

# Time dependent vehicle routing problem with a multi ant colony system

Alberto V. Donati \*, Roberto Montemanni, Norman Casagrande,  
Andrea E. Rizzoli, Luca M. Gambardella

*Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Galleria 2, 6928 Manno, Switzerland*

Received 1 June 2005; accepted 1 June 2006

Available online 18 October 2006

## Abstract

The Time Dependent Vehicle Routing Problem (TDVRP) consists in optimally routing a fleet of vehicles of fixed capacity when travel times are time dependent, in the sense that the time employed to traverse each given arc, depends on the time of the day the travel starts from its originating node. The optimization method consists in finding solutions that minimize two hierarchical objectives: the number of tours and the total travel time.

Optimization of total travel time is a continuous optimization problem that in our approach is solved by discretizing the time space in a suitable number of subspaces. New time dependent local search procedures are also introduced, as well as conditions that guarantee that feasible moves are sought for in constant time.

This variant of the classic Vehicle Routing Problem is motivated by the fact that in urban contexts variable traffic conditions play an essential role and can not be ignored in order to perform a realistic optimization. In this paper it is shown that when dealing with time constraints, like hard delivery time windows for customers, the known solutions for the classic case become unfeasible and the degree of unfeasibility increases with the variability of traffic conditions, while if no hard time constraints are present, the classic solutions become suboptimal.

Finally an application of the model to a real case is presented. The model is integrated with a robust shortest path algorithm to compute time dependent paths between each customer pairs of the time dependent model.

© 2006 Elsevier B.V. All rights reserved.

**Keywords:** Vehicle routing; Time dependent; Discretization; Ant colony system

## 1. Introduction

The Vehicle Routing Problem (VRP) has been largely studied because of the importance of mobility in logistic and supply-chains management that

relies on road network distribution. Many different variants of this problem have been formulated to provide a suitable application to a variety of real-world cases, with the development of advanced logistic systems and optimization tools. The features that characterize the different variants aim on one hand to take into account the constraints and details of the problem, while on the other to include different aspects of its nature, like its dynamicity,

\* Corresponding author.

E-mail address: [albertodonati@inwind.it](mailto:albertodonati@inwind.it) (A.V. Donati).

time dependency and/or stochastic aspects. The richness and difficulty of this type of problem, has made the vehicle routing an area of intense investigation.

In this paper we focus on the presence of variable traffic conditions on real road networks, like in urban environments, where these conditions can greatly affect the outcomes of the planned schedule. Accounting for variable travel times is particularly relevant when planning in presence of time constraints, such as delivery time windows. Solutions obtained without considering this variability will result in sub-optimality or unfeasibility with respect to these constraints, as it will be shown in the experimental results section.

This study is also motivated by the recent developments of real time traffic data acquisition systems. With access to these data, it is possible to include in the model dynamic and updated information, and obtain realistic and improved solutions.

The paper is organized as follow: problem formulation and review of the time dependent models; the Multi Ant Colony System is introduced for the classic VRP, and its extension to the time dependent case; the formulation of new time dependent local search procedures and related issues and discussion of issues related to the time dependency; the remainder of the paper is dedicated to computational results and its applications to a real world situation, with the use of real traffic data and integration with a Robust Shortest Path algorithm [1] to deal with realistic graphs representing the urban road network.

## 2. Problem description

In the classic VRP with hard time windows, VRPTW, a fleet of vehicles of uniform capacity is scheduled to visit the given set of  $N$  customers,  $c_i$ , each characterized by a demand  $q_i$ , a time window  $tw_i = [b_i, e_i]$ , and a service time  $s_i$ , with routes originating and ending at a depot, whose opening and closing time  $[t_c, t_c]$  is specified, and a fleet of trucks of uniform capacity  $C$  is available. Each delivery can be done no later than the ending time of the customer's time window, while if the arrival time at the customer's location is before the beginning of the customer's time window, the delivery has to wait until the beginning of the time window. The service time, the time necessary to complete the delivery, must have elapsed before it is possible to leave the location for the next delivery. Other assumptions

of the problem are: (1) the quantity requested by the customer is to be delivered in a single issue and in full; (2) all tours must originate and end at the depot, within the depot opening time; (3) the total quantity delivered in each tour cannot exceed the truck capacity  $C$ .

The problem is represented with a directed graph  $G(V, A)$ , where  $V$  is the set of nodes, representing the customers and the depot, and characterized by a geographical location, and  $A$  is the set of oriented arcs connecting pairs of nodes, and representing the roads as straight connections between nodes. A more complex representation will be considered in Section 7.

Traditionally, the optimization algorithm finds first the solution that minimizes the number of tours, and then minimizes the total length, which is given by the sum of the lengths of all tours. In this case the total length coincides with the total traveling time. A slightly modified problem defines for each arc a constant traveling speed, so a more accurate model is obtained, and the total traveling time (instead of the total length) is used as the minimization objective. This is sometimes referred to as the Constant Speed Model.

## 3. Review of time dependent VRP models

The presence of diversified conditions of traffic at different times of the day were first taken into account by Malandraki and Daskin in [2] (for the VRP as well as for the TSP). On each arc a step-function distribution of the travel time was introduced. A mixed integer programming approach and a nearest neighbor heuristic were used in the optimization.

Another approach to the time dependent VRP is presented by Ichoua et al. in [3], where the customers are characterized by soft time windows, that is, if the arrival time at a customer is later than the end of the time window, the cost function (the total travel time here) will be penalized by some amount. The optimization is done with a tabu search heuristic, and it is based on the use of an approximation function to evaluate in constant time the goodness of local search moves. The model is also formulated for a dynamic environment, where not all service requests are known before the start of the optimization. A direct comparison with the model presented in [3] is not possible, because of two main differences in the models: (1) capacity constraints for the trucks are not considered, and (2) the customers

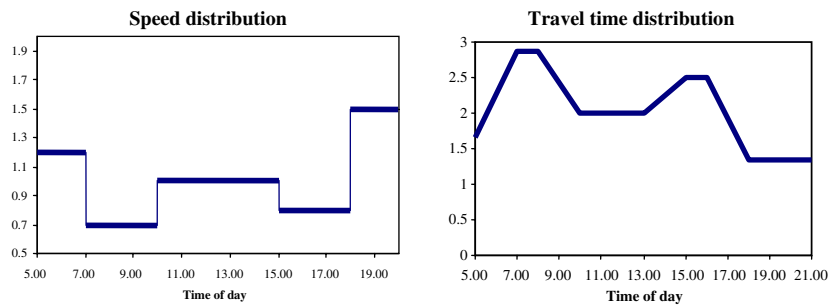


Fig. 1. (a) Example of speed distribution; (b) travel time distribution induced by the speed distribution (arc length of 2).

time windows are used as a soft constraint. As a consequence of this, in [3] there is no need of an optimization with respect to the number of tours (which is supposed to be known *a priori*), and no unfeasible solutions are found. In the TDVRP model, the number of tours is an objective of the optimization, and because of the constraint of the time windows, unfeasible solutions can also be found.

Nevertheless in [3] the First In, First Out (FIFO) principle is introduced: if two vehicles leave from the same location for the same destination traveling on the same path, the one that leaves first will always arrive first, no matter how speed changes on the arcs during the travel. This principle is important not only because it prevents some inconsistencies (e.g. a vehicle could wait at some location for the time when speeds are higher and then arrive at the desired location before another vehicle who had left before), but also because, as it will be shown later in this paper, it allows to keep linear the time required to check for the feasibility of local search moves, as in the constant speed/classic VRP case (discussion in Section 6).

The FIFO principle is guaranteed by using a step function for the speed distribution, from which the travel times are then calculated, instead of a step function for the travel time distribution.

A typical speed distribution is shown in Fig. 1a. In this way, the traveling time distribution (shown in Fig. 1b) deriving by the speed distribution is continuous, and since the distance between two nodes is fixed, it only depends on the time of the day when the travel starts.

#### 4. Ant colony optimization

Ant Colony Optimization (ACO) was introduced by Dorigo et al. in [4], and it is based on the idea

that a large number of simple artificial agents are able to build solutions via low-level based communication, inspired by the collaborative behavior of ant colonies. A variety of ACO algorithms has been proposed for discrete optimization, as discussed in [5], and have been successfully applied to the traveling salesman problem, symmetric and asymmetric [4,6,7,12] the quadratic assignment problem [14], graph-coloring problem [11], job-shop/flow-shop [10], sequential ordering [13], vehicle routing [8,9,15].

ACO can be applied to optimization problems based on graphs. The basic idea is to use a positive feedback mechanism to reinforce those arcs of the graph that belong to a good solution. This mechanism is implemented associating pheromone levels with each arc, which are then updated proportionally to the goodness of the solutions found. In such a way the pheromone levels encode locally the global information on the solution. Each artificial ant will then use this information weighted with an appropriate local heuristic function (e.g. for the TSP, the inverse of the distance) during the construction of a solution.

As showed by Dorigo and Gambardella in [12], this analogy has suggested a more elaborate and efficient computational paradigm, called the Ant Colony System (ACS), which differentiates from the previous for three aspects: (1) it enhances exploration around good solutions (local pheromone update); (2) it focuses on the found good solutions, with global pheromone update only on the arcs belonging to the best solution found so far; and (3) it implements a new state transition rule based on the choice of the edge with a pseudo-random proportional rule, which tends to reduce randomness. This method has been applied in [12] to the symmetric and asymmetric traveling salesman problem, and the results showed that this method is

among the best metaheuristics, especially when combined with specialized local search procedures.

Solving the VRP is known to be a combinatorial NP-hard optimization problem. When an exact approach exists, it often requires large computational times [16], and is not viable in the time scale of hours, usually the time scale required by distribution planners. With the development of real-time data acquisition systems, and the consideration of various dynamic aspects, it appears more and more advisable to find high quality solutions to updated information in sensibly shorter times.

## 5. The time dependent MACS-VRPTW

It has been shown by Gambardella et al. in [15], that ACO can be used to solve the VRP with hard time windows constraints (VRPTW). This approach consists in using the algorithm called Multi Ants Colony System (MACS-VRPTW) with a hierarchy of two artificial ant colonies, each one dealing with one of the objectives of the optimization: the first colony is named ACS-VEI and deals with tour minimization while ACS-TIME minimizes distance. The two colonies co-operate by exchanging information through pheromone updating. The MACS-VRPTW algorithm coordinates the activities of two colonies which simultaneously look for an improved and feasible solution, that is: (1) a solution that has a smaller number of tours; (2) it has the same number of tours and a shorter length. When a new best solution is found it is then used to perform a global pheromone update, so that both colonies can make use of the updated information about the performance of the new solution.

The results presented in [15] show that this method is comparable with the best known methods, in terms of computation time and quality of the solutions found.

In the following Sections 5.1–5.6 we recall the procedures used in the MACS-VRPTW presented in [15] for self-reference. The reader familiar with this model can go directly to Section 5.6 on Time dependent MACS-VRPTW (page 10).

### 5.1. Ant constructive procedure

Each ant of the colony attempts to complete a solution using the following constructive procedure until all the customers are serviced.

The ant moves from a node  $i$  (a customer or a depot) to the next  $j$  (a customer or a depot – a depot

only if  $i$  is a customer) by choosing among the feasible  $j$ s that have not been visited yet (except for the depot) and that do not violate any of the constraints of the problem (set  $J$ ), with the following probability distribution:

$$p(j) = \tau_{ij} \cdot h_{ij}, \quad j \in J \quad (1)$$

where  $\tau_{ij}$  are the pheromones on edge  $(i, j)$  and  $h_{ij}$  is the local heuristic function:

$$h_{ij} = \frac{1}{\max(1, (d_{ij} + wt_j) \cdot (e_j - t_a) - IN_j)} \quad (2)$$

where  $d_{ij}$  is the distance from  $i$ ,  $wt_j$  is waiting time at  $j$ , and  $e_j - t_a$  is the difference between the arrival time at  $j$  and the corresponding end of the time window. The term  $IN_j$  represents a bias factor, the number of times that a customer has not been included in a solution, and increases the probability of a customer of being included in a later solution. Also note that  $j$  can be a depot; in this case the tour is closed even if the truck has still some quantity left.

#### 5.1.1. Constraints

The next location  $j$  is considered a possible choice, if it satisfies all of the following constraints:

1. the arrival time at  $j$ ,  $t_a \leq e_j$  customer's time window;
2. the quantity left on the truck,  $q_j \leq Q_{\text{left}}$ ;
3. returning time at the depot from  $j$ , once the work is completed at  $j$ , cannot be greater than the depot closing time.

An ant uses the probability given by Eq. (1) in two ways, determined by a fixed cut-off parameter  $q_0 \in [0, 1]$ , and a random number  $r$  for each step,  $r \in [0, 1]$ :

- (a) exploiting: pick the  $j$  which maximizes  $p(j)$ , if  $r < q_0$ ;
- (b) exploring: pick the  $j$  distributed as  $p(j)$ , if  $r \geq q_0$ .

A typical value for  $q_0$  is  $q_0 = 0.9$ , which has been shown to give the best results for this algorithm.

When the next location  $j$  is chosen, the ant step there, the ant arrival time  $t_a$  and the new ant time  $t'_d$  (the new departing time) at  $j$  is updated:

$$t_a = t_d + d_{ij} \quad (3)$$

$$t'_d = t_a + wt_j + s_j \quad (4)$$

where  $t_d$  is the departing time from  $i$ ,  $wt_j$  the waiting time at  $j$  (if  $t_a > e_j$ ) and  $s_j$  is the service time at  $j$ . The whole process is repeated, until a depot is chosen for the next step or it is not possible to find a  $j$  satisfying the constraints. In this case the ant returns at the depot. If more customers need to be serviced and the number of tours does not exceed the maximum number of tours allowed (an argument that is passed to the algorithm), a new tour is initiated, otherwise the construction procedure is complete.

### 5.2. Pheromones update

Pheromones can be updated either locally or globally.

Local update is performed during the ant constructive procedure in the following way:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho \cdot \tau_0 \quad (5)$$

where  $i$  and  $j$  are the indexes of the traversed arc,  $\tau_0 = 1/(N \cdot J_{\psi^{NN}})$  is the initial value of the pheromones,  $N$  is the number of customers,  $J_{\psi^{NN}}$  is the total distance of the initial solution  $\psi^{NN}$  found with a nearest neighbor heuristics,  $\rho \in [0, 1]$  is the evaporation coefficient, usually set to  $\rho = 0.1$ . This update is equivalent to a decrement of the pheromone on the arc  $(i, j)$ , since the pheromones are initially set to  $\tau_0$ .

The global update is performed once the two colonies have finished their iterations, using the best solution found so far  $\psi^{gl}$ :

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij} + \rho / J_{\psi^{gl}}, \quad (i, j) \in \psi^{gl} \quad (6)$$

where  $J_{\psi^{gl}}$  is the length of  $\psi^{gl}$ .

### 5.3. ACS-TIME

The ACS-TIME colony has the objective of minimizing the total length of the solution. Using the ant constructive procedure, a number of  $k$  ants (usually  $k = 10$ ) search for an improved solution, with a maximum number of tours equal to  $nT_{best}$  (the number of tour of the best solution so far) and with the parameter  $IN_i = 0$  (in Eq. (2)) for all  $j$ . The algorithm's outline is shown in Fig. 2.

### 5.4. ACS-VEI

The ACS-VEI colony attempts to find a feasible solution with a lower number of tours than the best (and feasible)  $\psi^{gl}$  found so far. The best ACS-VEI solution found so far,  $\psi^{ACS-VEI}$ , is the unfeasible solution having the minimum number of undelivered customers. The algorithm also updates the term  $IN_i$  (in Eq. (2)), by incrementing it by one each time the customer is left out of a solution. The term is reset each time an improved  $\psi^{ACS-VEI}$  solution is

```

ACS-TIME(  $nT_{best}$  ) // run with a max number of tours equal to  $nT_{best}$  number of tours
                        // of the best solution so far.
while (keepLooping)

    for each ant k
        ant (  $nT_{best}$ ,  $IN=0$  ) // constructive procedure to find a solution  $\psi$  and the pheromone local update
        if (  $\psi$  is not feasible )
            try post-insertion // procedure to insert left customers in the tours of the unfeasible
                               // solution (see 5. Local search and other considerations).

        if (  $\psi$  is feasible )
            run local search on  $\psi$ 

        if (  $J_{\psi} < J_{\psi^{gl}}$  ) // better solution found
             $\psi^{gl} = \psi$ 
            keepLooping = false

    global pheromone update on  $\psi^{gl}$ 

```

Fig. 2. Outline of the ACS-TIME algorithm.

```

ACS-VEI (  $nT_{best} - 1$  ) // run with a max number of tours equal to  $nT_{best}$ , number of tours of the best solution so far.
// (  $nT_{best} - 1$  ) is because ACS-VEI tries to find a solution with a tour less.
while (keepLooping)
    for each ant k
        ant (  $nT_{best} - 1, IN$  ) // constructive procedure to find a solution  $\psi$  (includes the pheromones' local update)

        if (  $\psi$  is not feasible )
            try post-insertion // see section 5. on post-insertion description.
            run local search

        if (  $\psi$  is feasible ) // best solution found
             $\psi^{gl} = \psi$ 
            reset IN for all the customers
            reset all the  $\tau_{ij} = \tau_0$ 
            keepLooping = false

        if (  $nUndeliveredCustomers(\psi) < nUndeliveredCustomers(\psi^{ACS-VEI})$  )
             $\psi^{ACS-VEI} = \psi$  // better ACS-VEI solution found
            keepLooping = false

    global pheromone update with  $\psi^{gl}$ 
    global pheromone update with  $\psi^{ACS-VEI}$ 

```

Fig. 3. Outline of the ACS-VEI algorithm.

found. In the ACS-VEI algorithm, the  $\psi^{ACS-VEI}$  solution is also used to perform the global pheromone update. The outline of the algorithm is presented in Fig. 3.

### 5.5. MACS-VRPTW

The MACS-VRPTW algorithm initializes and coordinates the two colonies, by updating the best (feasible) solution,  $\psi^{gl}$ . When an ant finds a solution with a tour less, the colonies are stopped, and two new colonies are activated with the updated value of  $nT_{best}$ . The outline of the algorithm is presented in Fig. 4, where we denote by  $nT(\psi)$  the number of tours of the solution  $\psi$ .

### 5.6. Time dependent MACS-VRPTW

In the time dependent VRPTW, TDVRPTW, on each existing oriented arc  $a \in A$ , information about the travel time must be given to deduce the time necessary to traverse the arc when starting the trip at a time  $t$ . This information can be provided in two different ways: (1) a travel time distribution  $T_{ij}(t)$ ,

which is continuous in  $t$ , (2) a step-like speed distribution, from which a continuous travel time distribution can be obtained by integration. We use a speed distribution  $v_{ij}(t)$ , defined on the time interval  $[t_0, t_c]$ . A step-like speed distribution on each arc induces a partition of the time in periods of time  $S_k$  defined by intervals  $[t_0^s, \dots, t_k^s]$ . Within the intervals the speed is constant; this allows to formulate the algorithm in the time subspaces  $S_k$  as the classic VRP case. This is schematically represented in Fig. 5, where three subspaces are shown, and to simplify this representation, the partition of time is assumed to be the same for all the arcs considered.

The pheromones can then be represented in the following way: each oriented arc connecting the nodes  $(i, j)$  is associated with the time dependent distribution  $\tau_{ijk}$ , where the index  $k$  refers to the subspace  $S_k$ . In other words, the element  $\tau_{ijk}$  encodes the convenience of going from  $i$  to  $j$  when originating a trip from  $i$  at a time  $t$  in  $S_k$ .

The TDVRPTW feasible solution uses an update rule similar to one in Eq. (3) to update the tour length, but it is based on the travel time instead of the distance. In this formulation, the optimization



```

begin:
  find first feasible solution
  init best solution and optimization objectives:
     $\psi^{gl} = \psi^{NN}$ 
     $nT_{best} = nT(\psi^{NN})$  and  $J_{\psi^{gl}} = J_{\psi^{NN}}$ 
  initialize pheromones:
     $\tau_0 = 1/(N \cdot J_{\psi^{NN}})$  and  $\tau_{ij} = \tau_0$ , for all the existing arcs.
Iterate:
  while (optimization time is not expired)
    activate ACS-VEI( $nT_{best} - 1$ )

    activate ACS-TIME( $nT_{best}$ )
    while (ACS-VEI and ACS-TIME are active)

      wait for an improved solution,  $\psi$ 
       $\psi^{gl} = \psi$ 
      if ( $nT(\psi) < nT_{best}$ )
        stop the colonies
         $nT_{best} = nT(\psi)$ 

    check optimization time
  return  $\psi^{gl}$ 

```

Fig. 4. Outline of the MACS-VRPTW algorithm.

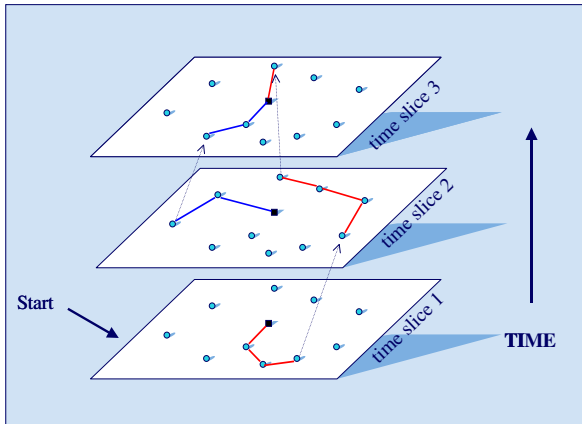


Fig. 5. The partition of the model time in subspaces, and process of construction of two tours through the time.

algorithm must find the solution that minimizes the number of tours and then the total traveling time. The probability distribution used by the ants during tour construction to select the next customer  $j$  of the feasible set  $J$ , corresponding to Eq. (1), depends now on the departing time from  $i$ . The rule is thus modified with:

$$p_j = \tau_{ijk} \cdot h_{ij}(t_d), \quad j \in J \quad (7)$$

where  $t_d \in T_k$  is the departing time from  $i$  (e.g. the time the work at  $i$  is completed) and  $k$  is then the corresponding time index, while:

$$h_{ij}(t_d) = \frac{1}{\max(1, (T_{ij}(t_d) + wt_j) \cdot (e_j - t_a) - IN_j)} \quad (8)$$

is the local heuristic function,  $T_{ij}(t_d)$  is the travel time when departing from  $i$  at the time  $t_d$ ,  $wt_j$  is waiting time at  $j$ , and  $e_j - t_a$  is the difference between the arrival time at  $j$  (that is  $t_a = t_d + T_{ij}(t_d)$ ) and the respective end of the time window.

The constructive procedure will proceed in the same way as before, but all occurrences of the concept of distance will be replaced by the concept of traveling time. In the same way, we will deal with total traveling time, instead of the total length of the solution.

We note that in the model presented here there is only one depot, while in the original implementation of the MACS-VRPTW the depot was replicated as many times as the current maximum number of

tours (the number of tours of the best solution, all depots having the same location). In the original algorithm, at each new tour a virtual depot is picked-up randomly, and then the first step is computed with Eq. (1), where each virtual depot has its own pheromone distribution. This mechanism is very effective to further diversify and improve the search, and it has been adapted in the Time Dependent version of VRPTW (TDVRPTW) where there is only one depot in the model formulation. Before starting the computation of a new tour, a set of  $n$  customers is created, where  $n$  is the number of tours left to complete (that is  $n = nT_{\max} - nT(\psi_{\text{current}})$ ), of customers not visited yet, that have the highest value of the probability as given by (1). Within this set then, the first customer to visit is picked randomly. In this way a mechanism similar to the replication of the depots of MACS-VRPTW is provided.

Note that each arc is an oriented arc, so in this model we consider  $a_{ij} \neq a_{ji}$ , and similarly for their travel times distributions. It is very common indeed the situation where speed sensibly differs according to the direction of travel; e.g. roads connecting city centers with residential areas are congested in the mornings in the direction of downtown, and vice-versa in the afternoon.

## 6. Local search and other considerations

Local search procedures have been proven to be very useful in improving the quality of the solution by evaluating if small modifications can return a better solution. The two basic operations we can perform in a local search procedure applied to the VRP are: (1) insertion of a new delivery in a tour, (2) removal of a delivery from a tour. In the case of the TDVRP, since both operations generate a time shift for all the customers following an insertion or a removal, the travel times from a customer to the next can change, and so delivering times. In particular also the removal of a customer can create a delay.

We present here a method based on the use of a *push* variable, as discussed by Kindervater and Savelsbergh in [17], adapted to the time dependent case. This variable, called the *slack time*, is stored for each delivery (and kept updated) and indicates how long the delivery can be delayed so that none of the time windows of the following customers (including the depot closing time) will be missed.

The slack times are calculated backwards, starting with the ending depot, once a tour is completed, by

$$s_i = \min(s_{i+1}, e_i - at_i) \quad (9)$$

where  $i$  is the customer's index in the tour, is the arrival time at  $i$ . The slack time is calculated starting with the last node (the depot):  $s_{N+1} = t_c - t_e$  the depot closing time  $t_c$  less the time when the tour ends  $t_e$ .

Because in the time dependent model any time shift produces a change in the travel time, the slack time of the next customer  $s_{i+1}$  needs to be appropriately adjusted. In other words, one needs to calculate the maximum delay before leaving  $i$ , keeping into account that within this delay the travel time might change.

This issue has been solved as follows. If  $g$  is the function representing the arrival time at the next location  $i + 1$ , the maximum delay  $\Delta_i$  on arrival time at  $i$ , must satisfy:

$$g(at_i + \Delta_i) - g(at_i) \leq s_{i+1} \quad (10)$$

The possibility of back-propagating the delay at  $i + 1$  is then guaranteed if we can univocally assign a value to  $\Delta_i$ , no matter what speed distribution is set on the arc and the starting time from  $i$ . From Eq. (10), provided  $g$  is invertible, we have:

$$\Delta_i = g^{-1}(s_{i+1} + g(at_i)) - at_i \quad (11)$$

To prove that function  $g$  is invertible, we need to prove that it is continuous and monotonic. The continuity is guaranteed by the fact that the arrival time is the sum of the departing time and the travel time. The monotonic behavior requires that for  $t' > t \Rightarrow g(t') > g(t)$ , which is guaranteed by the FIFO principle that makes the arrival times monotonically increasing with the departing times, as shown in Fig. 6.

The invertibility of  $g$  provides the possibility of back propagating the slack times, and then to test the feasibility of a move in constant time, by using an equation corresponding to Eq. (9) but with the back propagated slack time:

$$s_i = \min(\Delta_i, e_i - at_i) \quad (12)$$

where  $\Delta_i$  is given by Eq. (11).

Once we have proven that there is a unique value when back-propagating a delay, there are two ways to compute  $\Delta_i$ . One is to use an approximation of the function  $g^{-1}$ , the other one is an exact method,



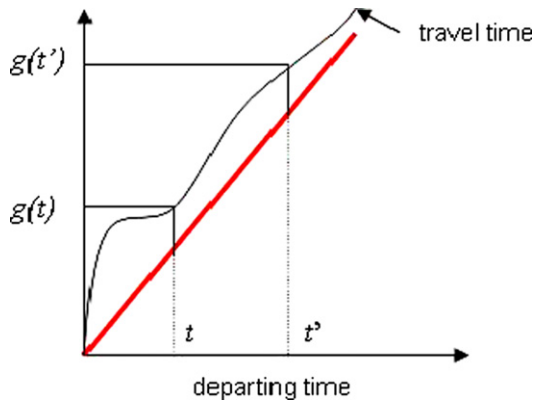


Fig. 6. The arrival time function, as a monotonic increasing function of the departing time.

which we have adopted in our model. It consists in finding the latest departing time  $ldt_i$  relative to the customer  $i$ , so that arrival time at  $i+1$  coincides with the latest arrival time,  $e_{i+1}$ . Once  $ldt_i$  is known, and the arrival time at  $i+1$ ,  $at_{i+1}$ , corresponds to the departing time  $dt_i$  (stored), the value of  $\Delta_i$  on  $i$  will be simply given by  $\Delta_i = ldt_i - dt_i$ .

The procedure to find the latest departing  $ldt_i$  is shown in Fig. 7. It computes the latest time at which the customer  $i$  must be left in order to arrive at the

next customer  $i+1$  no later than its upper time window. The procedure then takes as an argument the arrival time  $at_{i+1} = e_{i+1}$ , and back-propagates it to calculate  $ldt_i$ , where the index  $k$  is referring to the time subspaces  $S_k$  corresponding to  $at_{i+1}$ .

### 6.1. Neighbors' set

To maintain scalability on large instances, efficiency and improve the speed of the algorithm, it is very useful to introduce for each customer  $c_i$ , a set of neighbors. This is mainly motivated by the fact that in an optimized solution there will never be trips between distant locations, and this consideration sensibly speeds up the construction of the solution and local search procedures.

The set of neighbors is computed before starting the optimization, and it is composed of  $n$  closest customers  $c_j$ , in the sense of spatial-temporal closeness, that is distance and time windows overlapping. The move from node  $i$  to  $j$  is possible if, in the worst case, leaving at the latest time at  $e_i + s_i$  (end of the time window plus time when the work is complete), it is possible to reach the neighbor  $c_j$  at a time  $t < e_j$ . On the other hand, the earliest departing time from  $i$  is  $b_i + s_i$ ; thus, if wait time at  $c_j$  is too long, the cus-

```

calculateDepartingTime (ati+1) // returns the departing time at i

initialize:
  k = getTimeIndex(ati+1)
  dti = ati+1
  remainingDistance = di,i+1 // the distance to be covered between i and i+1
  distanceCoveredInTS = (ati+1 - tks) * speeds[k] // the distance traveled in the time subspace

  // this is the distance covered in the last time subspace Sk where tks is the time of the lower
  // limit of the interval defining the subspace when the speed changes

  // go backward over the time subspaces and calculate the departing time
  while ( remainingDistance > distanceCoveredInTS & k>0)
    k--
    remainingDistance -= distanceCoveredInTS
    dti = tks

    distanceCoveredInTS = (tks - tk-1s) * speeds[k]

  // final adjustment
  dti -= remainingDistance / speeds[k] // last time subspace
  return dti

```

Fig. 7. Procedure (defined within an arc object) to calculate the departing time from the starting node once the arrival time at the ending node is known.

customer is excluded by the neighbors set. The maximum wait is set to be a fraction (usually 1/4) of the time horizon.

Only arcs among neighbors are created for the problem, while each customer has a connecting arc to the depot, and the depot is connected to all the customers. Note that the neighbor's relationship is not symmetric. The numbers of neighbors is usually set to 30–50.

## 6.2. Description of the local search procedures

Once we have guaranteed the existence of the function  $g^{-1}$ , and therefore the validity of the procedure of Fig. 7, we present here the local search procedures we have used. Note that the analysis of the feasibility the first operation is always performed in order to verify the neighbor's relationships between the removed/inserted customers, and this can be done in  $O(1)$ , thanks to the use of appropriate sorted/indexed lists.

To evaluate the goodness of a move, in the time dependent context, where all traveling times on the following arcs can in principle be affected by the move, we proceed as follows. First, a local evaluation if any improvement is found for all the arcs affected by the change; second, the overall effect of the change is evaluated on the newly created tour(s) checking if the move has actually provided an improvement. A schematic outline is showed in Fig. 8.

For each operation, checks are done also on the available quantities and quantities delivered whenever performing exchanges of one or more customers between tours.

The local search procedures used here are the following.

1. *Customer relocation*: the procedure evaluates the advantage of moving the delivery to a customer in a different tour, or at a different moment/position in the same tour (in-tour relocation). This is done for all the tours and positions. The local evaluation consists in evaluating if the added arcs to the customer have a smaller travel time than the removed ones, and the insertion point that provides the maximum advantage is selected.
2. *Customer exchange*: from a tour to another, with one customer of the other tour. The best customer (and respective tour) is selected such to minimize the variation in total travel time of the newly obtained solution.
3. *In tour 2-k opt*: each customer  $c_i$  ( $i=0, \dots, d-2$ , where  $d$  is the number of deliveries in the tour) and all the following customers  $c_j$  (with  $j=i+1, \dots, d-1$ , where  $j=i$  would be equivalent to a customer in tour relocation) in the same tour are checked to see if the tour obtained by the inversion of the branch from  $i$  to  $j$ :

$$c_{i-1} \rightarrow c_j \rightarrow c_{j-1} \rightarrow \dots \rightarrow c_i \rightarrow c_{j+1}$$

provides an improved solution, where the nodes  $c_{i-1}$  and  $c_{j+1}$  are the depot if  $i=0$  or  $j=d-1$ . Note that on the customers of the branch  $c_j \rightarrow c_{j-1} \rightarrow \dots \rightarrow c_i$  no check on the slack time can be done, so only a time windows constraints check is done. On  $c_{j+1}$  the slack time can be checked, once the arrival time has been recalculated.

TD Local search:

```

check feasibility  $O(1)$ :
  neighborhoods relationship
  slack times
  other problem/time constraints (quantity available,
  arrival times if the slack time is not usable)

check local improvements ( $O(1)$ , for operations 1. and 2.
  below, involving single customers, cached partial
  travel time for 3. 4. 5.).

check global improvement: for the newly created
tour(s), all times following a change need to be
recalculated ( $O(n)$ ).
```

Fig. 8. Outline of the time dependent local search.

lated. This move allows eliminating crossings in the tours.

4. *Branch relocation*: consists in inserting a branch of a tour in another tour. For each tour  $t_1$ , tries are made exhaustively for all the customers  $i$  and  $j$  (with  $j = i + 1, \dots, d_1 - 1$ , where  $j = i$  would correspond to a customer relocation) trying all insertion points  $h$  (with  $h = 0, \dots, d_2 - 1$ ) of the second tour  $t_2$  (over all the other tours), to see whether:

new  $t_1 : c_{i-1} \rightarrow c_{j+1}$

new  $t_2 : c_{h-1} \rightarrow c_i \rightarrow \dots \rightarrow c_j \rightarrow c_h$

is feasible (including quantity checks) and provides an improved solution. If such an improved solution is found, the process continues with the next tour following  $t_1$  in the tour list. Note that again the slack time can only be checked on  $c_{j+1}$  and  $c_h$ , while for the customers  $c_i \rightarrow \dots \rightarrow c_j$  only a time windows constraint can be done. If such relocation is found and it is better, the relocation is performed, and tour after  $t_1$  (in the tours' list) is considered.

5. *Branch exchange*: consists in exchanging two branches of variable length among two tours. For all the customers pairs  $i$  and  $j$ , with  $j = i, \dots, d_1 - 1$  of the first tour  $t_1$ , and all the customers  $h$  and  $k$ , with  $k = h + 1, \dots, d_2 - 1$  (indeed for  $j = i$  and  $k = h$  it would be equivalent to a customers exchange) of the second tour  $t_2$ , are examined to see if the branch exchange:

new  $t_1 : c_{i-1} \rightarrow c_h \rightarrow \dots \rightarrow c_k \rightarrow c_{j+1}$

new  $t_2 : c_{h-1} \rightarrow c_i \rightarrow \dots \rightarrow c_j \rightarrow c_{k+1}$

is feasible (including quantities checks) and provides a shorter travel time for the 2 new tours. If such an exchange is found the next tour following  $t_1$  (in the tours' list) is considered. The second tour examined has an index greater than the first, given the symmetry in the operation of the exchange. Note that for operations 4 and 5 the local check cannot be done.

6. *Post insertion*: it is used when a solution found is not feasible, and a number of customers still need to be scheduled. In this case we try for each customer a post insertion, that is inserting the customers one by one in the tour that minimizes the increase of travel time, and that does not violate the constraints of the problem. The order in

which the undelivered customers are inserted is chosen probabilistically based on  $q_i$ , the quantity requested, so those with a higher demand will be more likely to be considered first.

7. *Shuffle the tours order*: this simple operation consists in randomly changing the order in which the tours are stored in the solution list, and it is done to ensure that any of the previous operation is not dependent on the order in which tours are stored.

A local search cycle consists in repeating steps 1–7 in order (with the exception of 6. that is performed on unfeasible solutions). This order is basically due to the increasing difficulty and complexity level of the operation, and consequently to perform a more complex operation assumes that the solution has been already optimized with respect to simpler procedures, e.g., it is advisable to check for tour  $2 - k$  opt (crossing) before checking for a branch exchange. The complete procedure is repeated a minimum of one time for a generic solution, and a minimum of 5 times for the best solution. Additional local search cycles are performed every time that any type of improvement has been found in the previous cycle, until no more improvements are found. Because of these repetitions, the effect of the order in which each operation is carried out, becomes negligible. The effects of local search procedures have been extensively discussed in [13], to show that the combination of an ACS and Local Search procedures gives better results with respect to other heuristics combined with the same procedures. The average improvement due to Local Search has been quantified in a range 10–20%. In particular, for the sequential ordering problem (SOP), that is an asymmetric TSP with precedence constraints, in [8] is reported that Ant Colony System (ACS) is better than genetic algorithm in combination with the same local search. This is due the fact that ACS produces starting solutions that are easily improved by the local search while starting solutions produced by the genetic algorithm quickly bring the local search to a local minimum.

## 7. Experimental results

Some experiments have been conducted to show some of the behaviors, issues and advantages of the use of this model.

### 7.1. Constant speed benchmark tests

The Solomon's problems are used to measure the performance of the MACS-TDVRPTW algorithm, when applied to the solution of the classic case. A constant speed distribution was used for all the arcs, with value set to 1, so that optimizing travel times is equivalent to optimize distances.

The Solomon's problems consist in 6 groups of problems, named R1, C1, RC1, R2, C2, and RC2. In each group there are 8–12 problems; in a group, customers have the same location and demand, but different time windows (usually broader from a problem to the following).

For each problem, the optimization is run three times (this is the usual number of runs in these benchmarks), and the average number of tours and distance is calculated. Once all the problems of the group are solved, the average number of tours and distance is calculated over the group at different times, up to 30 minutes (1800 s). These tests were run on a Pentium IV 2.66 GHz, and results are shown in Table 1.

A comparison with the analogous original results reported in [15] shows that at the end of the optimization (iteration at time 1800), the average deviations  $\delta$ , over all the families and the three runs, in tours and length of the MACS-TDVRPTW solutions from the original MACS-VRPTW solutions is  $\langle \delta nT \rangle = 0.08 \langle \delta J \rangle = -3.04$ . This means that even if the tours overall are shorter, the number of tours

is slightly higher, possibly due to experimental errors and that the new model is dealing with an higher dimensional space that slows down its computational speed.

The best solutions we have found are also compared with the absolute best solutions identified by various heuristic algorithms (as reported in [19]). These solutions have been obtained in an arbitrary computation time, and they are the absolute best known solutions found so far. This comparison shows that the results obtained by MACS-VRPTWTD are comparable to these solutions; moreover, the flexibility of our algorithm is shown by the consideration that it is compared against several different methods, and no single algorithm can find them all at once, even in an arbitrary time. As before, the average deviation  $\delta$  over all the families of the best solutions found by this model from the absolute best solutions is  $\langle \delta nT \rangle = 0.21 \langle \delta J \rangle = -5.81$ . The results are shown in Table 2.

Again the discrepancy in the number of tours, that is slightly higher in our case, can be also due to the more complex structure underlying MACS-VRPTWTD, which is dealing with an higher dimensional space that makes it run more slowly.

### 7.2. Classic solutions in a time dependent context

When solutions for the constant speed model are used in a time dependent context, their feasibility and optimality might considerably change, and this

Table 1  
Benchmark results in time on Solomon problems

Time (seconds)	R1		C1		RC1		R2		C2		RC2	
	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$
100	12.78	1216.38	10.00	830.48	12.63	1406.58	3.15	1002.79	3.00	596.19	3.63	1187.47
300	12.61	1209.65	10.00	828.82	12.29	1383.83	3.15	984.39	3.00	592.97	3.58	1168.63
600	12.61	1203.05	10.00	828.41	12.25	1374.49	3.12	977.15	3.00	597.06	3.54	1155.86
1200	12.61	1199.36	10.00	828.38	12.13	1373.78	3.09	972.31	3.00	590.49	3.46	1156.77
1800	12.61	1196.27	10.00	828.38	12.04	1372.71	3.09	966.95	3.00	590.49	3.38	1155.74

The average number of tours and distance were obtained at different times, to evaluate the speed and performance of the algorithm, and averaged over three different runs for each family of the problems (R, C, RC).

Table 2  
Comparison between best MACS-VRPTW solutions found in three runs of 1800 seconds each, and absolute best solution identified by any heuristics ever, as reported in [19]

	R1		C1		RC1		R2		C2		RC2	
	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$	$\langle nT \rangle$	$\langle \text{Dist} \rangle$
MACS-DTVRPTW best	12.33	1199.91	10.00	828.38	11.88	1359.84	3.09	946.21	3.00	589.50	3.38	1124.63
Absolute best	11.92	1209.89	10.00	828.38	11.50	1384.16	2.73	951.56	3.00	589.86	3.25	1119.36

change is proportional to the variability in speed distribution (that is, to traffic conditions).

In this simple experiment we used five different 4-valued speed distributions of the type  $[v_1, v_2, v_3, v_4]_h$  with  $h = 1, \dots, 5$ , and defined over four equal intervals of time dividing the depot time window, and such that the average speed is  $\langle v \rangle = 1.0$  for all  $h$ . The values used are defined in Table 3, where the type  $h = 1, \dots, 5$  is used to characterize the different types of roads.

In the tests, a further variation  $\gamma$  is introduced to progressively increase the variability in the speeds distributions to  $[v_1 - \gamma, v_2 + \gamma, v_3 - \gamma, v_4 + \gamma]_h$ , for  $h = 1, \dots, 5$ , while maintaining the average speed  $\langle v \rangle = 1.0$ , and a sensible comparison with the solutions of the Solomon problems is still possible. A total of six tests were conducted, each for the Solomon problems, R1, C1, RC1, R2, C2, and RC2. For each group, only the first problem is considered, being the one with the tighter time windows. For each problem, 10 different random assignments of the five speeds distributions on the arcs were done. For each of the assignment, the optimization is repeated 10 times for 30 s, and then the degree of unfeasibility (the percent of missed time windows) of the classic best solution known for the problem is calculated (mTW), and then the value of  $\gamma$  is increased. The average percent of missed time windows  $\langle \text{mTW} \rangle \%$  and its standard deviation has been calculated over the 10 different speed assignments,

Table 3  
Speed distribution used for the evaluation of the classic solutions in a time dependent context

	$v_1$	$v_2$	$v_3$	$v_4$
Type 1	0.90	1.10	0.80	1.20
Type 2	0.80	1.20	0.90	1.10
Type 3	0.70	1.30	0.50	1.50
Type 4	0.60	1.40	0.70	1.30
Type 5	0.50	1.50	0.60	1.40

Table 4  
Test of unfeasibility

$\gamma$	$\langle \text{mTW} \rangle \%$	$\langle \text{std} \rangle \text{mTW} \%$	$\langle \text{TT\_CT} \rangle$	$\langle \text{std} \rangle \text{TT\_CT}$	$\langle \text{TT\_TD} \rangle$	$\langle \text{std} \rangle \text{TT\_TD}$
0	9.37	2.85	1438.26	26.56	1587.02	68.97
0.1	15.80	3.71	1539.81	35.29	1778.93	94.95
0.2	27.90	4.76	1623.68	36.68	2010.29	96.20
0.3	44.77	5.42	1780.74	50.87	2415.42	107.29
0.4	59.10	5.55	2006.24	72.94	3006.92	171.26

Progressively increasing the degree of variability  $\gamma$ , in the variation of traffic conditions, the percent of missed time windows of the optimal solutions known to the six representative Solomon problems is calculated.

and then it has been averaged over the six groups. Similarly the total average travel time  $\langle \text{TT\_CT} \rangle$  for the constant time model, and the  $\langle \text{TT\_TD} \rangle$  for the time dependent model (with their standard deviations) has also been calculated.

The results are shown in Table 4.

The average percent of missed time windows is much higher in clustered problems (like C1 and C2, with values up to 95%) than for random problems (like in RC2 and R2, with a maximum around 40%). Also the gap in travel time increases between the classic solutions and the new solutions found, due also to the fact that the new solutions never miss any time window, while the classic solutions are increasingly unfeasible. We notice that in some cases the random assignments of the speeds result in a problem that is unsolvable also for the time dependent model. In this case we progressively anticipate the departing time from the depot (at earlier time than the opening time, up to 0.5 times of the depot opening window), till a feasible solution is found. In most cases it is possible to make the problem solvable.

The main conclusions of this analysis are then:

1. the degree of unfeasibility increases with the increase of the degree of time-dependency,
2. the classic solutions, even if they might seem to be better, are usually unfeasible,
3. if the classic solutions are feasible (large or no customers time windows), they are sub-optimal. This will be shown and discussed in Section 8.

### 7.3. Time dependent solutions

The aim of these experiments is to see if the optimization properly keeps into account the system variable traffic conditions.

First, we have created an *ad-hoc* problem, using the following settings. There are two time intervals

for the speeds distributions, and all the arcs have the same speed distribution (a two value speed distribution). The customers' locations are chosen so that there are two subsets of customers: those that are fairly close one to another one and those that are (in average) as double as distant. In this experiment we used three of such groups of customers. For this experiment the delivery time windows were removed, and service times properly chosen, since being hard constraints factors would have the effect to hide the presence of time dependency for the analysis we are interested in.

The experiment consists in using two distinct 2-valued speed distributions on all the arcs: one with a profile of type LOW  $\rightarrow$  HIGH (meaning the first period with a low value for the speed, the second with high value of the speed), and the other one of the type HIGH  $\rightarrow$  LOW.

Fig. 9 shows that three tours are formed but the orientation of the tours is inverted. This is evidently a consequence of the different speed distribution. In an optimized solution, the tours are formed in a way to use the high speed to complete the longer legs, and the low speed to complete shorter legs.

The same type of experiment has been repeated for a larger sample, for the all the Solomon problems, with no time windows to remove the effect of the constraints.

For this analysis, two main families of 4-valued speed distributions were used, the one in Table 3, and the other resembling a more realistic situation,

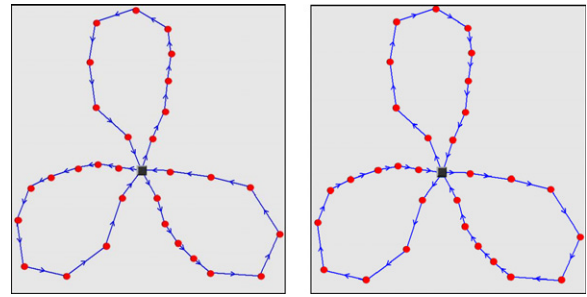


Fig. 9. Best solutions found, respectively relative to a LOW  $\rightarrow$  HIGH speed distribution (left), and relative to a HIGH  $\rightarrow$  LOW speed distribution (right). Note that the orientation of the tours is inverted.

Table 5

Another speed distribution used to calculate the distribution of the speed in relation to the arc length

	$v1$	$v2$	$v3$	$v4$
Type 1	1.00	1.00	1.00	1.00
Type 2	1.00	1.00	3.00	3.00
Type 3	3.00	1.00	1.00	3.00
Type 4	3.00	3.00	1.00	1.00
Type 5	3.00	3.00	3.00	3.00

presented in Table 5. This distribution contains arcs that are bottlenecks (type 1, always busy), the inflows (type 2, busy in the morning hours), those that are busy at mid day (type 3), those that are outflows (type 4, busy in the evening), and those that are seldom traveled (type 5).

bin	<speed>	times traveled	std speed
1	0.97	100.00	0.06
2	1.04	100.00	0.06
3	1.09	99.83	0.08
4	1.13	98.67	0.10
5	1.14	94.50	0.15
6	1.17	93.17	0.17
7	1.14	75.83	0.24
8	1.17	64.17	0.24
9	1.23	34.33	0.25
10	1.23	30.50	0.26
11	1.23	17.17	0.24
12	1.22	7.00	0.21
13	1.19	4.33	0.11
14	1.02	2.50	0.05
15	0.42	1.00	0.04
16	0.69	1.50	0.03
17	0.25	0.17	0.00
18	0.00	0.00	0.00
19	0.00	0.00	0.00
20	0.00	0.00	0.00

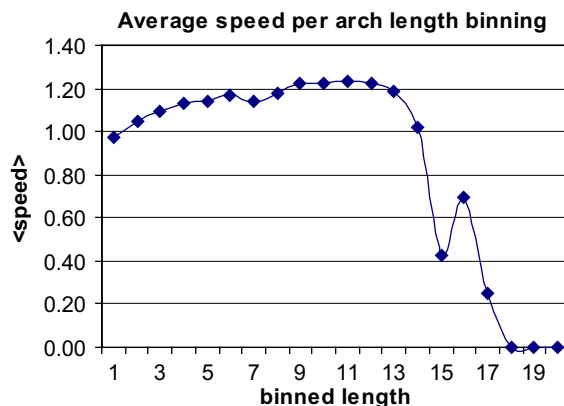


Fig. 10. Statistics on the average speed distribution in function of the binned arc length, for the speed distribution of Table 3.



bin	<speed>	times traveled	std speed
1	.62	100.00	0.19
2	.87	100.00	0.11
3	.96	100.00	0.07
4	.96	100.00	0.08
5	.98	96.00	0.05
6	.99	97.33	0.03
7	.99	92.17	0.04
8	.00	87.50	0.01
9	.99	76.17	0.03
10	3.00	62.17	0.04
11	3.00	46.67	0.04
12	3.00	29.67	0.01
13	2.99	25.33	0.05
14	3.00	13.33	0.03
15	2.99	7.00	0.05
16	3.00	4.83	0.03
17	3.00	2.67	0.02
18	0.00	0.00	0.00
19	0.00	0.00	0.00
20	0.00	0.00	0.00

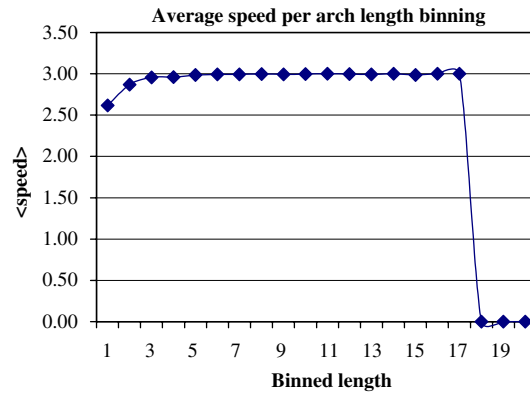


Fig. 11. Statistics on the average speed distribution in function of the binned arc length, for the speed distribution of Table 5.

For each of the two families of speed, we have considered the six Solomon problems without time windows. For each, we made 10 different random assignments of the speed distribution to the arcs, and for each, the 10 optimization runs were done for 5 minutes on a 2.66 GHz Pentium IV. At the end of each optimization, for the best solution found, the distribution of the binned arc length as a function of the number of times and the speed the arc was traveled was calculated. The results over the six Solomon problems for each of the two families of speeds are shown respectively in Figs. 10 and 11.

The two graphs show that the algorithm is favoring routes with longer legs in periods when speeds are higher, and shorter legs in periods when the speeds are lower. There is a tail effect in Fig. 10 due most likely to the fact that some routes have to be terminated due to the capacity constraint.

## 8. Application to a real road network

In this section we present the application of the MACS-TDVRPTW to a real road network. Real data obtained from the Padua logistic district, in the Veneto region of Italy, are used in this case study.

The customers are a set of nodes that is a subset of all the nodes of the graph representing the road network of Padua. Paths connecting each pair of customers need to be calculated. Since the time

dependent nature of this model, these paths are in principle also time dependent.

There are two alternatives: (1) calculate the shortest paths on the fly, that is, at departure time from a location to the next; (2) store one or a set of paths that represent a suitable approximation of the problem, so that the proper pre-calculated path from the list will be selected given the departing time.

The first option would imply the added computational effort to calculate at each location the paths and travel times for going to all the next possible remaining locations  $j$  when constructing the probability distribution of Eq. (6). For this reason, we have initially adopted the second method, pre-computing the shortest paths among all the customers' pairs with a robust shortest path algorithm [18]. A more accurate extension of this method involves a time dependent interval graph and a set of time intervals (computed for each arc) on which the path can be considered fixed. In this way, between each pairs of nodes there would be a set of paths, depending on the time of the day the trip is initiated. This extension has been implemented and applied to perform a scenario analysis, since it can deal with restricted access areas (like city centers) throughout the day.

The complete graph  $G = (V, A)$ , with all the nodes and arcs of the real road network, has been used for the computation, and a subset of  $V_c$  customers  $V_C \subset V$  has been placed on that graph.

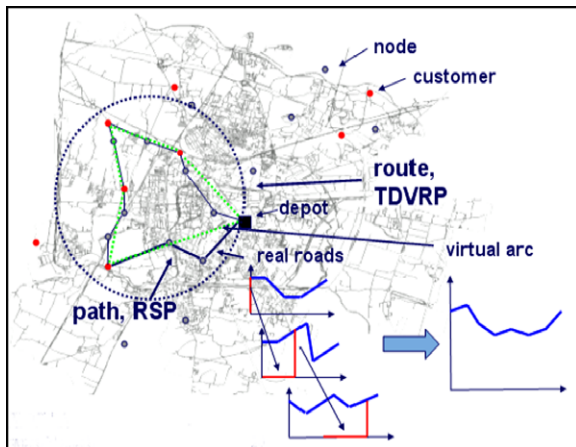


Fig. 12. The scheme of computation of traveling times on the virtual arcs once the robust shortest path (RSP) between the customer pair has been computed.

For each pair of customers, the robust shortest path is calculated. The optimization is then initialized by considering, among all nodes, only those relative to the depot and the customers, and by creating a set of oriented arcs  $A_c$  (with  $A_c \not\subset A$ ) among each customers' pair. Each arc  $a_c \in A_c$  is associated with a robust shortest path, and a time dependent travel time distribution is derived for each of such composite arcs, by considering the speed distribution on the arcs belonging to the path. These arcs are referred also as *virtual arcs*, in the sense that they encode a sequence of arcs  $a \in A$  with composite information about the traveling times. A visualization of the procedure to compute the new travel time distributions is shown in Fig. 12.

Note that when considering the time dependent paths case, this scheme is still valid with the same set  $A_c$ , except that each virtual arc will have a list of paths and a time partition, so that once the departing time is known, the proper path can be chosen and the proper travel time distribution calculated to reflect the path followed.

### 8.1. Computational results

A number of tests have been conducted using the data collected on the Padua road network by the automated traffic control system “Cartesio” [18]. The system records in real time the volumes of traffic and speeds of the vehicles, air and noise pollution data, and can provide traffic information through different channels (such as Variable Messaging System (VMS) panels, Radio Data System (RDS) mes-

sages or SMS messages). Logistic centers can access “Cartesio” on the Internet.

The data consists in a set of 1522 geo-referenced nodes and 2579 arcs. Road types (sections) and traffic data are measured for all arcs in rush hours (9.00 am), and for 50 of these, every hour. From these data the speed distribution of the 50 arcs can be deduced, and the speed distributions for all the other arcs are then deduced by using the average speed distribution over the 50 arcs adjusted (shifted by a constant) to match the value of the rush hour speed.

A set of a total of 60 customers and their demands is given, and for this application we considered the depot to be located at the Interporto Padova, with opening time [8.00, 18.00] and a fleet of 10 trucks. The model can deal with non-constant truck capacity, but no optimization is done in this sense at this point. No customers' time windows were specified for this set, while the service time was set to 1/2 hour for all the deliveries. We partitioned the depot opening time in four equal intervals.

Two types of experiment were run: the non-time dependent case using constant speeds (CS) obtained for each arc by averaging the traveling time distribution, and the time dependent case (TD).

We considered for each test a sample of 30 customers extracted randomly from the given sample of 60. For each test, the optimization was run five times for the CS model, and five times for the TD model, evaluating the best solution found for the CS model in the TD context. The average total travel time  $\langle T \rangle$  and standard deviation with respect to the runs, respectively for the CS solutions, the CS solutions in the TD context, and the TD solutions, are shown in Table 6 for each test. Times are expressed in hours, and fractions of hours. The number of tours is always equal to two in all the cases, so it has been omitted from Table 6. The computation time of each run was 5 minutes on a Pentium IV, 1.5 GHz machine.

The last column gives the difference between the  $\langle T \rangle$  of best TD-solutions with the  $\langle T \rangle$  of best CS-solutions for the data considered. In other words,  $\Delta$  evaluates the level of sub-optimality of a CS-solution in the time dependent context. The average of  $\Delta$ , over all the tests, is 7.58%, (for only 30 customers), with peaks to almost 12%.

In Fig. 13, an optimal solution for 30 customers is shown, using visualization software developed at IDSIA. Information about the solution (light/yel-

Table 6

Comparison of the best solutions obtained for the constant speeds model and the time dependent model

	Constant speeds		CS in TD		Time dependent		$\Delta$ (%)
	$\langle T \rangle$	$\sigma$	$\langle T \rangle$	$\sigma$	$\langle T \rangle$	$\sigma$	
Test 1	1.5186	0.0008	1.5255	0.0187	1.4396	0.0178	5.97
Test 2	1.5558	0.0002	1.6641	0.0630	1.5042	0.0426	10.63
Test 3	1.5974	0.0023	1.5784	0.0445	1.5000	0.0070	5.23
Test 4	1.6270	0.0005	1.6768	0.0131	1.4973	0.0063	11.98
Test 5	1.3472	0.0032	1.3750	0.0046	1.2892	0.0075	6.65
Test 6	1.54005	0.00319	1.5360	0.0442	1.4652	0.0284	4.83
Test 7	1.481206	0.00584	1.4061	0.0503	1.2728	0.0991	10.48
Test 8	1.16595	0.01211	1.1846	0.0311	1.1185	0.0106	5.91
Test 9	1.435494	0.00888	1.4305	0.0580	1.3427	0.0093	6.54

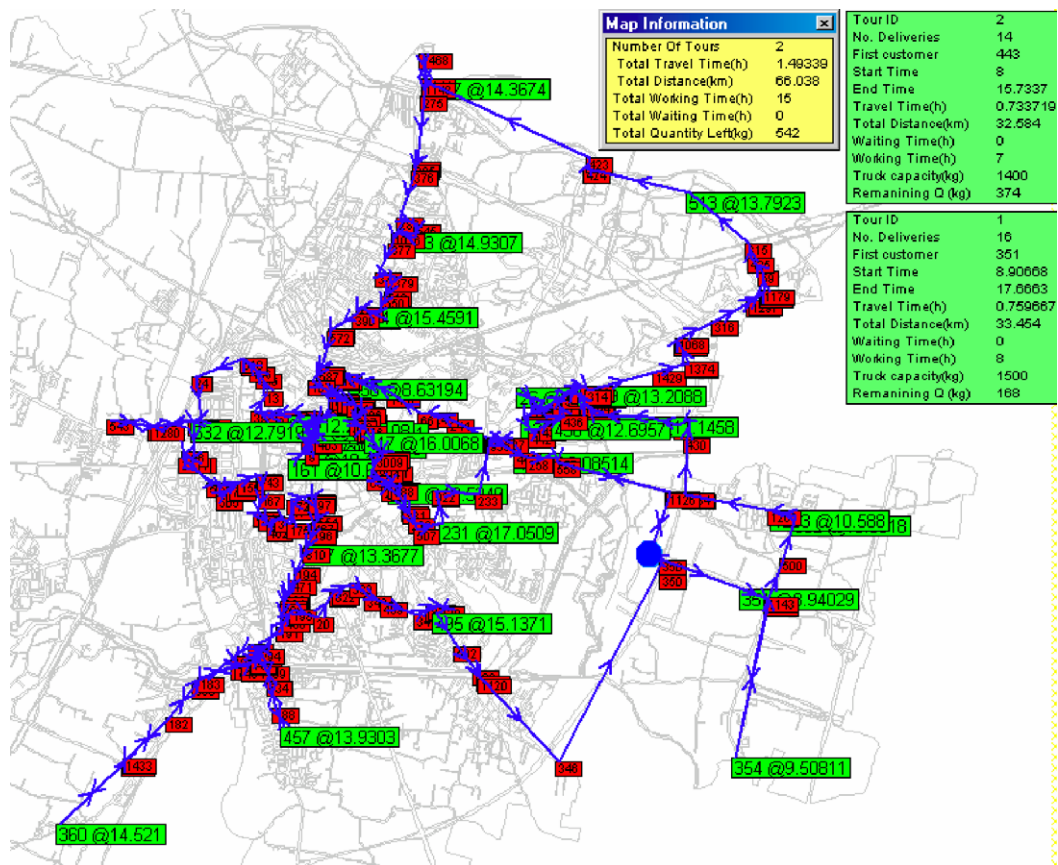


Fig. 13. Visualization of a TD solution. Red labels are the intermediate nodes' IDs, green are the customers' IDS with arrival times. The circle represents the depot.

low box)<sup>1</sup> and the tours (dark/green boxes) are also shown. Small/darker labels are the intermediate nodes' and their IDs, larger/lighter labels are the customers and their IDs, with arrival times, while the circle represents the depot.

<sup>1</sup> For a color version of this figure, the reader is referred to the web version of this article.

## 9. Conclusions

We have presented a time dependent model for the vehicle routing problem based on the MACS-VRPTW. The algorithms are supported by enhanced local search procedures, adapted to the time dependent case with a discretization model, to perform efficiently in terms of computation times

and quality of the solutions found. Advantages and issues of considering a time dependent model are discussed, as well as the quality and feasibility of the solutions in various cases. In conclusion, time dependent models can provide a better description in those cases when variable traffic conditions have a considerable influence.

## Acknowledgements

This work was co-funded by the European Commission IST project MOSCA: “Decision Support System For Integrated Door-To-Door Delivery: Planning and Control in Logistic Chain”, grant IST-2000-29557. The information provided is the sole responsibility of the authors and does not reflect the Community’s opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

## References

- [1] R. Montemanni, L.M. Gambardella, A.V. Donati, A branch and bound algorithm for the robust shortest path problem with interval data, *Operations Research Letters* 32 (3) (2004) 225–232.
- [2] C. Malandraki, M.S. Daskin, Time dependent vehicle routing problems: Formulations, properties and heuristic algorithms, *Transportation Science* 26 (1992) 185–200.
- [3] S. Ichoua, M. Gendreau, J.-Y. Potvin, Vehicle dispatching with time-dependent travel times, *European Journal of Operational Research* 144 (2) (2003) 379–396.
- [4] M. Dorigo, V. Maniezzo, A. Coloni, The ant system: Optimization by a colony of cooperating agent, *IEEE Transactions on Systems, Man and Cybernetics, Part B* 26 (1) (1996) 29–41.
- [5] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, *Artificial Life* 5 (2) (1999) 37–172.
- [6] T. Stutzle, M. Dorigo, ACO Algorithms for the Traveling Salesman Problem, in: P. Neittaanmaki, J. Periaux, K. Miettinen, M. Makela (Eds.), *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolutions Strategies, Evolutionary Programming, Genetic Programming, and Industrial Applications*, John Wiley & Sons, 1999.
- [7] T. Stutzle, H. Hoos, The MAX-MIN ant system and local search for the traveling salesman problem, in: T. Baeck, Z. Michalewicz, X. Yao, (Eds.), *IEEE International Conference on Evolutionary Computation and Evolutionary Programming Conference, Proceedings of IEEE-ICEC-EPS’97*, 1997, pp. 309–314.
- [8] B. Bullnheimer, R.F. Hartl, C. Strauss, Applying the ant system to the vehicle routing problem, in: I.H. Osman, S. Voss, S. Martello, C. Roucairol (Eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academics, 1998, pp. 109–120.
- [9] B. Bullnheimer, R.F. Hartl, C. Strauss, An improved ant system algorithm for the vehicle routing problem, presented at the Sixth Viennese workshop on Optimal Control, Dynamic Games, Nonlinear Dynamics and Adaptive Systems, Vienna (Austria), May, 1997, in: Dawid, Feichtinger, Hartl (Eds.), *Annals of Operations Research, Nonlinear Economic Dynamics and Control*, in press.
- [10] A. Coloni, M. Dorigo, V. Maniezzo, M. Trubian, Ant system for job-shop scheduling, *Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL)* 34 (1994) 39–53.
- [11] D. Costa, A. Hertz, Ants can colour graphs, *Journal of the Operational Research Society* 48 (1997) 295–305.
- [12] M. Dorigo, L.M. Gambardella, Ant colony system: A cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation* 1 (1997) 53–66.
- [13] L.M. Gambardella, M. Dorigo, HAS-SOP: An ant colony system hybridized with a new local search for the sequential ordering problem, *INFORMS, Journal on Computing* 12 (3) (2000).
- [14] L.M. Gambardella, E.D. Taillard, M. Dorigo, Ant colonies for the quadratic assignment problem, *Journal of the Operational Research Society* 50 (1999) 167–176.
- [15] L.M. Gambardella, E. Taillard, G. Agazzi, MACS-VRPTW: Vehicle Routing Problem with Time Windows, in: D. Corne, M. Dorigo, F. Glover (Eds.), *New Ideas in Optimization*, McGraw-Hill, London, 1999, pp. 63–76.
- [16] C. Blum, A. Roli, Metaheuristics in combinatorial optimization: Overview and conceptual comparison, *ACM Computing Surveys* 35 (3) (2003) 268–308.
- [17] G.A.P. Kindervater, M.W.P. Savelsbergh, Vehicle Routing: Handling Edge Exchanges, in: E.H.L. Aarts, J.K. Lenstra (Eds.), *Local Search in Combinatorial Optimization*, Wiley, Chichester, 1997, pp. 337–360.
- [18] Comune di Padova, IL SISTEMA CARTESIO ANALISI DEI FLUSSI DI TRAFFICO, DATI 1999–2000, Padua Municipality.
- [19] Best solutions to Solomon problems identified by heuristics (SINTEF VRP page): <http://www.sintef.no/static/am/opti/projects/top/vrp/bknown.html>. Last visited on 18.01.2006.