MACS-VRPTW: A MULTIPLE ANT COLONY SYSTEM FOR VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS

Luca Maria Gambardella, Éric Taillard and Giovanni Agazzi
IDSIA, Corso Elvezia 36, 6900 Lugano, Switzerland
Tel +41 91 9119838
Fax +41 91 9119839
email: {luca,eric,agazzi}@idsia.ch.
http://www.idsia.ch

TECHNICAL REPORT IDSIA

IDSIA-06-99

LUGANO, SWITZERLAND, 1999

Abstract

MACS-VRPTW, an Ant Colony Optimization based approach useful to solve vehicle routing problems with time windows is presented. MACS-VRPTW is organized with a hierarchy of artificial ant colonies designed to successively optimize a multiple objective function: the first colony minimizes the number of vehicles while the second colony minimizes the traveled distances. Cooperation between colonies is performed by exchanging information through pheromone updating. We show that MACS-VRPTW is competitive with the best known existing methods both in terms of solution quality and computation time. Moreover, MACS-VRPTW improves some of the best solutions known for a number of problem instances in the literature.

Chapter 5

MACS-VRPTW: A MULTIPLE ANT COLONY SYSTEM FOR VEHICLE ROUTING PROBLEMS WITH TIME WINDOWS

5.1. Introduction

This chapter presents MACS-VRPTW, a Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. MACS-VRPTW is based on Ant Colony System (ACS) (Gambardella and Dorigo, 1996, Dorigo and Gambardella, 1997a, 1997b), and, more generally, on Ant Colony Optimization (ACO), a new metaheuristic approach inspired by the foraging behavior of real colonies of ants.

The basic ACO idea (see Chapter 2 in this book and (Dorigo, Di Caro and Gambardella, 1998) for a detailed description of the ACO metaheuristic) is that a large number of simple artificial agents are able to build good solutions to hard combinatorial optimization problems via low-level based communications. Real ants cooperate in their search for food by depositing chemical traces (pheromones) on the floor. An artificial ant colony simulates this behavior (Dorigo, Maniezzo and Colorni, 1991, 1996). Artificial ants cooperate by using a common memory that corresponds to the pheromone deposited by real ants. This artificial pheromone is one of the most important components of ant colony optimization and is used for constructing new solutions.

In the ACO metaheuristic, artificial pheromone is accumulated at run-time during the computation. Artificial ants are implemented as parallel processes whose role is to build problem solutions using a constructive procedure driven by a combination of artificial pheromone, problem data and a heuristic function used to evaluate successive constructive steps.

Recently, many ACO algorithms have been proposed to solve different types of combinatorial optimization problems. In particular, ACO algorithms have been shown to be very efficient when combined with specialized local search procedures to solve the symmetric and asymmetric traveling salesman problems (TSP/ATSP, Dorigo and Gambardella, 1997a, 1997b, Stützle, 1998, Stützle and Dorigo, 1999), the sequential ordering problem (SOP, Gambardella and Dorigo, 1997), the quadratic assignment problem (QAP, Gambardella, Taillard and Dorigo, 1999, Taillard and Gambardella, 1997), the bi-quadratic assignment problem and the p-median problem (Taillard, 1998).

One of the most efficient ACO based implementations has been Ant Colony System (ACS), (Gambardella and Dorigo, 1996, Dorigo and Gambardella, 1997a, 1997b) that introduced a particular pheromone trail updating procedure useful to intensify the search in the neighborhood of the best computed solution. This chapter presents an ACS extension able to solve the vehicle routing problem with time windows (VRPTW).

VRPTW is defined as the problem of minimizing time and costs in case a fleet of vehicles has to distribute goods from a depot to a set of customers. The VRPTW considered in this chapter minimizes a multiple, hierarchical objective function: the first objective is to minimize the number of tours (or vehicles) and the second is to minimize the total travel time. A solution with a lower number of tours is always preferred to a solution with a higher number of tours even if the travel time is higher. This hierarchical objectives VRPTW is very common in the literature and in case problem constraints are very tight (for example when the total capacity of the minimum number of vehicles is very close to the total volume to deliver or when customers time windows are narrow), both objectives can be antagonistic: the minimum travel time solution can include a number of vehicles higher than the solution with minimum number of vehicles (see e.g. Kohl et al., 1997).

To adapt ACS for these multiple objectives the idea is to define two ACS colonies, each dedicated to the optimization of a different objective function. In MACS-VRPTW the colonies cooperate by exchanging information through pheromone updating. MACS-VRPTW is shown to be competitive with the best existing methods both in terms of solution quality and computation time. Moreover, MACS-VRPTW improves the best solutions known for some problem instances of the literature.

This chapter is organized as follows: First, vehicle routing problems are introduced by presenting a formal definition of the capacitated vehicle routing problem (CVRP) and the vehicle routing problem with time windows (VRPTW). Second, ACS main characteristics are analyzed by explaining how ACS has been applied to the traveling salesman problem (TSP). Then, ACS is extended to deal with VRPTW and the resulting MACS-VRPTW is investigated by presenting its main components. Last, numerical results are reported and some conclusions are drawn.

5.2 Vehicle Routing Problems

The most elementary version of the vehicle routing problem is the capacitated vehicle routing problem (CVRP). The CVRP is described as follows: n customers must be served from a unique depot. Each customer asks for a quantity q_i of goods (i = 1,..., n) and a vehicle of capacity Q is available to deliver goods. Since the vehicle capacity is limited, the vehicle has to periodically return to the depot for reloading. In the CVRP, it is not possible to split customer delivery. Therefore, a CVRP solution is a collection of tours where each customer is visited only once and the total tour demand is at most Q. From a graph theoretical point of view the CVRP may be stated as follows: Let $G = (C,L)$ be a complete graph with node set $C = (c_o, c_1, c_2,..., c_n)$ and arc set $L = (c_i, c_j)$: $c_i, c_j \in C$, $i \neq j$. In this graph model, c_o is the depot and the other nodes are the customers to be served. Each node is associated with a fixed quantity q_i of goods to be delivered (a quantity $q_o = 0$ is associated to the depot c_o). To each arc (c_i, c_j) is associated a value t_{ij} representing the travel time between c_i and c_j. The goal is to find a set of tours of minimum total travel time. Each tour starts from and terminates at the depot c_o, each node c_i (i = 1,..., n) must be visited exactly once, and the quantity of goods to be delivered on a route should never exceed the vehicle capacity Q.

One of the most successful exact approaches for the CVRP is the *k-tree* method of (Fisher, 1994) that succeeded in solving a problem with 71 customers. However, there are smaller instances that have not been exactly solved yet. To treat larger instances, or to compute solutions faster, heuristic methods must be used. Among the best heuristic methods are tabu searches (Taillard, 1993, Rochat and Taillard, 1995, Rego and Roucairol, 1996, Xu and Kelly, 1996, Toth and Vigo, 1998) and large neighborhood search (Shaw, 1998).

The CVRP can be extended in many ways. For example a service time s_i for each customer (with s_o = 0) and a time limit over the duration of each tour can be considered. The goal is again to search for a set of tours that minimizes the sum of the travel times. An important extension of the CVRP that is the subject of this chapter is the vehicle routing problem with time windows (VRPTW). In addition to the mentioned CVRP features, this problem includes, for the depot and for each customer c_i (i = 0,..., n) a time window [b_i, e_i] during which the customer has to be served (with b_0 the earliest start time and e_0 the latest return time of each vehicle to the depot). The tours are performed by a fleet of v identical vehicles. The additional constraints are that the service beginning time at each node c_i (i = 1,..., n) must be greater than or equal to b_i, the beginning of the time window, and the arrival time at each node c_i must be lower than or equal to e_i, the end of the time window. In case the arrival time is less than b_i, the vehicle has to wait till the beginning of the time window before starting servicing the customer.

In the literature the fleet size v is often a variable and a very common objective is to minimize v. This objective is related to the real situation in which driver salaries are variable costs for the company or when the company has to rent vehicles to perform deliveries.

Usually, two different solutions with the same number of vehicles are ranked by alternative objectives such as the total traveling time or total delivery time (including waiting and service times). These objectives are also used for companies owning a fixed fleet of vehicles.

A number of exact and heuristic methods exist for the VRPTW. Among exact methods, that of Kohl et al., 1997 is one of the most efficient and succeeded in solving a number of 100 customers instances. Note that exact methods are more efficient in case the solution space is restricted by narrow time windows since less combinations of customers are possible to define feasible tours. The most successful heuristic methods for the VRPTW are adaptive memory programs (see Taillard et al., 1998 for an introduction to adaptive memory programming), embedding tabu searches (Rochat and Taillard, 1995, Taillard et al., 1997, Badeau et al., 1997), guided local search (Kilby, Prosser and Shaw, 1999) and large neighborhood search (Shaw, 1998); note that the method of Taillard et al., 1997 can also be viewed as a kind of large neighborhood search.

5.3 Ant Colony System

This section introduces and presents the original Ant Colony System (ACS) (Gambardella and Dorigo, 1996, Dorigo and Gambardella, 1997a, 1997b) applied to the traveling salesman problem (TSP) Indeed, MACS-VRPTW has been proposed to solve a VRPTW where both the number of vehicles and the travel time have to be minimized. This multiple objective minimization is achieved by using two artificial ant colonies based on ACS.

The TSP is the problem of finding a shortest closed tour which visits all the cities in a given set. ACS is applied to the TSP by associating two measures to each arc of the TSP graph: the closeness η_{ij}, and the pheromone trail τ_{ij}. Closeness, defined as the inverse of the arc length, is a static heuristic value, that never changes for a given problem instance, while the pheromone trail is dynamically changed by ants at runtime. Therefore, the most important component of ACS is the management of pheromone trails which are used, in conjunction with the objective function, for constructing new solutions. Informally, pheromone levels give a measure of how desirable it is to insert a given arc in a solution. Pheromone trails are used for exploration and exploitation. Exploration concerns the probabilistic choice of the components used to construct a solution: a higher probability is given to elements with a strong pheromone trail. Exploitation chooses the component that maximizes a blend of pheromone trail values and heuristic evaluations.

ACS goal is to find a shortest tour. In ACS *m* ants build tours in parallel, where *m* is a parameter. Each ant is randomly assigned to a starting node and has to build a solution, that is, a complete tour. A tour is built node by node: each ant iteratively adds new nodes until all nodes have been visited. When ant *k* is located in node *i*, it chooses the next node *j* probabilistically in the set of feasible nodes $\mathcal{N}_i^k$ (i.e., the set of nodes that still have to be visited). The probabilistic rule used to construct a tour is the following: with probability q_0 a node with the highest $\tau_{ij}\cdot[\eta_{ij}]^{\beta}$, $j\in\mathcal{N}_i^k$ is chosen (exploitation), while with probability (1-q_0) the node *j* is chosen with a probability p_{ij} proportional to $\tau_{ij}\cdot[\eta_{ij}]^{\beta}$, $j\in\mathcal{N}_i^k$ (exploration, Equation 1).

$$p_{ij} = \begin{cases} \dfrac{\tau_{ij}\cdot[\eta_{ij}]^{\beta}}{\sum_{l\in N_i^k}\tau_{il}\cdot[\eta_{il}]^{\beta}} & \text{if } j\in N_i^k \\ 0 & \text{otherwise} \end{cases} \qquad (1)$$

where β and q_0 are parameters: β weighs the relative importance of the heuristic value, while q_0 ($0\le q_0\le 1$) determines the relative importance of exploitation versus exploration: the smaller q_0 the higher the probability to use the probabilistic rule described with Equation 1.

Once each ant has built a complete solution, this is tentatively improved using a local search procedure. Next, the best solution found from the beginning of the trial is used to update the pheromone trails. Then, the process is iterated by starting again *m* ants until a termination condition is met. ACS terminates when at least one of the following conditions becomes true: a fixed number of solutions has been generated, a fixed CPU time has elapsed, or no improvement has been achieved during a given number of iterations.

In ACS, pheromone trail is updated both locally and globally. Local updating is performed during solutions construction while global updating is performed at the end of the constructive phase. Shortly, the effect of local updating is to change dynamically the desirability of edges: every time an ant uses an edge the quantity of pheromone associated to this edge is decreased and the edge becomes less attractive. On the other side, global updating is used to intensify the search in the neighborhood of the best solution computed.

In ACS, only the best solution is used to globally modify the pheromone trail. This updating strategy (Gambardella and Dorigo, 1995, Gambardella and Dorigo, 1996, Dorigo and Gambardella, 1997a, 1997b) has been proved to be more efficient than the one used in Ant System (AS) (Dorigo, Maniezzo and Colorni, 1991, 1996) where all the constructed solutions are used to update the pheromone trails. The rationale is that in this way a "preferred route" is memorized in the pheromone trail matrix and future ants will use this information to generate new solutions in a neighborhood of this preferred route. The τ_{ij} are updated as follows:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho / J_{\psi}^{gb} \qquad \forall (i,j)\in\psi^{gb} \qquad (2)$$

where ρ ($0 \le \rho \le 1$) is a parameter and J_{ψ}^{gb} is the length of ψ^{gb}, the shortest path generated by ants since the beginning of the computation. This global updating procedure is applied at the end of each cycle, that is, each time the constructive phase has been completed.
Local updating is performed as follows: when an ant moves from node *i* to node *j*, the amount of pheromone trail on arc (*i*,*j*) is decrease according to the following rule:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_0 \qquad (3)$$

where τ_0 is the initial value of trails. It was found that $\tau_0 = 1/(n \cdot J_{\psi}^{h})$ is a good value for this parameter where J_{ψ}^{h} is the length of the initial solution produced by the nearest neighbor heuristic (Flood, 1956) and n is the number of nodes.
An interesting aspect of the local updating is that while edges are visited by ants, Equation 3 makes their trail diminish, making them less and less attractive, and favoring therefore the exploration of not yet visited edges and diversity in solution generation.

5.4 MACS-VRPTW for vehicle routing problems with time windows

Although VRPs are a relatively direct extensions of the TSP, the first ant system for vehicle routing problems has been designed only very recently by Bullnheimer et al. (1997, 1999) who considered the most elementary version of the problem: the capacitated vehicle routing problem (CVRP). This chapter considers a more elaborated vehicle routing problem with two objective functions: (i) the minimization of the number of tours (or vehicles) and (ii) the minimization of the total travel time, where number of tours minimization takes precedence over travel time minimization.

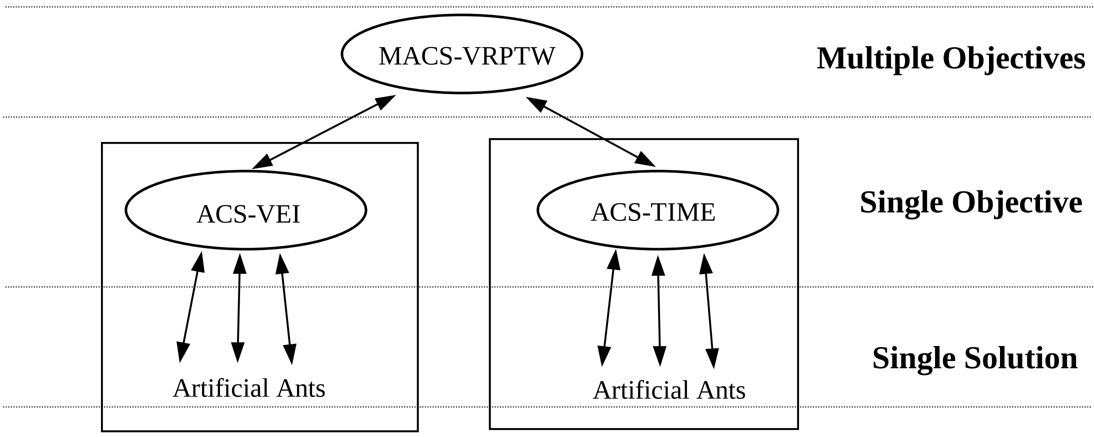

Figure 1. Architecture of the Multiple Ant Colony System for the Vehicle Routing Problem with Time Windows

```
/* MACS-VRPTW: Multiple Ant Colony System for Vehicle Routing Problems with Time Windows */
Procedure MACS-VRPTW()
1. /* Initialization */
    /* ψ^gb is the best feasible solution: lowest number of vehicles and shortest travel time
       #active_vehicles(ψ) computes the number of active vehicles in the feasible solution ψ */
    ψ^gb ← feasible initial solution with unlimited number of vehicles produced
           with a nearest neighbor heuristic
2. /* Main loop */
   Repeat
           v ← #active_vehicles(ψ^gb)
           Activate ACS-VEI(v - 1)
           Activate ACS-TIME(v)
           While ACS-VEI and ACS-TIME are active
               Wait an improved solution ψ from ACS-VEI or ACS-TIME
               ψ^gb ← ψ
               if #active_vehicles(ψ^gb) < v then
                          kill ACS-TIME and ACS-VEI
           End While
    until a stopping criterion is met
```

Figure 2. The MACS-VRPTW procedure

This hierarchical objectives VRPTW is very common in the literature but sometimes it is mixed up with other VRPTW variants that consider only one objective (for example, Kohl et al. 1997 consider instances built on the same data set as hierarchical objectives VRPTW instances, but with travel time as unique objective).

In the MACS-VRPTW algorithm (Figures 1 and 2) both objectives are optimized simultaneously by coordinating the activities of two ACS based colonies. The goal of the first colony, ACS-VEI, is to try to diminish the number of vehicles used, while the second colony, ACS-TIME, optimizes the feasible solutions found by ACS-VEI. Both colonies use independent pheromone trails but collaborate by sharing the variable ψ^{gb} managed by MACS-VRPTW. Initially, ψ^{gb} is a feasible VRPTW solution found with a nearest neighbor heuristic. Then, ψ^{gb} is improved by the two colonies. When ACS-VEI is activated, it tries to find a feasible solution with one vehicle less than the number of vehicles used in ψ^{gb}. The goal of ACS-TIME is to optimize the total travel time of solutions that use as many vehicles as vehicles used in ψ^{gb}. ψ^{gb} is updated each time one of the colonies computes an improved feasible solution. In case the improved solution contains less vehicles than the vehicles used in ψ^{gb}, MACS-VRPTW kills ACS-TIME and ACS-VEI. Then, the process is iterated and two new colonies are activated, working with the new, reduced number of vehicles.

5.4.1 ACS-TIME and ACS-VEI colonies

The working principles of ACS-VEI and ACS-TIME colonies are described in Figure 3 and Figure 4.

```
/* ACS-TIME: Travel time minimization. */
Procedure ACS-TIME(v)
/* Parameter v is the smallest number of vehicles for which a feasible solution has been computed */
1. /* Initialization */
   initialize pheromone and data structures using v
2. /* Cycle */
   Repeat
         for each ant k
            /* construct a solution ψ^k */
            new_active_ant(k, local_search=TRUE, 0)
         end for each
         /* update the best solution if it is improved */
         if ∃ k : ψ^k is feasible and J_ψ^k < J_ψ^gb then
                send ψ^k to MACS-VRPTW
         /* perform global updating according to Equation 2*/
         τ_ij = (1−ρ)·τ_ij + ρ/J_ψ^gb        ∀(i, j) ∈ ψ^gb
   until a stopping criterion is met
```

Figure 3. The ACS-TIME procedure

ACS-TIME colony (Figure 3) is a traditional ACS based colony whose goal is to compute a tour as short as possible. In ACS-TIME *m* artificial ants are activated to construct problems solutions $\psi^1,...,\psi^m$. Each solution is build by calling the *new_active_ant* procedure, a constructive procedure explained in details in Section 5.4.3 that is similar to the ACS constructive procedure designed for the TSP. When $\psi^1,...,\psi^m$ have been computed, they are compared to ψ^{gb} and, in case one solution is better, it is sent to MACS-VRPTW. MACS-VRPTW uses this solution to update ψ^{gb}. After solutions generation, the global updates are performed using Equation 2 and ψ^{gb}.

ACS-VEI colony (Figure 4) searches for a feasible solution by maximizing the number of visited customers. ACS-VEI starts its computation using *v*-1 vehicles, that is, one vehicle less than the smallest number of vehicles for which a feasible solution has been computed (i.e., the number of vehicles in ψ^{gb}). During this search the colony produces unfeasible solutions in which some customers are not visited. In ACS-VEI, the solution computed since the beginning of the trial with the highest number of visited customers is stored in the variable $\psi^{\text{ACS-VEI}}$. A solution is better than $\psi^{\text{ACS-VEI}}$ only when the number of visited customers is increased. Therefore ACS-VEI is different from the traditional ACS applied to the TSP. In ACS-VEI the current best solution $\psi^{\text{ACS-VEI}}$ is the solution (usually unfeasible) with the highest number of visited customers, while in ACS applied to TSP the current best solution is the shortest one.

```
/* ACS-VEI: Number of vehicles minimization. */
Procedure ACS-VEI(s)
  /* Parameter s is set to v-1, that is, one vehicle less than the smallest number of vehicles for which a feasible solution
     has been computed
    #visited_customers(ψ) computes the number of customers that have been visited in solution ψ */
1. /* Initialization */
    initialize pheromone and data structures using s
    ψ^ACS-VEI ← initial solution with s vehicles produced with a nearest neighbor
            heuristic. /* ψ^ACS-VEI is not necessary feasible */
2. /* Cycle */
     Repeat
           for each ant k
               /* construct a solution ψ^k */
               new_active_ant(k,local_search=FALSE,IN)
               ∀ customer j ∉ ψ^k : IN_j ← IN_j + 1
           end for each
           /* update the best solution if it is improved */
           if ∃ k : #visited_customers(ψ^k) > #visited_customers(ψ^ACS-VEI) then
                  ψ^ACS-VEI ← ψ^k
                  ∀ j: IN_j ← 0 /* reset IN */
                  if ψ^ACS-VEI is feasible then
                     send ψ^ACS-VEI to MACS-VRPTW
           /* perform global updating according to Equation 2 using both ψ^ACS-VEI and ψ^gb */
```

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho / J_{\psi}^{ACS\text{-}VEI} \qquad \forall (i,j) \in \psi^{ACS\text{-}VEI}$$

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho / J_{\psi}^{gb} \qquad \forall (i,j) \in \psi^{gb}$$

```
     until a stopping criterion is met
```

Figure 4. The ACS-VEI procedure

In order to maximize the number of customers serviced, ACS-VEI manages a vector *IN* of integers. The entry IN_j stores the number of time customer *j* has not been inserted in a solution. *IN* is used by the constructive procedure *new_active_ant* for favoring the customers that are less frequently included in the solutions. In ACS-VEI, at the end of each cycle, pheromone trails are globally updated with two different solutions: $\psi^{ACS-VEI}$, the unfeasible solution with the highest number of visited customers and ψ^{gb}, the feasible solution with the lowest number of vehicles and the shortest travel time. Numerical experiments have shown that a double update greatly improves the system performances. Indeed, the updates with $\psi^{ACS-VEI}$ are not increasing the trails toward the customers that are not included in the solution. Since ψ^{gb} is feasible, the updates with ψ^{gb} are increasing trails toward all customers.

5.4.2 Solution model

MACS-VRPTW uses a solution model in which each ant builds a single tour (Figure 5). A solution is represented as follows: First, the depot with all its connections to/from the customers is duplicated a number of times equal to the number of available vehicles. Distances between copies of the depot are set to zero. This approach makes the vehicle routing problem closer to the traditional traveling salesman problem.

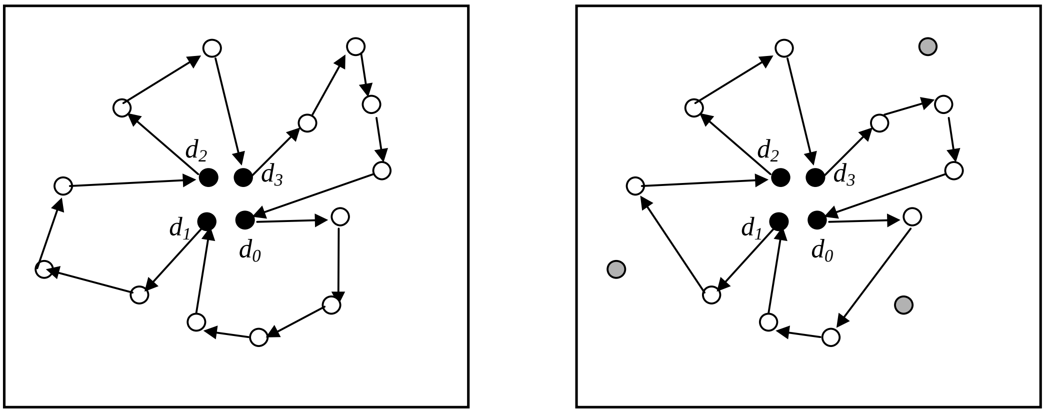

Figure 5. Feasible (left) and unfeasible (right) solutions for a vehicle routing problem with four duplicated depots and four active vehicles

So, both in the TSP and in this model a feasible solution is a path that visits all the nodes exactly once. Figure 5 shows a vehicle routing problem solution represented as a single tour. Duplicated depots (d_0,...,d_3) are black points while clients are white points. All duplicated depots have the same coordinates but they have been split to clarify the picture. An advantage of such a solution representation is that the trails in direction of the duplicated depots are less attractive than in case of a single depot (due to the pheromone update rules). This positively affects the quality of the solutions produced by the constructive procedure.

5.4.3 Solution constructive procedure

ACS-VEI and ACS-TIME use the same *new_active_ant* constructive procedure that is presented in details in Figure 6. This constructive procedure is similar to the ACS constructive procedure designed for the TSP: Each artificial ant starts from a randomly chosen copy of the depot and, at each step, moves to a not yet visited node that does not violate time window constraints and vehicle capacities. The set of available nodes, in case the ant is not located in a duplicated depot, also includes not yet visited duplicated depots. An ant positioned in node i chooses probabilistically the next node j to be visited by using exploration and exploitation mechanisms. The attractiveness η_{ij} is computed by taking into account the traveling time t_{ij} between nodes i and j, the time window $[b_j,e_j]$ associated to node j and the number of times IN_j node j has not been inserted in a problem solution. When the *new_active_ant* is called by ACS-TIME, the variables *IN* are not used and the corresponding parameter is set to zero.

```
/* new_active_ant: constructive procedure for ant k used by ACS-VEI and ACS-TIME */
Procedure new_active_ant(k, local_search, IN)
1. /* Initialization*/
        put ant k in a randomly selected duplicated depot i
        ψ^k ← ⟨i⟩
        current_time_k ← 0, load_k ← 0
2. /* This is the step in which ant k builds its tour. Tour is stored in ψ^k */
    Loop
        /* Starting from node i compute the set N_i^k of feasible nodes (i.e., all the nodes j still to be visited and such that
          current_time_k and load_k are compatible with time windows [b_j,e_j] and delivery quantity q_j of customer j)
          ∀j ∈ N_i^k compute the attractiveness η_ij as follows: */
                delivery_time_j ← max(current_time_k + t_ij, b_j)
                delta_time_ij ← delivery_time_j - current_time_k
                distance_ij ← delta_time_ij *( e_j - current_time_k)
                distance_ij ← max(1.0, (distance_ij - IN_j))
                η_ij ← 1.0/ distance_ij
        Choose probabilistically the next node j using η_ij in exploitation
        and exploration (Equation 1) mechanisms
        ψ^k ← ψ^k + ⟨j⟩
        current_time_k ← delivery_time_j
        load_k ← load_k + q_j
        If j is a depot then current_time_k ← 0, load_k ← 0
        τ_ij ← (1-ρ)· τ_ij + ρ· τ_0 /* Local pheromone updating (Equation 3) */
        i ← j /* New node for ant k */
    Until N_i^k = {} /* no more feasible nodes are available */
3. /* In this step path ψ^k is extended by tentatively inserting non visited customers */
    ψ^k ← insertion_procedure(ψ^k)
4. /* In this step feasible paths are optimized by a local search procedure.
    The parameter local_search is TRUE in ACS-TIME and it is FALSE in ACS-VEI*/
    if local_search = TRUE and ψ^k is feasible then
        ψ^k ← local_search_procedure(ψ^k)
```

Figure 6. The new_active_ant constructive procedure used by ACS-VEI and ACS-TIME

Each time an ant moves from one node to another, a local update of the pheromone trail is executed according to Equation 3. Last, at the end of the constructive phase, the solution might be incomplete (some customers might have been omitted) and the solution is tentatively completed by performing further insertions. The insertion is executed by considering all the non visited customers sorted by decreasing delivery quantities. For each customer it is searched for the best feasible insertion (shortest travel time) until no further feasible insertion is possible.

In addition, ACS-TIME implements a local search procedure to improve the quality of the feasible solutions. The local search uses moves similar to CROSS exchanges (Taillard et al., 1997). This procedure is based on the exchange of two sub-chains of customers. One of this sub-chain may eventually be empty, implementing a more traditional customer insertion.

5.5 Numerical results

This section reports computational results showing the efficiency of MACS-VRPTW. MACS-VRPTW has been tested on a classical set of 56 benchmark problems (Solomon, 1987) composed of six different problem types (C1,C2,R1,R2,RC1,RC2). Each data set contains between eight to twelve 100-node problems. The names of the six problem types have the following meaning. Sets C have clustered customers whose time windows were generated based on a known solution. Problem sets R have customers location generated uniformly randomly over a square. Sets RC have a combination of randomly placed and clustered customers. Sets of type 1 have narrow time windows and small vehicle capacity. Sets of type 2 have large time windows and large vehicle capacity. Therefore, the solutions of type 2 problems have very few routes and significantly more customers per route.

Experiments are made by executing, for each problem data, 3 runs that are stopped after a fixed computation time. Solutions are then averaged for each problem type and the result is reported in the tables. Experiments have been done with the following parameter settings: m=10 ants, q_0=0.9, β=1 and ρ=0.1. The code was written in C++.

Table 1. Performance comparison among the best VRPTW algorithms for different computational time (in seconds). RT=Rochat and Taillard (1995), SW = Shaw (1998), KPS = Kilby, Prosser and Shaw (1999), CW = Cordone and Wolfler-Calvo (1998), TB= Taillard et al. (1997)

	R1			C1			RC1			R2			C2			RC2		
	VEI	DIST	TIME	VEI	DIST	TIME	VEI	DIST	TIME	VEI	DIST	TIME	VEI	DIST	TIME	VEI	DIST	TIME
	12.55	1214.80	100	10.00	828.40	100	12.46	1395.47	100	3.05	971.97	100	3.00	593.19	100	3.38	1191.87	100
MACS-	12.45	1212.95	300	10.00	828.38	300	12.13	1389.15	300	3.00	969.09	300	3.00	592.97	300	3.33	1168.34	300
VRPTW	12.38	1213.35	600	10.00	828.38	600	12.08	1380.38	600	3.00	965.37	600	3.00	592.89	600	3.33	1163.08	600
	12.38	1211.64	1200	10.00	828.38	1200	11.96	1385.65	1200	3.00	962.07	1200	3.00	592.04	1200	3.33	1153.63	1200
	12.38	1210.83	1800	10.00	828.38	1800	11.92	1388.13	1800	3.00	960.31	1800	3.00	591.85	1800	3.33	1149.28	1800
	12.83	1208.43	450	10.00	832.59	540	12.75	1381.33	430	3.18	999.63	1600	3.00	595.38	1200	3.62	1207.37	1300
RT	12.58	1202.00	1300	10.00	829.01	1600	12.50	1368.03	1300	3.09	969.29	4900	3.00	590.32	3600	3.62	1155.47	3900
	12.58	1197.42	2700	10.00	828.45	3200	12.33	1269.48	2600	3.09	954.36	9800	3.00	590.32	7200	3.62	1139.79	7800
	12.45	1198.37	900				12.05	1363.67	900									
SW	12.35	1201.47	1800				12.00	1363.68	1800									
	12.33	1201.79	3600				11.95	1364.17	3600									
KPS	12.67	1200.33	2900	10.00	830.75	2900	12.12	1388.15	2900	3	966.56	2900	3.00	592.29	2900	3.38	1133.42	2900
CW	12.50	1241.89	1382	10.00	834.05	649	12.38	1408.87	723	2.91	995.39	1332	3.00	591.78	292	3.38	1139.70	946
	12.64	1233.88	2296	10.00	830.41	2926	12.08	1404.59	1877	3.00	1046.56	3372	3.00	592.75	3275	3.38	1248.34	1933
TB	12.39	1230.48	6887	10.00	828.59	7315	12.00	1387.01	5632	3.00	1029.65	10116	3.00	591.14	8187	3.38	1220.28	5798
	12.33	1220.35	13774	10.00	828.45	14630	11.90	1381.31	11264	3.00	1013.35	20232	3.00	590.91	16375	3.38	1198.63	11596

Table 1 compares MACS-VRPTW with a number of the best methods available for the VRPTW. The methods considered are: the adaptive memory programming of Rochat and Taillard, 1995 (RT), the large neighbourhood search of Shaw, 1998 (SW), the guided local search of Kilby, Prosser and Shaw, 1999 (KPS), the alternate K-exchange Reduction of Cordone and Wolfler-Calvo, 1998 (CW) and the adaptive memory programming of Taillard et al., 1997 (TB). Table 1 provides 3 columns for each data set: the average number of vehicles (main goal), the average tour length and the computation time (in seconds). The computational times cannot be directly compared for different reasons. First, the authors have used different computers; second, some methods (RT and TB) were designed to solve harder problems than the VRPTW and implementations specifically designed for the VRPTW might be faster.

Table 2. Average of the best solutions computed by different VRPTW algorithms. Best results are in boldface. RT=Rochat and Taillard (1995), TB= Taillard et al. (1997), CR=Chiang and Russel (1993), PB=Potvin and Bengio (1996), TH= Thangiah et al. (1994)

	R1		C1		RC1		R2		C2		RC2	
	VEI	DIST	VEI	DIST	VEI	DIST	VEI	DIST	VEI	DIST	VEI	DIST
MACS-VRPTW	**12.00**	1217.73	**10.00**	**828.38**	11.63	1382.42	**2.73**	967.75	3.00	**589.86**	**3.25**	1129.19
RT	12.25	1208.50	**10.00**	**828.38**	11.88	1377.39	2.91	961.72	3.00	**589.86**	3.38	1119.59
TB	12.17	1209.35	**10.00**	**828.38**	**11.50**	1389.22	2.82	980.27	3.00	**589.86**	3.38	1117.44
CR	12.42	1289.95	10.00	885.86	12.38	1455.82	2.91	1135.14	3.00	658.88	3.38	1361.14
PB	12.58	1296.80	10.00	838.01	12.13	1446.20	3.00	1117.70	3.00	589.93	3.38	1360.57
TH	12.33	1238.00	10.00	832.00	12.00	1284.00	3.00	1005.00	3.00	650.00	3.38	1229.00

MACS-VRPTW was executed on a Sun UltraSparc 1 167MHz, 70 Mflop/s (Dongarra, 1997), RT used a 15 Mflop/s Silicon Graphics computer, SW used a 63 Mflop/s Sun UltraSparc, KPS used a 25Mflops/s DEC Alpha, CW used a 18 Mflop/s Pentium and TB used a 10 Mflop/s Sun Sparc 10. Table 1 provides computational results for MACS-VRPTW when stopped after 100, 300, 600, 1200 and 1800 seconds. The RT, SW and TB iterative methods were also stopped after different computation times while KPS and CW provide results for a unique computation time.

In Table 1 is shown that MACS-VRPTW is very competitive: for C1 and RC2 types it is clearly the best method and it is always among the best methods for the other problem sets. A characteristic of MACS-VRPTW is that it is able to produce relatively good solutions in a short amount of time.

Table 2 reports the average of the best solutions obtained in all our experiments. Similar results were also provided by other authors. In addition to the methods of Rochat and Taillard, 1995 (RT) and Taillard et al., 1997 (TB) already compared in Table 1, Table 2 includes the results of the hybrid method of Chiang and Russel (CR, 1993), the genetic algorithm of Potvin and Bengio (PB, 1996) and the hybrid method of Thangiah et al. (TH, 1994). With the exception of RC1 type problem, MACS-VRPTW has been able to produce the best results for all other problem types.

During this experimental campaign, the best solution known of a number of problem instances have been improved. The value of these new best solutions are reported in Table 3. In addition to the VRPTW instances the ACS-TIME colony has been tested on CVRP instances. In Table 3 are also reported new best solution value for CVRP problem instances tai*nnn* used in Rochat and Taillard (1995), where *nnn* stands for the number of customers.

Table 3. New best solution values computed by MACS-VRPTW.
RT=Rochat and Taillard (1995), S = Shaw (1998), TB= Taillard et al. (1997)

	Old Best			New Best	
Problem	**source**	**vehicles**	**length**	**vehicles**	**length**
r112.dat	RT	10	953.63	**9**	982.140
r201.dat	S	4	1254.09	4	**1253.234**
r202.dat	TB	3	1214.28	3	**1202.529**
r204.dat	S	2	867.33	2	**856.364**
r207.dat	RT	3	814.78	**2**	894.889
r208.dat	RT	2	738.6	2	**726.823**
r209.dat	S	3	923.96	3	**921.659**
r210.dat	S	3	963.37	3	**958.241**
rc202.dat	S	4	1162.8	**3**	1377.089
rc203.dat	S	3	1068.07	3	**1062.301**
rc204.dat	S	3	803.9	3	**798.464**
rc207.dat	S	3	1075.25	3	**1068.855**
rc208.dat	RT	3	833.97	3	**833.401**
tai100a.dat	RT	11	2047.90	11	**2041.336**
tai100c.dat	RT	11	1406.86	11	**1406.202**
tai100d.dat	RT	11	1581.25	11	**1581.244**
tai150b.dat	RT	14	2727.77	14	**2656.474**

5.6 Conclusions

This chapter introduced MACS-VRPTW, a new Ant Colony Optimization based approach to solve vehicle routing problems with time windows. In particular, MACS-VRPTW has been designed to solve vehicle routing problems with two objective functions: (i) the minimization of the number of tours (or vehicles) and (ii) the minimization of the total travel time, where number of tours minimization takes precedence over travel time minimization. MACS-VRPTW introduces a new methodology for optimizing multiple objective functions. The basic idea is to coordinate the activity of different ant colonies, each of them optimizing a different objective. These colonies work by using independent pheromone trails but they collaborate by exchanging information. This is the first time a multi-objective function minimization problem is solved with a multiple ant colony optimization algorithm.

MACS-VRPTW is shown to be competitive with the best existing methods both in terms of solution quality and computation time and has been able to improve the best solutions known for a number of problem instances of the literature.

Acknowledgments

This research has been partially supported by the Swiss National Science Foundation project “Adaptive Memory Programming for Dynamic Optimization Problems”.

References

P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, É. D. Taillard, A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows, *Transportation Research-C* 5, 1997, 109-122.

B. Bullnheimer, R. F. Hartl, C. Strauss, An Improved Ant System Algorithm for the Vehicle Routing Problem, Technical Report POM-10/97, Institute of Management Science, University of Vienna, Austria, 1997. Accepted for publication in *Annals of Operations Research*.

B. Bullnheimer, R. F. Hartl, C. Strauss, Applying the Ant System to the Vehicle Routing Problem, in *Meta-heuristics: Advances and Trends in Local Search for Optimization*, S.Voss, S. Martello, I.H. Osman and C. Roucairol (eds.), Kluwer Academic Publishers, Boston, 1999, 285-296.

W. C. Chiang, R. Russel, Hybrid Heuristics for the Vehicle Routing Problem with Time Windows, Working Paper, Department of Quantitative Methods, University of Tulsa, OK, USA, 1993.

R. Cordone, R. Wolfler-Calvo, A Heuristic for the Vehicle Routing Problem with Time Windows. To appear in *Journal of Heuristics*.

J. J. Dongarra, Performance of Various Computers Using Standard Linear Equations Software, Technical Report CS-89-85, Computer Science Department, University of Tennesse, USA, 1997.

M. Dorigo, V. Maniezzo, A. Colorni, Positive Feedback as a Search Strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1991.

M. Dorigo, V. Maniezzo, A. Colorni, The Ant System: Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics–Part B* 26, 1996, 29-41.

M. Dorigo, G. Di Caro, L. M. Gambardella, Ant Algorithms for Discrete Optimization, Technical Report IRIDIA/98-10, Université Libre de Bruxelles, Belgium, 1998. Accepted for publication in *Artificial Life.*

M. Dorigo, L. M. Gambardella, Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, *IEEE Transactions on Evolutionary Computation* 1, 1997a, 53-66.

M. Dorigo, L. M. Gambardella, Ant Colonies for the Traveling Salesman Problem, *BioSystems* 43, 1997b, 73-81.

M. L. Fisher, Optimal Solution of Vehicle Routing Problems Using Minimum K-trees, *Operations Research* 42, 1994, 626-642.

M. M. Flood, The Traveling Salesman Problem, *Operations Research* 4, 1956, 61-75.

L. M. Gambardella, M. Dorigo, Ant-Q: a Reinforcement Learning Approach to the Traveling Salesman Problem, *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, A. Prieditis and S. Russell (eds.), Morgan Kaufmann, 1995, 252–260.

L. M. Gambardella, M. Dorigo, Solving Symmetric and Asymmetric TSPs by Ant Colonies, *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC96*, IEEE Press, 1996, 622-627.

L. M. Gambardella, M. Dorigo, HAS-SOP: Hybrid Ant System for the Sequential Ordering Problem, Technical Report IDSIA-11-97, IDSIA, Lugano, Switzerland, 1997.

L. M. Gambardella, É. D. Taillard, M. Dorigo, Ant Colonies for the Quadratic Assignment Problems, *Journal of Operational Research Society*, 50, 1999, 167-176.

P. Kilby, P. Prosser, P. Shaw, Guided Local Search for the Vehicle Routing Problems With Time Windows, in *Meta-heuristics: Advances and Trends in Local Search for Optimization*, S.Voss, S. Martello, I.H. Osman and C.Roucairol (eds.), Kluwer Academic Publishers, Boston, 1999, 473-486.

N. Kohl, J. Desrosiers, O. B. G. Madsen, M. M. Solomon, F. Soumis, K-Path Cuts for the Vehicle Routing Problem with Time Windows, Technical Report IMM-REP-1997-12, Technical University of Denmark, 1997.

J.-Y. Potvin, S. Bengio, The Vehicle Routing Problem with Time Windows - Part II: Genetic Search, *INFORMS Journal of Computing* 8, 1996, 165-172.

C. Rego, C. Roucairol, A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem, in *Meta-heuristics: Theory and applications*, I.H. Osman, J. Kelly (eds.), Kluwer Academic Publishers, Boston, 1996, 661-675.

Y. Rochat, É. D. Taillard, Probabilistic Diversification and Intensification in Local Search for Vehicle Routing, *Journal of Heuristics* 1, 1995, 147-167.

M. Solomon, Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints, *Operations Research* 35, 1987, 254-365.

P. Shaw, Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, *Proceedings of the Fourth International Conference on Principles and Practice of Constraint Programming* (CP '98), M. Maher and J.-F. Puget (eds.), Springer-Verlag, 1998, 417-431.

T. Stützle, Local Search Algorithms for Combinatorial Problems - Analysis, Improvements, and New Applications, *PhD Thesis*, Intellectics Group, Department of Computer Science, Darmstadt University of Technology, Germany, 1998.

T. Stützle, M. Dorigo, ACO Algorithms for the Traveling Salesman Problem, *Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications*, P. Neittaanmaki, J. Periaux, K. Miettinen and M. Makela, (eds.), John Wiley & Sons, 1999.

É. D. Taillard, Parallel Iterative Search Methods for Vehicle Routing Problems, *Networks* 23, 1993, 661-673.

É. D. Taillard, FANT: Fast Ant System, Technical Report IDSIA-46-98, IDSIA, Lugano, Switzerland, 1998.

É. D. Taillard, P. Badeau, M. Gendreau, F. Guertin, J.-Y. Potvin, A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows, *Transportation Science* 31, 1997, 170-186.

É. D. Taillard, L. M. Gambardella, Adaptive Memories for the Quadratic Assignment Problem, Technical Report IDSIA-87-97, IDSIA, Lugano, Switzerland, 1997.

É. D. Taillard, L. M. Gambardella, M. Gendreau, J.-Y. Potvin, Adaptive Memory Programming: A Unified View of Meta-Heuristics, Technical Report IDSIA-19-98, IDSIA, Lugano, Switzerland, 1998. Also published in EURO XVI Conference Tutorial and Research Reviews booklet (semi-plenary session), Brussels, July 1998.

S. R. Thangiah, I. H. Osman, T. Sun, Hybrid Genetic Algorithm Simulated Annealing and Tabu Search Methods for Vehicle Routing Problem with Time Windows, Technical Report 27, Computer Science Department, Slippery Rock University, 1994.

P. Toth, D. Vigo, The Granular Tabu Search (and its Application to the Vehicle Routing Problem), Technical Report, Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Italy, 1998.

J. Xu, J. Kelly, A Network Flow-Based Tabu Search Heuristic for the Vehicle Routing Problem, *Transportation Science* 30, 1996, 379-393.