MULTIPLE ANT COLONY SYSTEM FOR A VRP WITH TIME WINDOWS AND SCHEDULED LOADING

MÚLTIPLES SISTEMAS DE COLONIAS DE HORMIGAS PARA UN VRP CON VENTANAS DE TIEMPO Y PROGRAMACIÓN DE LA CARGA

Pablo Ortega[1] **Cristian Oliva**[2] **Jacques Ferland**[3] **Manuel Cepeda**[2]

Recibido 28 de febrero de 2008, aceptado 10 de septiembre de 2009
Received: February 28, 2008 Accepted: September 10, 2009

RESUMEN

El problema de rutas para vehículos con ventanas de tiempo y programación de la carga no solo requiere el diseño de las rutas que satisfagan las restricciones temporales y de capacidad de los vehículos, sino que también la programación de las salidas de los vehículos desde un terminal dado un tiempo de carga debido a los recursos limitados disponibles para cargar las demandas de los clientes en los vehículos.
No solo se presenta una formulación del problema de diseño de rutas para vehículos con ventanas de tiempo y programación de la carga, sino que también se propone e implementa una metaheurística basada en múltiples sistemas de colonias de hormigas, cada una con una sola función objetivo, organizadas de manera jerárquica. Se incorpora una forma de actualización de tiempo dentro del procedimiento constructivo para actualizar y programar la salida de un vehículo desde el terminal cuando cada hormiga se mueve a un nuevo cliente. Se utiliza propagación de restricciones para determinar un movimiento factible a un nuevo cliente. Como el [VRPTWSL] incorpora la programación de la salida de los vehículos, el algoritmo presentado en este artículo tiene una directa aplicación a problemas reales, de esta manera, el [VRPTWSL] se puede tomar como un importante avance para problemas de diseño de rutas para vehículos.

Palabras clave: Problema de rutas para vehículos, sistema de colonias de hormigas, programación.

ABSTRACT

The vehicle routing problem with time windows and scheduled loading [VRPTWSL] requires not only the design of routes with time windows and capacity constraints, but also a schedule of the departures of vehicles from the depot given a load time due to the limited resources available to load the demand in the vehicles.
A mathematical formulation of the vehicle routing problem with time windows and scheduled loading is presented and a metaheuristics based on Multiple Ant Colony System is proposed and implemented where two ant colonies, each with a single objective function, are organized in a hierarchical way. A time update procedure is incorporated into the ant constructive procedure to update and schedule the departure of a vehicle from the depot when each ant moves to a new customer-node. Constraint programming is used to determine a feasible move to a new customer-node. As [VRPTWSL] incorporates the vehicle departure scheduling, the algorithm presented in this paper has a direct application to real problems, in this way [VRPTWSL] can be taken as an important advance for practical vehicle routing problems.

Keywords: Vehicle routing problem, ant colony system, scheduling.

INTRODUCTION

Throughout the last fifty years, the scientific community has been interested by several vehicle routing problems, mostly because of their inherent complexity. Furthermore, in almost every supply chain problem, some transportation of goods is required between the internal or external components.

The vehicle routing problem with time windows is specified in terms of minimizing the total time and cost for a fleet of vehicles to distribute goods from a depot to customers

[1] Departamento de Ingeniería Industrial. Facultad de Ingeniería. Universidad de Concepción. E-mail: paborteg@udec.cl
[2] Departamento de Ingeniería Industrial. Facultad de Ingeniería. Universidad Católica de la Santísima Concepción. Corresponding author. E-mail: coliva@ucsc.cl; mcepeda@ucsc.cl
[3] Département d'informatique et de recherche opérationnelle. Faculté des arts et des sciences. Université de Montreal. E-mail: ferland@iro.umontreal.ca

who need to be visited exactly once within their time windows (time intervals). All routes start and end at the same depot, and the total demand of all customers serviced by a vehicle along a particular route must not exceed its total capacity. Typical objective functions are to minimize the number of vehicles and the total travel time.

In this paper we propose a variant of the vehicle routing problem with time windows referred to as the Vehicle Routing Problem with Time Windows and Scheduled Loading [VRPTWSL]. In this problem, we have to account for the additional feature related to loading on each vehicle at the depot, the goods delivered to the customers on the associated route. Since the number of loading equipment (loaders, loading platforms, etc.) is limited, this induces a scheduling problem of the vehicles at the loading equipments. Consequently, the loading time and the waiting time before loading have an impact on the vehicle departure times from the depot. Such a problem occurs in the forestry industry where pine plants and eucalyptus trees packed in boxes have to be transported from a plant nursery where the number of loading equipments is limited, to different timber sites.

The [VRPTW] has been widely studied in literature. The first state of art reviews can be found in [9], [17], [18] and [24]. In [5] and [10] the authors mostly focus on exact approaches. Two of the best metaheuristics approaches for [VRPTW] are given in [16] and [3].

In this paper we formulate the [VRPTWSL]. The solution procedure implemented considers the special case where only one loading equipment is available (inducing a vehicle scheduling problem at the depot).This procedure is a hybrid of the Ant Colony System approach with the constraint programming approach to deal with the [VRPTWSL] constraints. We present some numerical results for problems randomly generated according to Solomon [23] process modified to account for the loading constraints of the [VRPTWSL].

The rest of the paper is organized as follows. First we include a formal description of the [VRPTWSL], and then, we introduce a mathematical model. After that, we present the hybrid method of the Ant Colony System and Constraint Programming approaches to solve the [VRPTWSL]. Later, we include the numerical results. Finally, we summarize the main conclusions and remarks.

VEHICLE ROUTING PROBLEM WITH TIME WINDOWS AND SCHEDULED LOADING

Consider the [VRPTW] specified as follows:

- n customers must be delivered from a unique depot;
- $[a_i, b_i]$ denotes the time window to service customer i;
- q_i denotes the demand of goods of customer i;
- p_i and h_i denotes the loading time of customer i goods at the depot and the unloading time at the destination, respectively;
- a set V of vehicles having homogeneous capacity Q is available;
- each route completed by a vehicle $v \in V$ starts and ends at the depot;
- a set G of homogeneous loading equipment is available at the depot ;
- each vehicle $v \in V$ must be loaded by a unique equipment $g \in G$ at the depot.

Underlying any [VRPTW], there is a complete graph G = (N, A) where N is the set of nodes and A is the set of edges. More specifically N = {0, n+1} U N', where 0 and n+1 denotes the depot (all routes start at 0 and end at n+1) and N' = {1,..., n} is the set of customers. Also $A = \left\{(i,j) : (i,j) \in \left(\left\{0, n+1\right\} \times N'\right) \bigcup N' \times N'\right\}$, where $N' \times N'$ denotes the set of edges connecting the customers and $\{0, n+1\} \times N'$, the set of edges connecting the customers and the depot.

The non negative value t_{ij} associated with each arc (i, j) denotes the travel time between i and j. Note that the delivery of goods for each customer i must start during its time window $[a_i, b_i]$. Hence the vehicle v delivering the goods to customer i can arrive before time a_i, but then it has to wait until this time to make the delivery.

MATHEMATICAL FORMULATION

Next we specify a mathematical model formulation for the [VRPTWSL] based on the model for the [VRPTW] presented by Toth and Vigo [25] where we introduce additional constraints related to the loading feature. The following variables are used in the mathematical model:

- Flow variables x_{ijv}, $(i, j) \in A$, $v \in V$,

$$x_{ijv} = \begin{cases} 1 & \text{if vehicle } v \text{ visits node } j \text{ after node } i \\ 0 & \text{otherwise.} \end{cases}$$

- w_{iv} = the starting time for the delivery of customer i by vehicle v, i (N, v (V. If vehicle v does not deliver goods to customer i, then wiv = 0.
- w_{0v} = the time when the loading of vehicle v by some equipment at the depot is completed, $v \in V \cup \{v_0\}$. Note that v_0 and $\bar{v}_0$ denote the first and the last dummy vehicles loaded by every loading equipment g, respectively. For this reason, we fix $w_{0v_0} = 0$
- Scheduling variables y_{lv}^g $l \in V \cup \{v_0\}$, $v \in V \cup \{\bar{v}_0\}$, $g \in G$,

$$y_{lv}^g = \begin{cases} 1 & \text{if vehicle } v \text{ is loaded by loading} \\ & \text{equipment } g \text{ after vehicle } l \\ 0 & \text{otherwise.} \end{cases}$$

From the definition of v_0 and $\bar{v}_0$, it follows that if $y_{v_0 v}^g = 1$ or $y_{v\bar{v}_0}^g = 1$, then v is in fact the first or the last vehicle loaded by equipment g, respectively.

The [VRPTWSL] model is summarized as follows:

In the mathematical model, the constraints (2) to (7) are those used by Toth and Vigo [25] to formulate the [VRPTW], and the constraints (8) to (11) are the additional constraints required for the loading feature of the problem.

The objective function (1) is a linear function formulated as the sum of the total travel times for the selected routes and a total fixed cost increasing with the number of vehicles used. The value of F is a parameter to be adjusted according to priority of the administrator to reduce the number vehicles used with respect to the total travel time.

Constraints (2) impose that each customer has his goods delivered by exactly one vehicle. Constraints (3) require that whenever a vehicle v leaves the depot (i.e., vehicle v is used), it must return to the depot. The flow conservation constraints are formulated in (4). The constraints (5) and (6) ensure that the time window constraints are satisfied. It is worth noticing that in constraints (5) h_0, the unloading time at the depot is equal to 0 (i.e., $h_0 = 0$). The constraints (7) are the vehicle capacity constraints.

The other constraints related to the loading feature of the problem can be seen as those of a (VRP) where the loading equipments correspond to the vehicles and the vehicles to the customers. Constraints (8) require that each vehicle is loaded by unique equipment. Constraints (9) indicate that the first and the last vehicle loaded by any equipment are the corresponding dummy vehicles.

The flow conservation constraints (10) guarantee that a unique vehicle l can be the next vehicle to be loaded by any equipment after loading vehicle v. Finally, we use the constraints (11) to determine the time when the loading of each vehicle is completed by some equipment.

$$\text{Min} \quad \sum_{v \in V} \sum_{(i,j) \in A} t_{ij} \cdot x_{ijv} + F \cdot \sum_{v \in V} \sum_{j \in N'} x_{0jv} \qquad (1)$$

Subjet to

$$\sum_{v \in V} \sum_{j \in \Delta^+(i)} x_{ijv} = 1 \qquad \forall i \in N' \qquad (2)$$

$$\sum_{j \in \Delta^+(0)} x_{0jv} = \sum_{i \in \Delta^-(n+1)} x_{i,n+1,v} \leq 1 \qquad \forall v \in V \qquad (3)$$

$$\sum_{i \in \Delta^-(j)} x_{ijv} - \sum_{i \in \Delta^+(j)} x_{jiv} = 0 \quad \forall v \in V, j \in N' \qquad (4)$$

$$x_{ijv} \cdot \left(w_{iv} + h_i + t_{ij} - w_{jv} \right) \leq 0 \quad \forall v \in V, (i,j) \in A \qquad (5)$$

$$a_i \cdot \sum_{j \in \Delta^+(i)} x_{ijv} \leq w_{iv} \leq b_i \cdot \sum_{j \in \Delta^+(i)} x_{ijv} \quad \forall v \in V, i \in N' \qquad (6)$$

$$\sum_{i \in N'} q_i \sum_{j \in \Delta^+(i)} x_{ijv} \leq Q \qquad \forall v \in V \qquad (7)$$

$$\sum_{g \in G} \sum_{l \in (V-v) \cup \{v_0\}} y_{lv}^g = 1 \qquad \forall v \in V \qquad (8)$$

$$\sum_{v \in V} y_{v_o v}^g = \sum_{v \in V} y_{v\bar{v}_0}^g \leq 1 \qquad \forall g \in G \qquad (9)$$

$$\sum_{l \in (V-v) \cup v_0} y_{lv}^g - \sum_{l \in (V-v) \cup \bar{v}_0} y_{vl}^g = 0 \quad \forall g \in G, v \in V \qquad (10)$$

$$y_{lv}^g \cdot \left(w_{0l} + \sum_{i \in N'} p_i \sum_{j \in \Delta^+(i)} x_{ijv} - w_{0v} \right) \leq 0$$

$$\forall l \in V \cup \{v_0\}, v \in V, \forall g \in G \qquad (11)$$

$$x_{ijv} \in \{0,1\} \qquad \forall v \in V, (i,j) \in A \qquad (12)$$

$$y_{lv}^g \in \{0,1\} \qquad \forall v, l \in V \cup \{v_0, \bar{v}_0\}, g \in G \qquad (13)$$

The binary conditions (12) and (13) on the flow and scheduling variables allow linearizing the constraints (5) and (11) as follows:

$$w_{iv} + h_i + t_{ij} - w_{jv} \leq (1 - x_{ijv}) \cdot M_{ij} \quad \forall v \in V, (i,j) \in A \qquad (14)$$

$$w_{0l} + \sum_{i \in N'} p_i \sum_{j \in \Delta^+(i)} x_{ijv} - w_{0v} \leq (1 - y_{lv}^g) \cdot R \quad \forall l \in V \cup v_0, \forall v \in V, g \in G \qquad (15)$$

where the constants M_{ij} and R take large values such that

$$M_{ij} \geq [b_i + h_i + t_{ij} - a_j], \forall (i,j) \in A, \; R \geq \sum_{j \in N'} p_j. \qquad (16)$$

MULTI ANT COLONY SYSTEM FOR VRPTWSL

Ant Colony System is a metaheuristics from the family of [ACO] algorithms, where a set of artificial Ants share information to build solutions. [ACS] was first proposed by [11] for the Traveling Salesman Problem and in recent years applications for the [VRPTW] have been proposed by [3] and [16] showing some of the best performances for this kind of problem; in particular [16] proposes a multi ant colony system for the [VRPTW] where the problem is transformed into a [TSP] considering a number of depots equal to the number of vehicles which must be used to serve the customers and in which two colonies in parallel minimize an objective function, sharing the global best solution. We propose an algorithm for the [VRPTWSL] based in a modified version of [MACS] presented by [23], where the two colonies run sequentially and we incorporate a *time update procedure* for the vehicle scheduling at the depot and a constraint programming based procedure to manage the constraints of a [VRPTW].

The aim of this paper is to propose an algorithm for the [VRPTWSL] based in [MACS] when only one load resource *g* is available at the depot; in that case vehicles scheduling is needed.

Following the previous we have named this procedure as [MACS-VRPTWSL] (Figure 1), where [MACS] is for multiple ant colony system, the first colony *ACS-VEI* minimizes the number of vehicles used to fulfill the total demand and has priority over the second colony *ACS-TIME*, which minimizes the total amount of time for the minimum number of vehicles found by *ACS-VEI*, both *colonies* use independent *pheromone trails* but share the variable ψ^{gb} which save the global best so far solution.

Each *colony* works in a very similar way, where every *ant* uses a constraint programming based procedure to determine a feasible domain, with probability q_0 applying the *Pseudo Random Proportional Rule* exploration or exploitation to choose the next node to be visit and call a *Time Update procedure* to update the departure and visits time in the previous nodes visited given the new node incorporated to the sub-route.

A feasible solution is a list of nodes starting from the depot and alternating customer nodes with depots. The last element from the list is the last customer node visited by the vehicle before it goes back to the depot. In this way the number of depots is equal to the number of vehicles. On the other hand the objective function incorporates the distance between the last customer node and the depot.

The first step of the algorithm is to determine an initial solution not necessarily feasible with the nearest neighbor heuristic incorporating the *Time Update Procedure* after every node selection.

```
/*MACS-VRPTWSL Procedure*/
Procedure MACS-VRPTWSL()
1. Initialization
/* ψ^gb: is the best feasible solution: lowest number of
vehicles and shortest travel time.
N: number of nodes in the graph.
v: current number of vehicles.
L_NN: total cost of the tour obtained by the Nearest
Neighbor heuristic.
L_ψ^gb :total cost of the best solution found*/

ψ^gb: initial solution found by Nearest Neighbor heuristic
not necessarily feasible
v: number of vehicles found by Nearest Neighbor
heuristic
Initialize variables using v /*Create a number of v
depots*/

2. /*Main loop*/
Do
    v <- v – 1
    ACS-VEI(v)
While ACS-VEI is feasible
v <- v + 1
ACS-TIME(v)
```

Figure 1. MACS-VRPTWSL procedure.

Follow the Nearest Neighbor [NN] heuristic solution with a number of |*V*| vehicles, *ACS-VEI* is call and tries to find a feasible solution with |*V*-1| vehicles used in ψ^{gb}. In this way, every time *ACS-VEI* finds a feasible solution, the process is restarted with one less vehicle until no feasible solution is found. The algorithm determines the minimum number of vehicles to be used and *ACS-TIME* is activated to improve the total time with the number of vehicles found by *ACS-VEI*. In *ACS-TIME* one *ant* can find a feasible solution with fewer vehicles found by *ACS-VEI*. In that case a depot node is dropped and the *ants* will work with the new graph. In this situation *ACS-TIME* will work with a number of vehicles less than the one found by *ACS-VEI*.

Time Updating

In this particular vehicle routing problem we have dropped the assumption that every vehicle starts its service from the depot at time zero because of the limited number of load

resources at the depot. Given this we must considerer the loading time of the demand at the depot in the vehicle every time a vehicle adds a new customer node to the sub-route. Thus a time update procedure must be defined in which dynamically updated total load time at the depot given the load of the new customer node incorporated to the sub-route and the *delay* obtained when the vehicle leaves the depot given this load. This procedure is named *Time Updating* and it is incorporated to the *ant's* construction procedure. To understand the *Time Updating procedure,* three concepts must be defined:

– Delay (*Delay*): The delay is the time in which the departure of the vehicle from the depot is delayed due to the incorporation of a new customer to the sub-route. This time delay can change the arrival time of the vehicle in the following customer nodes previously incorporated in the sub-route. The delay at the depots departure is equal to the loading time.

– Delta Time (*Delta_Time*): every time a vehicle arrives to a node before its bottom time window (a_i) it produces a *Delta Time*. The *Delta Time* from every node previously incorporated in the sub-route is equal to the bottom time window less the arrival time of the vehicle to the respective node (a_i – $Arrival\_Time_i$); when a vehicle arrives after the bottom time windows *Delta Time* is zero for the respective node.

– Accumulated Time (*Accumulated_Time*): *Accumulated Time* is the sum of every *Delta Times* present before a particular node in the sub-route. When accumulated time is bigger than zero for a particular node, *Accumulated Time* has the ability to diminish part of the *Delay* in the following nodes; this is because if the *accumulated time* for a node is big enough then the arrival time to this node can be delayed enough without affecting the arrival time to the following nodes.

The *Time Updating Procedure* (Figure 2) includes three stages: Update the *accumulated times* for every node previously incorporated in the sub-route given the last time update; update the arrival times of the vehicle for every node in the sub-route given the *delay* produced because of the new node to be visited and the *delays* that are diminished because the *accumulated time* of a particular node; finally, update the departure time of the vehicle from every node in the sub-route because the arrival time update to the node.

The *Update Time* procedure is as follows:

1. Locate the vehicle at the depot with departure time equal to the last vehicle departure; if no previous vehicle or sub-route is present the departure time is zero.
2. Choose a new customer node to be visited and update arrival time to the customer.
3. Calculate *Delays* at the depot and propagate to the nodes previously incorporated to the sub-route, and update arrival times for each node given the respective *Accumulated Times*.
4. Update *Accumulated Times*.
5. Go to stage 2 until there are feasible nodes to be incorporated in the sub-route.

```
/*Time Updating Procedure*/
Procedure Time_Updating(ψ^k, Time,
Accumulated_Time)
Arrival_Time_j <- time + t_ij
Delay <- p_j

If j is not a depot Then
For each i ∈ ψ^k from the current subtour_k
/*Update arrival time using accumulated time from
the previous time updating call*/

        If Accumulated_Time_i -1 = 0 Then
            Arrival_Time_i <- Arrival_Time_i + Delay
        Else
            If Accumulated_Time_i-1 ≥ Delay Then
                Delay <- 0
            Else
                Arrival_Time_i <- Arrival_Time_i +
                        Delay – Accumulated_Time_i-1
                Delay <- Delay – Accumulated_Time_i-1
            End If
        End If

/*Update departure time, delta time and accumulated
time from the respective node*/

        Calculate Departure_Time_i = max(a_i,
                        Arrival_Time_i + t_(i-1)i) + h_i
        Calculate Delta_Time_i = max(0, a_i –
                                Arrival_Time_i)
        Accumulated_Time_i = Delta_Time_i +
                            Accumulated_Time_i-1
End For each

/*Assign current time*/

Time <- Departure_Time_j
Else
        Accumulated_Time_j = 0
        Arrival_Time_j = last depot Arrival Time
        Departure_Time_j = Arrival_Time_j
End If
```

Figure 2. Time Update Procedure.

Set of Feasible Nodes

Let N_i^k set of feasible nodes *i* to be visited by *Ant k*. To determine N_i^k a constraint model is formulated to exploit Constraint Programming propagation properties. If *i* is a depot, N_i^k will be the set of nodes not yet visited; if *i* is a customer N_i^k is determined according to the constraint model. A constrained variable s_i is defined, where:

s_i: successor of node *i* in the sub-route. Where $s_i \in$ {*feasible nodes to visit*}

At the beginning, the ant is located in a depot. For this reason the domain of possible values for s_i is the set of all nodes. This domain must be updated as the ant constructs the sub-route. This domain must satisfy the following conditions:

4 No visited nodes by a vehicle in the route must be included.
5 When incorporating s_i Time windows constraint for the actual sub-route is still feasible after the incorporation, which means *Delay* is smaller than the difference between time windows b_i for every node in the sub-route and the sum between arrival time and accumulated time for every node in the sub-route.
6 The capacity constraint is satisfied.
7 Time windows constraints are satisfied.

If the set of feasible nodes is empty or at least 25% of capacity is used, N_i^k contains also all non-visited depots.
The constrained model is:
Let:
$Tour_k$: the tour follows by *ant k* before the incorporation of a new node.
u: the last node incorporated in $Tour_{k.}$
s_u: successor of node u.
$Subtour_k$: set of nodes visited by the current vehicle incorporated by *ant k*.
v: current vehicle.
Time: current time.
$Depature\_Time_i$: Departure time from node *i*.
Contraint 1: $s_u \neq tour_k[i] \quad \forall i \in tour_k$
Contraint 2:

$$b_i - Departure\_Time_i + Accumulated\_Time_i \geq Delay_{s_u} \quad \forall i \in subtour_k$$

Contraint 3: $q_{s_u} + \sum_{i=1}^{subtour_k} q_i \leq Q_v \quad \forall i \in subtour_k$

Contraint 4: $b_{s_u} - t_{0s_u} - h_{s_u} \leq b_0$

Constraint 5: Three cases:
Case 1: if accumulated time in u is equal to zero:

$$Time + t_{us_u} + h_{s_u} + Delay_{s_u} \leq b_{s_u}$$

Case 2: if accumulated time in u is greater than Delay.

$$time + t_{us_u} + h_{s_u} \leq b_{s_u}$$

Case 3: if accumulated time in u is smaller than Delay but greater than zero.

$$time + t_{us_u} + h_{s_u} + Accumulated\_Time_u - Delay_{s_u} \leq b_{s_u}$$

The constraint programming model was programmed in ILOG/solver 6.0.

ACS-VEI Colony

ACS-VEI searches for a feasible solution by maximizing the number of visited customers, in other words minimize the number of vehicle used.

```
/*ACS-VEI Procedure*/
Procedure ACS-VEI(v)
/*#Visited_Customer: Computes the number of
customers visited by ψ^k */
1.- /*Initialization*/
Initialize pheromone levels and heuristic
information using s
τ_ij ← 1/(|N| · L_NN), ∀(i,j) ∈ ψ^gb
2.- /*Cycle*/
Do
    For each ant k
        /*Build a solution ψ^k */
        Tour_Construction(k, IN)
        ∀cliente_j ∈ ψ^k : IN_j ← IN_j + 1
        If #Visited_Customers(ψ^k) = |N| Then
                ψ^gb <- ψ^k
                Pherecord_ij <- τ_ij
                Exit Procedure
        Else
                If #Visited_Customers(ψ^k) >
                #Visited_Customers(ψ^ACS-VEI) Then
                ψ^ACS-VEI <- ψ^k
                End If
    End If
End For each
/*Global Pheromone update using ψ^ACS-VEI and ψ^gb */
τ_ij = (1 − ρ) · τ_ij + ρ/L_ψ^ACS−VEI, ∀(i,j) ∈ ψ^ACS−VEI
τ_ij = (1 − ρ) · τ_ij + ρ/L_ψ^gb, ∀(i,j) ∈ ψ^gb
While a stopping criterion is met
```

Figure 3. ACS-VEI procedure.

ACS-VEI (Figure 3) starts its search with one less vehicle than the previous feasible solutions found, that is, a solution with |*V*| -1 vehicles. During this search *ACS-VEI* builds unfeasible solutions in which some customers have been not yet visited. The solution with the highest number of customers visited for |*V*|-1 is stored in a variable $\psi^{ACS\text{-}VEI}$, a solution is better than $\psi^{ACS\text{-}VEI}$ only when the number of visited customers is increased.

In *ACS-VEI* to maximize the number of visited customer, manages a vector of integers *IN*. The entry IN_j stores the number of times a customer *j* has been not incorporated in the solution. *IN* is used in the *constructive procedure* by the *Ants* for favoring less visited customer.

Finally at the end of each cycle, *ACS-VEI* increases the *pheromone trails* τ_{ij} in the path stored by the unfeasible solution $\psi^{ACS\text{-}VEI}$ with the highest number of visited customers so far and the feasible solution ψ^{gb} with the lowest number of vehicles computed and the lowest route time calculated so far.

The global pheromone levels are increased as follows:

Global updating rule for $\psi^{ACS\text{-}VEI}$:

$$\tau_{ij} \leftarrow (1-\rho)\times\tau_{ij} + \frac{\rho}{L_{\psi^{ACS-VEI}}}, \forall (i,j) \in \psi^{ACS-VEI}$$

Global updating rule for ψ^{gb}:

$$\tau_{ij} \leftarrow (1-\rho)\times\tau_{ij} + \frac{\rho}{L_{\psi^{gb}}}, \forall (i,j) \in \psi^{gb}$$

Where ρ ($0 \leq \rho \leq 1$) is a parameter that representing the pheromone level to evaporate, and $L_{\psi}{}^{gb}$ the objective value of the global best so far solution ψ^{gb}.

Each time *ACS-VEI* is activated it initializes the levels of pheromone to $\tau_0 = \frac{1}{|N| \times L_{NN}}, \forall (i,j) \in \psi^{gb}$ where L_{NN} is equal to the solution calculated by the [NN] heuristic and *N* the number of nodes. Each time *ACS-VEI* finds a feasible solution, it stores its *pheromone trails* in a matrix call *pherecord*. This matrix will be used by *ACS-TIME* to initialize its *pheromone trails*, and start their search with the *pheromone trails* equal to the last feasible solution found by *ACS-VEI*. According to [14] the time to find a better solution by *ACS-TIME* is diminished. *ACS-VEI* activates *m* ants to search for solutions.

ACS-TIME Colony

ACS-TIME (Figure 4) works very similarly to the traditional [ACS] based *colony* whose goal is to compute a tour as short as possible. *ACS-TIME* is activated only once, when *ACS-VEI* has found the lowest number of vehicles, and its *pheromone trails* are initialized with the *pherecord* matrix. In *ACS-TIME m* ants are activated to construct solutions.

If a feasible solution constructed by an *ant* is better than the one stored in ψ^{gb}, then ψ^{gb} is replaced with the new solution.

One *ant* could find a new solution with a number of vehicles less than the one found by *ACS-TIME*. In this case the procedure continues with a number of vehicles equal to the number of vehicles found by *ACS-TIME*.

The next step is to update the pheromone level as follows:

Global updating rule for ψ^{gb}:

$$\tau_{ij} \leftarrow (1-\rho)\times\tau_{ij} + \frac{\rho}{L_{\psi^{gb}}}, \forall (i,j) \in \psi^{gb}$$

Procedure ACS-TIME(v)
/*Parameter v is minimum number of vehicles found by ACS-VEI for which a feasible solution has been found */
1.-/*Initialization*/
Initialize heuristic information using v
τ_{ij} <- pherecord_{ij}
2.-/*Cycle*/
Do
For each ant k
/*Build a solution ψ^k */
Tour_Construction(k, IN=0)
If #Visited_Customers(ψ^k) = |N| **Then**
If The last node is a depot **Then**
v <- v-1
ψ^{gb} <- ψ^k
Else
If $L_{\psi}{}^{k} < L_{\psi}{}^{gb}$ **Then**
ψ^{gb} <- ψ^k
End If
End If
End If
End For each
/*Global pheromone update*/
$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho/L_{\psi^{gb}}, \forall (i,j) \in \psi^{gb}$
While a stop criterion is met

Figure 4. ACS-TIME procedure.

Solution Model

The solution model for [MACS-VRPTWSL] is very similar to the one proposed by [16], where every *Ant* builds a single tour per time.

A solution is represented as follows:

1. The depots with all their connections to their customers are duplicated a number of times equal to the number of available vehicles.
2. Distances between copies of the depots are set to a number *M*, where *M* is a sufficiently big value.

Here, the routing problem is closer to the traditional [TSP] for the [ACS], where a feasible solution is a path where all the nodes have to be visited exactly once. According to [16] the advantage of such an approach is that the trails in direction to a copy of a depot are less attractive than the case with a single depot, due to the *pheromone* update.

Constructive Procedure

The *Constructive Procedure* for *ACS-VEI* and *ACS-TIME* (Figure 5) works in a very similar way. In this procedure an artificial ant is located randomly in a copy of the duplicated depots, and then an *ant* determines the set of feasible nodes to be visit according to the *constraint programming model* (if the vehicle capacity used is more than 25% then the set of feasible nodes also contains the depot not yet visited). An *Ant* located in *i* chooses the next node to visit from the set of feasible nodes N_i^k according to the *Pseudo Random Proportional Rule* defined in [ACS] [11]. The *Pseudo Random Proportional Rule* is defined as follows:

$$j = \begin{cases} \text{argmax}_{j \in N_i^k} \left\{ \tau_{ij} \cdot \left[\eta_{ij} \right]^{\beta} \right\}, & \text{if } q \le q_0; \\ J, & \text{otherwise;} \end{cases} \quad (17)$$

$$J = \begin{cases} \dfrac{\tau_{ij} \cdot \left[\eta_{ij} \right]^{\beta}}{\sum_{l \in N_i^k} \tau_{ij} \cdot \left[\eta_{il} \right]^{\beta}}, & \text{if } j \in N_i^k; \\ 0, & \text{otherwise;} \end{cases} \quad (18)$$

Where β and q_0 are parameters, β weighs the heuristic information η_{ij} importance and q_0 ($0 \le q_0 \le 1$) determine the probability of exploitation or exploration. q is a random variable uniform distributed in [0,1], where with a higher value of q_0, the probability of exploration is bigger, and J is a random variable selected according to the probability distribution given by equation (18). In other words with q_0 probability, an *ant* makes the best possible move as the *pheromone trails* and *heuristic information* indicates, while with probability ($1-q_0$) it performs an exploration of the arcs.

For [MACS-VRPTWSL] the *heuristic information* η_{ij} is calculated as presented in [16] for [VRPTW] problems, where it takes into consideration travel time between nodes t_{ij}, time windows $[a_j, b_j]$ associated to node *j*, and the number of times IN_j the node *j* has not been inserted into a solution (when the procedure is called by *ACS-TIME*, the variables *IN* are not used and the value is set to zero).

This is calculated as follows:

$\forall j \in N_i^k$

$Delivery\_Time_j$ <- max($Time + t_{ij}$, a_j)
$Delta_{ij}$ <- $Delivery\_Time_j - Time$
$Distance_{ij}$ <- $Delta_{ij}$ (b_j - $Time$)
$Distance_{ij}$ <- max(1.0, ($Distance_{ij} - IN_j$))
η_{ij} <- 1.0/$Distance_{ij}$

In this way, *heuristic information* privileges the nodes where its time windows are closer to the current time and nodes less included in a solution. This is relevant for a problem where the vehicles departure has to be updated constantly due to the incorporation of a new customer in the route.

Every time an *ant* has chosen a new node to be visited it performs a *time update procedure*. Finally, each time an *ant* moves from one node to another, a local *pheromone trail* update is executed according to the following equation:

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

Where τ_0 is the initial pheromone value. A good value for τ_0 is $\tau_0 = \dfrac{1}{|N| \times L_{NN}}$, where L_{NN} is the objective function value calculated with the [NN] heuristic and N is the number of nodes.

/*Tour Construction Procedure*/

Procedure Tour_Construction(k, IN)

1.-/*Initialization*/
Locate Ant k in one of the duplicated depots i randomly

ψ^k <- <i>
Time <- 0
Cap_v <- 0
$Arrival\_Time_i$ <- 0
$Departure\_Time_i$ <- 0
$Accumulated\_Time_i$ <- 0

2.-/*Ant k build a tour: The tour is stored in ψ^k*/
Do
/*Starting from node i compute the set of feasible nodes N_i^k: if accumulated capacity from the last depot inserted is bigger than a 25% from available capacity or the set of feasible nodes is empty, then the set of feasible nodes also include not yet visited depots*/

N_i^k = feasible domain

$\forall j \in N_i^k$ Calculate heuristic information η_{ij}, as follows:

$Delivery\_Time_j$ <- max(Time + t_{ij}, a_j)
$Delta_{ij}$ <- $Delivery\_Time_j$ – Time
$Dinstance_{ij}$ <- $Delta_{ij}$ (b_j - Time)
$Distance_{ij}$ <- max(1.0, ($Distance_{ij}$ – IN_j))
η_{ij} <- 1.0/$Distance_{ij}$

If $\exists N_i^k$ **Then**
Choose probabilistically the next node j using η_{ij} in exploitation or exploration (equation (17)(18))
ψ^k <- ψ^k + <j>
Cap_v <- Cap_v + q_j
Time_Updating(ψ^k, Time, Accumulated_Time)
/*Local pheromone updating*/
$\tau_{ij} \leftarrow (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_0$
i <- j
End If
While $N_i^k \neq \{\}$

Figure 5. Tour Constructive procedure.

NUMERICAL RESULTS FOR MACS-VRPTWSL

In this section we present some instances for [MACS-VRPTWSL], and we show that this approach is effective to solve this kind of problem.

The algorithm was coded in C++ and ILOG/Solver 6.0 and run on an Intel Core Duo T2400 @ 1.83 GHz, 504 Mb RAM.

As far as we know there are not instances in the literature for [VRPTWSL], in this case [MACS-VRPTWSL] has been tested in a subset of 17 instances presented by [23] for the [VRPTW]. The instances we have chosen are problems with 25 nodes and belong to the class of C1 and C2 problems, where C is for *Clustered Customers*, whose Time Windows where generated based on a known solution. The set of type 1 problems has narrow time windows and small vehicle capacity, and set of type 2 has large time windows and large vehicle capacity. The service time and load time of every customer demand for every instance is 0.5 demand time unities. Results in Table 1 are obtained with the following parameters:

- $\beta = 2$
- m = 10
- $q_0 = 0.9$
- $\rho = 0.1$
- maximum number of iterations = 659
- maximum number of iterations without improvement = 10
- maximum computational time = 250 seconds

It is observable that for the *state transition rule* presented in equation (17) and (18) the importance level has been taken for the *pheromone levels* first and second on the *heuristic information*.

Table 1. Results for MACS-VRPTWSL.

Instances	Nº Cust.	Problem	Nº Veh	Tour Length	Time in sec.
1	25	C101	5	698	191.3
2	25	C102	4	540	190.3
3	25	C103	3	356	190.3
4	25	C104	3	345	186.9
5	25	C105	4	496	191.3
6	25	C106	5	727	191.3
7	25	C107	3	518	191.3
8	25	C108	3	415	191.3
9	25	C109	3	297	191.3
10	25	C201	2	408	214.7
11	25	C202	2	336	214.5
12	25	C203	2	330	214.6
13	25	C204	1	296	213.1
14	25	C205	1	362	214.7
15	25	C206	1	358	214.7
16	25	C207	2	382	214.5
17	25	C208	1	358	214.1

CONCLUSIONS

This paper introduces a mathematical formulation for the problem [VRPTWSL] based in the model presented by Toth and Vigo [25] for the [VRPTW], in which the assumption that every vehicle departs from the depot at time zero is eliminated. This model incorporates new variables and sequencing constraints due to the limited number of load resources, as well a multi-objective function.

An algorithm approach based in multi-ant colony system is designed and implemented, in which two objective functions are minimized: the number of vehicles and the total tour length.

A *Time Update* and *constrained procedures* are proposed to be implemented in the *Ants constructive procedure*, to perform the update of a new customer incorporated into the sub-route and the schedule that this action implies in the load resources and to determine the set of feasible nodes to be visited by the ant, respectively.

Even if it not possible to judge the results presented for [MACS-VRPTWSL], this paper shows that [MACS-VRPTWSL] is effective in the resolution of this kind of problem, and is a good first step for further analysis for the problems of vehicle routing problem with time windows and vehicle departure scheduling. Possible developments for [VRPTWSL] can incorporate a local search procedure or a more dominant use of constraint programming for the *time update procedure*. Also, one can formulate an algorithm that for a first step avoids the elimination of the time zero departure assumption and in a second step implements a local search procedure with constraint programming to make feasible solution in a similar way presented by [22].

Also, further analysis can take the case of multiple *load resources* to perform the demand load in the vehicle.
As [VRPTWSL] incorporates the vehicle departure scheduling, the algorithm presented in this paper has a direct application to real problems, in this way [VRPTWSL] can be taken as an important advance for practical vehicle routing problems.

ACKNOWLEDGEMENT

The first author has been partially supported by DIUC 205.097.009-1.0 of the University of Concepción, Chile. The second and fourth authors have been partially supported by DIN 10/2008 of Universidad Católica de la Santísima Concepción, Chile.

REFERENCES

[1] E. Aarts and J.K. Lenstra. “Local Search in Combinatorial Optimization”. Wiley Interscience Series in Discrete Mathematical and Optimization. UK. 1997.

[2] J. Bramel and D. Simchi-Levi. “On the effectiveness of set covering formulations for the vehicle routing problem with time windows”. Operations Research. Vol. 45, pp. 295-301. 1997.

[3] O. Bräysy and W. Dullaert. “A Fast Evolutionary Metaheuristic for the Vehicle Routing Problem with Time Windows”. International Journal on Artificial Intelligence Tools. Vol. 12 Nº 2, pp. 153-172. 2003.

[4] A. Colorni, M. Dorigo and V. Maniezzo. “Distributed optimization by ant colonies”. In: Varela F.J. and Bourgine P. (Eds.). Proceedings of the First European Conference on Artificial Life. Elsevier publishing, pp. 134 - 142. Paris, France. 1991.

[5] J. Cordeau, G. Desaulniers, J. Desrosiers, M.M. Solomon and F. Soumis. “Vrp with time windows”. In: Toth P. and Vigo D. (Eds.). The vehicle routing problem. Society for Industrial and Applied Mathematics monograph on discrete mathematics and applications, pp. 157–193. 2001.

[6] C. de Jong, G. Kant and A. Van Vlient. “On finding minimal route durations in the vehicle routing problem with multiple time windows”. Technical report, Manuscript. Department of Computer Science. Utrecht University. Netherlands. 1996.

[7] J.L. Deneubourg, S. Aron, S. Goss and J.M. Pasteels. “The self-organizing exploratory pattern of the argentine ant”. Journal of Insect Behavior. Vol. 3, pp. 159-168. 1990.

[8] M. Desrochers, J. Desrosiers, M.M. Solomon. “A new optimization algorithm for the vehicle routing problem with time windows”. Journal of Operations Research. Vol. 40, pp. 342-354. 1992.

[9] M. Desrochers and F. Soumis. “A generalized permanent labeling algorithm for the shortest path problem with time windows”. Society for Industrial and Applied Mathematics, pp. 157–193. 1988.

[10] J. Desrosiers, Y. Dumas, M.M. Solomon and F. Soumis. "Time constrained routing and scheduling". In: Ball M.O., Magnanti T.L., Monma C.L. and Nemhauser G.L. (Eds), Network Routing. Handbooks in Operations Research and Management Science. Vol. 8, pp. 35-139. 1995.

[11] M. Dorigo and L.M. Gambardella. "Ant colonies for the traveling salesman problem". BioSystems. Vol. 43 Nº 2, pp. 73-81. 1997a.

[12] M. Dorigo and L.M. Gambardella. "Ant colony system: A cooperative learning approach to the traveling salesman problem". IEEE Transactions on Evolutionary Computation. Vol. 1 Nº 1, pp. 53-66. 1997b.

[13] M. Dorigo and T. Stuzle. "Ant Colony Optimization". MIT Press, Massachussets Institute of Technology. Cambridge. 2004.

[14] D. Favaretto, E. Moretti and P. Pellegrini. "Ant colony system for a vrp with multiple time windows and multiple visits". Accepted for publication in Journal of Interdisciplinary Mathematics. 2006.

[15] M. Flood. "The traveling salesman problem". Operations Research.Vol. 4, pp. 61-75. 1956.

[16] L.M. Gambardella, E. Taillard and G. Agazzi. "A multiple ant colony system for vehicle routing problem with time windows". Technical Report IDSIA-06-99. IDSIA. Lugano, Switzerland. 1999.

[17] B. Golden and A. Assad. "Vehicle routing with time-window constraints". American Journal of Mathematical and Management Sciences. Vol. 6 Nº 3–4, pp. 251-260. 1986.

[18] B.L. Golden, E.A. Wasil, J.P. Kelly and I.M. Chao. "Metaheuristics in vehicle routing". In: Crainic T.G., Lapote C. (Eds). Fleet Management and Logistics, pp. 33-56. Kiuwer, Boston, MA. 1998.

[19] N. Kohl, J. Desrosiers, O.B.G. Madsen, M.M. Solomon and F. Soumis. "2-path cuts for the vehicle routing problem with time windows". Transportation Science. Vol. 33, pp. 101-116. 1999.

[20] G. Laporte. "The traveling salesman problem: an overview of exact and approximate algorithms". European Journal of Operational Research. Vol. 59, pp. 231-247. 1992.

[21] M. Montemanni, L.M. Gambardella, A.E. Rizzoli, A.V. Donati. "Ant colony system for a dynamic vehicle routing problem". Journal of Combinatorial Optimization. Vol. 10, pp. 327-343. 2005.

[22] P. Shaw. "Using constraint programming and local search methods to solve vehicle routing". Lecture Notes in Computer Science. Vol. 1520, pp. 417-431. 1998.

[23] M.M. Solomon. "Algorithms for the vehicle routing and scheduling problem with time windows constraints". Operations Research. Vol. 35, pp. 254-365. 1987.

[24] M.M. Solomon and J. Desrosiers. "Time window constrained routing and scheduling problems". Transportation science. Vol. 22 Nº 1, pp. 1-13. 1988.

[25] P. Toth and D. Vigo. "The Vehicle Routing Problem". Society for Industrial and Applied Mathematic, Siam Monographs on Discrete Mathematics and Applications. 2001.