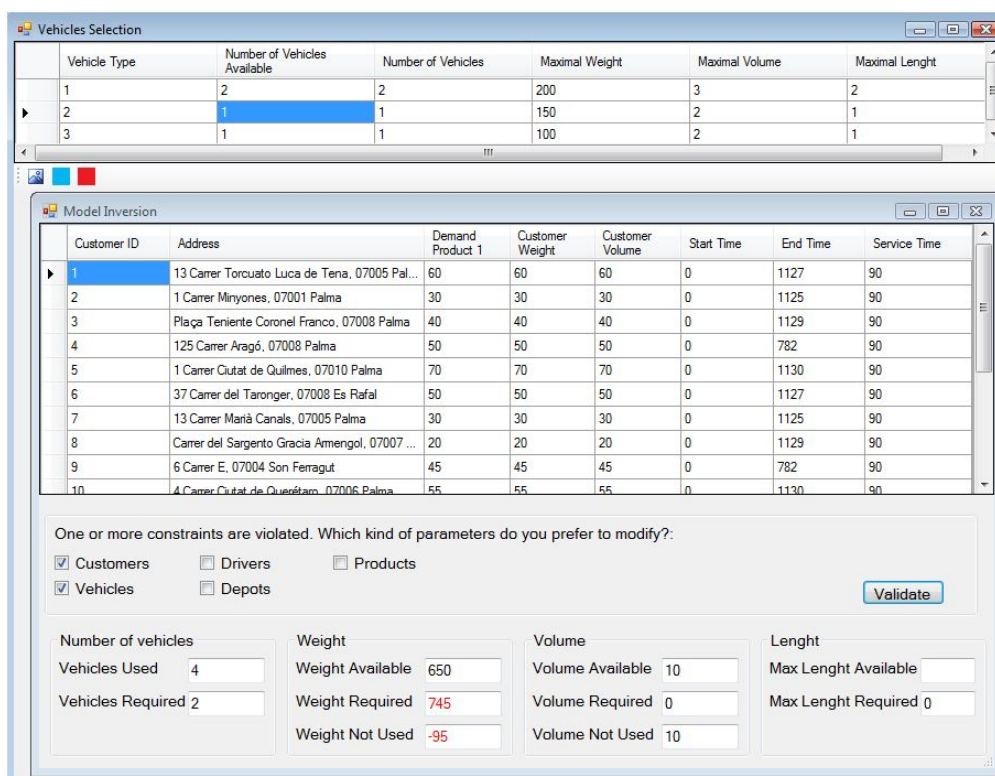


Figure 5: **Model inversion user interface for vehicle selection.** *The scheduler selects the family of parameters authorised to be modified for constraint relaxation when the problem is infeasible (in that case, the weight-capacity constraint is violated).*

The first step for model inversion development is the identification of the parameters integrating the constraints. Once the parameters have been identified, it is then necessary to propose a model inversion mechanism adapted to each parameter for each problem constraint. These mechanisms are launched after a constraint is violated (see Figure 5). First, it is necessary to determine the parameters susceptible to be relaxed. Indeed, there always exist parameters having a more important influence on the violated constraint. In a second phase, an algorithm computes how the parameters have to be relaxed in order to satisfy the violated constraint.

We stress that the complicated part of model inversion framework are the design of efficient

algorithms to deal with a set of parameters. For instance, it is sometimes difficult to select between a set of customers which one has to be postponed for another day or to decide the customer which its forecasted service time has to be modified in order to get a feasible solution. In that context, model inversion techniques based on data classification methods in order to determine the most suitable constraints to relax in priority are proposed. However, these generic algorithms need to be adapted to each problem constraint to increase their efficiency.

## 4.2 Data classification based algorithms

In this section, the generic data classification methods with different constraint adapted criteria used to propose to the scheduler a list of options for constraint relaxation are described.

Data analysis offers a set of methods designed to structure data information in order to identify connections between individuals. The goal of data classification methods is to propose sets of homogeneous individuals. A similarity measure is often used to group individuals. The goal of the measure is to quantify the similarity between two individuals. The groups of individuals are then proposed trying to maximize the similarity between the individuals.

Two data classification methods to group the customers following a geographical and a temporal criterion are considered. When a constraint is violated, an analysis of the groups of clients in order to determine the most suitable constraints to relax is proposed. However, we remark that these methods can be easily extended to other relevant properties of individuals, in our case the customers, in order to adapt the algorithms to each problem constraint.

### 4.2.1 Geographical criterion

We propose the *k-means* algorithm (Forgy, 1965; Lloyd, 1982) to group customers following a geographical criterion. The *k-means* algorithm is used to group the  $nc$  customers into  $K$  clusters ( $P = P_1 \cup P_2 \cup \dots \cup P_K$ ) as homogeneous as possible in relation to a similarity measure defined for each couple of customers. In the context of a geographical criterion, the customer location is selected as similarity criterion. The properties defining customers are then their location  $(x_i, y_i)$ . The number of clusters is the number of vehicles of the solution. Thus each cluster may be seen as a route.

The similarity measure is then the distance between the customers:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (11)$$

The *k-means* algorithm is an iterative algorithm (see Algorithm 3). First, the  $K$  means are created. The initial means are uniformly spread along the longest axis of the problem taking the centroid of the customers as the centre point. Then, at each iteration the distance between the customers and the means is computed. Each customer is allocated to the cluster with the nearest mean. Finally, the  $K$  means are updated as the centroid of the new clusters. The algorithm stops when no more changes between the clusters of two successive iterations are observed.

---

#### **Algorithm 3:** *k-means* algorithm for geographical customer classification

---

Step 1: Select a set of  $K$  means

**repeat**

    Step 2: Compute distances between each customer and the means (Equation 11)

    Step 3: Allocate each customer to the cluster with the nearest mean

    Step 4: Update the means of each cluster  $P_k$

**until** *no changes are observed between the clusters of two successive iterations;*

---

Once the sets of customers are performed, an analysis of the clusters in order to determine the

best constraints to relax is performed. This analysis has however to be adapted to the constraint being violated.

#### Case study: Postpone a customer delivery

We study here the case when a customer delivery has to be postponed because the problem constraints cannot be satisfied. In the geographical criterion context, the customers being part of the same cluster have a high probability to be served by the same vehicle. In that case, a parameter allowing to identify the less homogeneous customers for the clusters is proposed. This parameter ( $dm_i$ ) is calculated using the mean distance between the customer and the other customers of its cluster and the distance between the customer and the nearest depot in order to penalize the customers not located close of the depots.

$$dm_i = \frac{\min_{l \in Depot} d_{il} + \sum_{j \in P_k} d_{ij}}{|P_k|} \quad (12)$$

The customer can be then classed according to this parameter. A priority list of customers to postpone based on geographical information can be proposed to the scheduler .

#### 4.2.2 Temporal criterion

A temporal-based customer classification is also proposed. In that case, the clustering approach uses the Dynamic Cluster Algorithm (DCA) introduced by Diday (1971). DCA is an extension of  $k$ -means algorithm. The similarities here can no longer be shown as an Euclidean distance; an allocation function (a dissimilarity measure) and also a new way to represent the clusters (the centroid does not exist anymore) need to be defined. A dissimilarity measure based on customer time windows (Levy, 1996) is proposed. The parameter measures the degree of centring between two time windows. Figure 6 displays the main relations between two time windows and the value of the dissimilarity measure ( $\delta$ ) for each configuration.

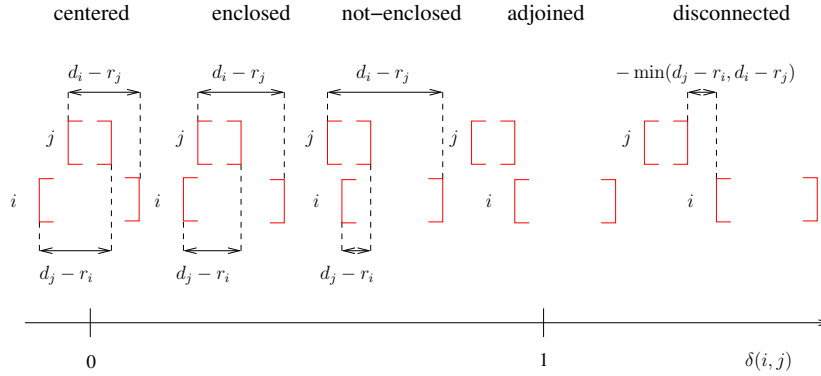


Figure 6: Main configurations between two time windows

The dissimilarity measure is defined as follows:

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

An allocation and a representation function are also necessary. 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  $[\min_{j \in P_k} r_j, \max_{j \in P_k} r_j]$ . At each iteration, a customer  $C_i$  is allocated to the cluster  $P_k$  with a core  $C_k$  minimizing  $\delta$ . The algorithm is very similar to  $k$ -means algorithm (Algorithm 3); at each iteration the distances between the customer and the core of the  $K$  clusters ( $\delta(C_k, C_i), \forall k = 1..K$ ) are computed for each customer, each customer is then allocated to the cluster with a minimal distance (minimize  $\delta$ ) and finally the core of the new clusters are updated.

To initialize the cores, Algorithm 4 is proposed. The cores are uniformly distributed all along the release times scale.

---

**Algorithm 4:** Core initialisation for temporal-based classification

---

Step 1: Compute temporal horizon  $TH$ ,  $TH \leftarrow \max_{i=1..nc} r_i - \min_{i=1..nc} r_i$

Step 2: Compute  $step$ ,  $step \leftarrow \frac{TH}{K}$

Step 3: Compute rough time for first core,  $t_1 \leftarrow \min_{i=1..nc} r_i + \frac{step}{2}$

Step 4: Determine the core of the first cluster  $C_1$ ,  $C_1 \leftarrow$  customer  $i$  with  $r_i$  nearest to  $t_1$

Step 5: Initialise cluster counter,  $k \leftarrow 2$

**for** ( $k \leq K$ ) **do**

    Step 6: Compute rough time for cluster  $k$ ,  $t_k \leftarrow t_k + step$

    Step 7: Determine the core of cluster  $k$ ,  $C_k \leftarrow$  customer  $i$  with  $r_i$  nearest to  $t_k$  and the customer is not already a core of a cluster

    Step 8: Increase cluster counter,  $k \leftarrow k + 1$

---

**Case study: Postpone a customer delivery**

The dissimilarity measure groups together customers with time windows interaction. Once the groups of customers are performed, the clusters are analysed in order to identify the most conflicting customers. For each cluster, a parameter called critical index ( $CI_k$ ) is defined. This parameter is computed as the ratio between the number of customers of the cluster and the number of vehicles. The critical index gives an idea of how critical the cluster is. Indeed, a big value for the  $CI_k$  means that the planner may have some troubles to serve the customers of the cluster because of customers time windows incompatibilities. The  $CI_k$  is then used as a parameter to point the customers to postpone in case of infeasibility.

---

**Algorithm 5:** Identification of a customer to postpone the delivery for a temporal-based criterion

---

Step 1: Core initialization for temporal-based classification (Algorithm 4)

**repeat**

    Step 2: Compute  $\delta$  between each customer and the  $K$  cores

    Step 3: Allocate each customer on a cluster which its core minimizes  $\delta$

    Step 4: Update cores of each cluster  $P_k$

**until** *no changes are observed between two successive iterations;*

Step 5: Identify clusters of conflicting customers ( $CI_k > limit$ )

**for** *each conflicting cluster* **do**

    Step 6: Launch algorithm for geographical classification (Algorithm 3)

    Step 7: Save into a list  $L$  the customer with the most important  $dm_i$

Step 8: Propose the customer of the list  $L$  with the biggest  $dm_i$  as the first customer to postpone

---

We observe that travel time between customers is not considered in temporal-based classification. To cover this lack, both classification criteria (geographical and temporal) are mixed in order to increase efficiency. In that case, the idea is first to target a set of critical customers using the  $CI_k$  index and then the  $k$ -means algorithm is used to select a customer between the critical customers. Algorithm 5 describes the mechanism to propose a priority list of customers to be postponed based on temporal information.

### 4.3 Computational results

The algorithms based on data classification methods proposed to select the best customer to postpone the delivery are evaluated in this section. These algorithms are first compared on a set of small-size instances (9 customers) where a complete enumeration of all feasible solutions can be performed in a reasonable time. The number of feasible solutions is used to evaluate the flexibility provided by the criteria. Indeed, a larger number of feasible solutions widen planners' degrees of freedom facilitating the construction of real-world adapted solutions. The instances of the second set are more realistic instances with 25 customers. Model inversion criteria have been compared considering a geographical optimization function: travel distance minimization.

The small-size instances (9 customers) of the capacitated vehicle routing problem with customer time windows (CVRPTW) are optimally solved. Indeed, each instance is solved nine times suppressing each time one customer from the set of customers. The results obtained taking out of the problem the customer for which the criteria give the priority are compared: the *GDC* criterion (mean distance between customers of the same cluster after the geographical-based clustering), the *GDD* criterion (largest distance with the nearest depot), and the *TDC* criterion (mean distance between customers of the same cluster after the temporal-based clustering).

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

In Table 3, the first column *NbOptDist* specifies the number of instances the optimal solution for distance minimization is reached when the problem is solved without the customer proposed by the criterion. The total number of instances is put in brackets. The second column (*AvgDev*) specifies the average deviation from the optimal solution for the instances where the optimal solution is reached when the suppressed customer is not the customer selected by the criterion. *AvgPos* indicates the average position of the solution when the solutions are sorted 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 providing a bigger number of feasible solutions.

These results show that *GDC* is the most efficient criterion for small instances: it reaches the best suitable 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 best suitable decision, the proposition of the criterion keeps being a good proposition. The *GDC* and *GDD* criteria outperform the *TDC* criterion, which is not a surprise because the objective (distance minimization) used to evaluate the algorithms is a geographical-based function such as *GDC* and *GDD*, contrarily to *TDC* which is a temporal-based criterion. The last line shows that all criteria together are really efficient. Indeed, the most suitable choice is selected for 37 over 56 instances by one of the three criteria when distance minimization is considered. It is interesting to note that following the advices of the criteria generally (38 over 56 instances) leads to a problem accepting a greater number of feasible solutions, thus increasing planners' opportunities for their behaviour in order to find better real-world suited solutions.

Table 4 displays the results for the instances with 25 customers. A complete enumeration

56 instances				
$nc = 9$	$NbOptDist$	$AvgDev$	$AvgPos$	$NbSol$
<i>GDC-C</i>	13 (17)	8.73 %	2.7	6
<i>GDC-R</i>	9 (23)	5.99 %	3.5	17
<i>GDC-RC</i>	9 (16)	5.01 %	3.2	5
<i>TDC-C</i>	5 (17)	14.98 %	4.5	7
<i>TDC-R</i>	3 (23)	11.46 %	5.4	7
<i>TDC-RC</i>	4 (16)	9.30 %	5.3	7
<i>GDD-C</i>	8 (17)	15.76 %	4.3	8
<i>GDD-R</i>	5 (23)	10.97 %	4.7	11
<i>GDD-RC</i>	3 (16)	9.12 %	4.5	4
<i>All criteria</i>	37	5.81 %		38

Table 3: Results for distance optimization and the number of feasible solutions

of solutions in a reasonable time is not possible here, the instances have been therefore solved using Algorithm 2. The last column of Table 3 calculated from the complete enumeration of feasible solutions has no longer any sense in Table 4. For this reason, the column  $NbSol$  has been replaced by  $Nb5Best$  indicating, in that case, the number of instances the decision proposed by the criterion is among the five best decisions considering distance optimization.

56 instances				
$nc = 25$	$NbOptDist$	$AvgDev$	$AvgPos$	$Nb5Best$
<i>GDC-C</i>	6 (17)	2.74 %	6.55	9
<i>GDC-R</i>	4 (23)	4.48 %	6.84	15
<i>GDC-RC</i>	3 (16)	11.35 %	9.85	6
<i>TDC-C</i>	1 (17)	7.88 %	7.69	9
<i>TDC-R</i>	1 (23)	5.70 %	9.05	6
<i>TDC-RC</i>	0 (16)	10.81 %	11.31	3
<i>GDD-C</i>	0 (17)	10.43 %	13.94	0
<i>GDD-R</i>	6 (23)	5.36 %	8.76	13
<i>GDD-RC</i>	5 (16)	8.67 %	7.36	10
<i>All criteria</i>	17 (56)	4.68 %		42

Table 4: Results for distance optimization

The results are very similar than for the instances with only 9 customers. The *GDC* criterion proves to be the most efficient criterion, especially for instances with clustered and randomized customers, type C and type R, respectively. Nevertheless, these results show that *GDD* criterion is the best criterion for the RC instances. It can be due to a possible loss of efficiency of the *k*-means algorithm for this type of problems (the clusters of customers after the *k*-means algorithm do not correspond exactly with the vehicle routes determined by the problem-solving algorithm). However, it is worth remarking that the efficiency of all criteria considered together is really good. Indeed, for most of instances (42 over 56) there is a criterion that proposes a decision reaching one of the five best solutions for distance minimization and the average deviation is less than 5 % meaning that the criteria complement each other.

## 5 Conclusions

In this paper, a decision support system for transportation scheduling has been proposed. An interdisciplinary approach has been followed for the system design; a link has been done between problem solving methods (operations research techniques and data classification algorithms) and cognitive engineering techniques (interfaces design, decision sharing between human and machine). The paper focuses on the solving mechanism and scheduler-oriented algorithms integrating the decision support system.

First, a work domain analysis using an abstraction hierarchy for the vehicle routing problem has been described. A new approach is proposed, based on the study of the different variants of the problem detailed in the literature. This approach facilitates to reach completeness in domain constraints identification at the same time that the abstraction hierarchy remains event- and device-independent. The main advantage of the approach is that the problem representation after the analysis does not restrict strategies for scheduler behaviour. The domain constraints revealed by the analysis make up the reference system shared by the human and the machine. In that context, a decision support system architecture has been then defined around the work domain analysis.

The solving mechanism designed to assist the scheduler to perform each problem subtask has been presented. To this purpose, different solving control modes seeking to facilitate scheduler participation in problem solving processes have been proposed. The idea behind control modes, contrarily to a fully automated approach, is that system, through user-interaction tools, gives the schedulers the possibility to built solutions according to their preferences, letting to take into consideration new constraints and/or occasionally violating some of the constraints in order to find better real-world adapted solutions. In that context, a set of useful user-oriented algorithms to assist the scheduler to solve the problem have been proposed and efficiently tested on instances of the literature.

Finally, an original framework aiming at supporting problem constraints relaxation when this becomes infeasible has been proposed. Constraint relaxation support is a topic barely discussed in the literature. In this paper, the principles of a methodology for constraint relaxation seeking to support the scheduler by proposing a list of interesting possibilities for action have been defined. In that context, model inversion mechanisms using data classification algorithms have been proposed and efficiently tested on instances of the literature. The results show the interest of such a Human-Machine cooperation in terms of number of feasible solutions obtained and their quality. Besides, it is worth remarking that the proposed model inversion mechanisms are generic enough to be used on different kinds of problems others than vehicle routing. Indeed, spatial and temporal decomposition methods suit and can be easily adapted for planning and scheduling problems.

## Acknowledgments

The authors are indebted to the anonymous referees for their numerous constructive remarks.

This research was initiated while the first author was with LAAS-CNRS, Université de Toulouse, France.

## References

- B. De Backer, V. Furnon, P. Kilby, P. Prosser, and P. Shaw. Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics. *Journal of Heuristics*, 6:501–523, 2000.

- C. Basnet, L. Foulds, and M. Igbaria. FleetManager: a microcomputer-based decision support system for vehicle routing. *Decision Support systems*, 16(3):195–207, 1996.
- C. M. Burns. Navigation strategies with ecological displays. *International Journal of Human-Computer Studies*, 1(52):111–129, 2000.
- Y. Caseau, F. Laburthe, and G. Silverstein. A meta-heuristic factory for vehicle routing problems (meta-programming for meta-heuristics). In J. Jaffar, editor, *Lecture Notes in Computer Science*, volume 1713, pages 144–158. Springer, 1999. Principles and Practice of Constraint Programming (CP’99).
- J. Cegarra. A cognitive typology of scheduling situations: A contribution to laboratory and field studies. *Theoretical Issues in Ergonomics Science*, 9(3):201–222, 2008.
- J. Cegarra and W. van Wezel. A comparison of task analysis methods for planning and scheduling. In Jan C. Fransoo, Toni Waefer, and John R. Wilson, editors, *Behavioral Operations in Planning and Scheduling*. Springer, 2010.
- J. Cegarra, B. Gacias, and P. Lopez. Implications of technological changes in vehicle routing systems for planners’ constraint processing. *Accepted in Human Factors and Ergonomics in Manufacturing & Service Industries*, 2011.
- G. Clarke and J. V. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.
- E. Diday. Une nouvelle méthode en classification automatique et reconnaissance des formes : la méthode des nuées dynamiques. *Revue de Statistique Appliquée*, 19(2):19–33, 1971.
- S. B. Eom and Y. B. Kim. Survey of Decision Support System Applications (1995-2001). *Journal of the Operational Research Society*, 57(11):1264–1278, 2006.
- S. B. Eom and S. M. Lee. DSS Applications Development Research: Leading Institutions and Most Frequent Contributors (1971-April 1988). *Decision Support Systems*, 6(3):269–275, 1990.
- S. B. Eom, S. M. Lee, E. B. Kim, and C. Somarajan. A Survey of Decision Support System Applications (1988-1994). *Journal of The Operational Research Society*, 49(2):109–120, 1998.
- E. W. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. *Biometrics*, 21(3):768, 1965.
- B. Gacias. *Une approche interdisciplinaire pour l’ordonnement des transports*. PhD thesis, Université de Toulouse, 2010.
- B. Gacias, C. Artigues, and P. Lopez. Parallel machine scheduling with precedence constraints and setup times. *Computers & Operations Research*, 37(12):2141–2151, 2010a.
- B. Gacias, J. Cegarra, and P. Lopez. Work domain analysis and ecological interface for the vehicle routing problem. In *11th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Human-Machine System*, Valenciennes (France), 2010b.
- M. R. Garey and D. S. Johnson, editors. *Computers and intractability: A guide to the theory of NP-Completeness*. Freeman & Co, New-York (USA), 1979.
- B. E. Gillet and L. R. Miller. A heuristic algorithm for the vehicle dispatch problem. *Operations Research*, 22(2):340–349, 1974.

- W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, 1995.
- P. G. Higgins. Interaction in hybrid intelligent scheduling. *International Journal of Human Factors in Manufacturing*, 6:185–203, 1996.
- P. G. Higgins. Architecture and interface aspects of scheduling decision support. In *Human performance in planning and scheduling: fieldwork studies, methodologies and research issues*, pages 245–279. Taylor and Francis, London, 2001.
- J. Itoh, A. Sakuma, and K. Monta. An ecological interface for supervisory control of BWR nuclear power plants. *Control Engineering Practice*, 3:231–239, 1995.
- S. Jackson, J. R. Wilson, and B. L. MacCarthy. A new model of scheduling in manufacturing: Tasks, roles, and monitoring. *Human Factors*, 46:533–550, 2004.
- N. Jussien. e-constraints: explanation-based constraint programming. In *Workshop on User-Interaction in Constraint Satisfaction (CP'01)*, Paphos (Cyprus), 2001.
- P. Kilby, P. Prosser, and P. Shaw. A comparison of traditional and constraint-based heuristic methods on vehicle routing problems with side constraints. *Constraints*, 5(4):389–414, 2000.
- B. Kim, S. Kim, and S. Sahoo. Waste collection vehicle routing problem with time windows. *Computers & Operations Research*, 33(12):3624–3642, 2006.
- M.-L. Levy. *Méthodes par décomposition temporelle et problèmes d'ordonnancement*. PhD thesis, Institut National Polytechnique de Toulouse, 1996.
- F.-H. F. Liu and S.-Y. Shen. A route-neighborhood-based metaheuristic for vehicle routing problem with time windows. *European Journal of Operational Research*, 118(3):485–504, 1999.
- S. P. Lloyd. Least square quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.
- P. Lopez and P. Esquirol. Consistency enforcing in scheduling: A general formulation based on energetic reasoning. In *5th International Workshop on Project Management and Scheduling (PMS'96)*, pages 155–158, Poznan (Poland), 1996.
- N. F. Matsatsinis. Towards a decision support system for the ready concrete distribution system: A case of a Greek company. *European Journal of Operational Research*, 152(2):487–499, 2004.
- J. E. Mendoza, A. L. Medaglia, and N. Velasco. An evolutionary-based decision support system for vehicle routing: The case of a public utility. *Decision Support Systems*, 46(3):730–742, 2009.
- M. Milano and A. Roli. On the relation between complete and incomplete search: an informal discussion. In *Fourth International Workshop on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CP-AI-OR'02)*, Le Croisic (France), 2002.
- N. Naikar, R. Hopcroft, and A. Moylan. Work domain analysis: Theoretical concepts and methodology. Technical report, Australian Government, Department of Defence, Defence Science, 2005.

- G. Pesant and M. Gendreau. A constraint programming framework for local search methods. *Journal of Heuristics*, 5:255–279, 1999.
- D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007.
- M. Rahimi and M. Dessouky. A hierarchical task model for dispatching in computer-assisted demand-responsive paratransit operation. *ITS Journal*, 6:199–223, 2001.
- J. Rasmussen, A. M. Pejtersen, and L. P. Goodstein. *Cognitive Systems Engineering*. Wiley, New York, 1994.
- R. Ruiz, C. Maroto, and J. Alcaraz. A Decision Support System for a Real Vehicle Routing Problem. *European Journal of Operational research*, 153(3):593–606, 2004.
- P. M. Sanderson. The human planning and scheduling role in advanced manufacturing systems: an emerging human factors domain. *Human Factors*, 31:635–666, 1989.
- L. Santos, J. Coutinho-Rodrigues, and J. R. Current. Implementing a multi-vehicle multi-route spatial decision support system for efficient trash collection in Portugal. *Transportation Research*, 42 (Part A):922–934, 2008.
- M. W. P. Savelsbergh. The vehicle routing problem with time windows: Minimizing route duration. *INFORMS Journal on Computing*, 4(2):146–154, 1992.
- M. M. Solomon. *Vehicle routing and scheduling with time window constraints: Models and algorithms*. PhD thesis, University of Pennsylvania, USA, 1983.
- E. Taillard, P. Badeau, M. Gendreau, F. Guertin, and J.-Y. Potvin. A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31(2):170–186, 1997.
- P. Toth and D. Vigo, editors. *The vehicle routing problem*. Society for Industrial and Applied Mathematics, Philadelphia (USA), 2001.
- R. van Paassen. New visualisation techniques for industrial process control. In *6th IFAC/IFIP/IFORS/IEA Symposium on Analysis, Design and Evaluation of Man-Machine Systems*, pages 457–462, Cambridge, MA (USA), 1995. Pergamon Press, Oxford.
- W. van Wezel, J. Cegarra, and J.-M. Hoc. Allocating function to humans and algorithms in scheduling. In Jan C. Fransoo, Toni Waefer, and John R. Wilson, editors, *Behavioral Operations in Planning and Scheduling*. Springer, 2010.
- K. J. Vicente. *Cognitive Work Analysis: Toward safe, productive, and healthy computer-based work*. Lawrence Erlbaum Associates, Inc., Mahwah, NJ (USA), 1999.
- K. J. Vicente. Work domain analysis and task analysis: A difference that matters. In J. M. Schraagen, S. F. Chipman, and V. L. Shalin, editors, *Cognitive task analysis*, pages 101–118. Mahwah, NJ: Lawrence Erlbaum Associates, 2000.
- B. L. W. Wong and A. Blandford. Analysing ambulance dispatcher decision making: Trialing emergent themes analysis. In F. Vetere, L. Johnston, and R. Kushinsky, editors, *Human Factors 2002, the Joint Conference of the Computer Human Interaction Special Interest Group and The Ergonomics Society of Australia, HF2002 (CD-ROM publication)*, Melbourne (Australia), 2002.