A REVISED ANT COLONY SYSTEM APPROACH TO VEHICLE ROUTING PROBLEMS

by

ELİF İLKE GÖKÇE

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of
the requirements for the degree of
Master of Science

Sabancı University
July 2004

A REVISED ANT COLONY SYSTEM APPROACH TO VEHICLE ROUTING PROBLEMS

APPROVED BY:

Assistant Prof. Dr. Bülent Çatay (Thesis Supervisor)	…………………………
Assistant Prof. Dr. Hüsnü Yenigün	……………..………..
Assistant Prof. Dr. Kemal Kılıç	…..………………………
Assistant Prof. Dr. Kerem Bülbül	…..…………………..…
Assistant Prof. Dr. Tonguç Ünlüyurt	…..………………………

DATE OF APPROVAL: ………………………….

ACKNOWLEDGEMENTS

I would like to thank my thesis advisor Assistant Prof Dr. Bülent Çatay for his encouragement, motivation and considerable time he spent from beginning to end of my thesis.

I thank to graduate committee members of my thesis. Assistant Prof. Dr. Kemal Kılıç, Assistant Prof. Dr. Kerem Bülbül and Assistant Prof. Dr. Hüsnü Yenigün for their worthwhile suggestions and remarks.

ABSTRACT

Vehicle routing problems have various extensions such as time windows, multiple vehicles, backhauls, simultaneous delivery and pick-up, etc. The objectives of all these problems are to design optimal routes minimizing total distance traveled, minimizing number of vehicles, etc that satisfy corresponding constraints.

In this study, an ant colony optimization based heuristic that can be used to solve various vehicle routing problems is proposed. The objective function considered to minimize the total distance traveled by all vehicles. The heuristic is applied to vehicle routing problem with time windows and vehicle routing with simultaneous delivery and pick-up. Vehicles are identical and capacities of the vehicles are finite. The time window constraints in the first problem are assumed to be strict.

The proposed heuristic consists of four steps. First, a candidate list is formed for each customer in order to reduce computational time. Second, a feasible solution is found, and initial pheromone trails on each arc is calculated using it. Then, routes are constructed based on Dorigo *et al.* (1997). While visibility is calculated during route construction process, the distance between two customers, customers' distance to the depot and the time window associated with the customer to whom the ant is considered to move are considered. Pheromone trails are modified by both local and global pheromone update. Finally, constructed routes are improved using 2-opt algorithm.

The algorithm have been tested on the benchmark problem instances of Solomon (1987) for vehicle routing problem with time windows, and benchmark problem instances of Min (1989) and Dethloff (2001) for vehicle routing with simultaneous delivery and pick-up. The algorithm is proven to give good results when compared to the best known results in the literature.

ÖZET

Araç Rotalama Problemlerinin zaman kısıtı, değişik özellikli araçlar, eşzamanlı dağıtım ve toplama gibi çok çeşitli uzantıları vardır. Bütün problemdeki amaç ise tüm kısıtları sağlayan kat edilen toplam mesafeyi, kullanılan araç sayısını vs. azaltan optimal rotalar oluşturmaktır.

Bu çalışmada çeşitli Araç Rotalama Problemlerinin çözümü için kullanılabilecek karınca kolonisi optimizasyonuna dayanan bir sezgisel yaklaşım önerilmiştir. Modeldeki amaç fonksiyonu, araçlar tarafından kat edilen toplam mesafenin en küçüklenmesidir. Önerilen yaklaşım Zaman Kısıtlı Araç Rotalama Problemine ve Eş Zamanlı Dağıtım ve Toplamalı Araç Rotalama Problemine uygulanmıştır. Tüm araçlar aynı özelliklere sahiptir ve araçların kapasiteleri göz önünde bulundurulmaktadır.

Önerilen sezgisel yöntem dört aşamadan oluşmaktadır. Ilk olarak hesaplama zamanını azaltmak için aday listeleri oluşturulur. İkinci olarak olurlu bir çözüm bulunur ve bu çözüm kullanılarak her bir yol üzerindeki başlangıç feromen seviyeleri hesaplanır. Daha sonra Dorigo (1997) tarafından önerilen yönteme dayanılarak rotalar oluşturulur. Rotaların oluşturulması sırasında uygunluk hesaplanırken müşteriler arasındaki uzaklık, müşterilerin depoya olan uzaklıkları ve zaman kısıtı göz önünde bulundurulur. Feromen seviyeleri ise hem yerel hemde global feromen yenileme yontemleri ile değiştirilir. Son olarak oluşturulan rotalar 2-opt algoritmasi kullanılarak iyileştirilir.

Algoritma, zaman kısıtlı araç rotalama problemi için Solomon'un 1987 yılında oluşturduğu kıyaslama problemi örnekleri ile, eş zamanlı dağıtım ve toplamalı araç rotalama problemi için ise Min' in 1989 yılında ve Dethloff' un 2001 yılında oluşturduğu kıyaslama problemi örnekleri ile test edilmiştir. Algoritma, problemlerin literatürde bilinen en iyi sonuçları ile karşılaştırldığında iyi sonuçlar vermektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The problem of transportation of people, goods or information is commonly denoted as routing problem. As the routing problem has wide areas of application, optimization of the transportation has become an important issue.

The basic routing problem is the Traveling Salesman Problem (TSP). The TSP is the problem of finding a minimal length closed tour that visits all cities of a given set exactly once. The Vehicle Routing Problem (VRP) is the TSP with *m* vehicles where a demand is associated with each city and the system has various constraints. VRP was first formulated by Dantzig and Ramser in 1959. The problem can be defined as the design of a set of minimum-cost vehicle routes, originating and terminating at a central depot, for a fleet of vehicles that services a set of customers with known demand (Dantzig and Ramser, 1959).

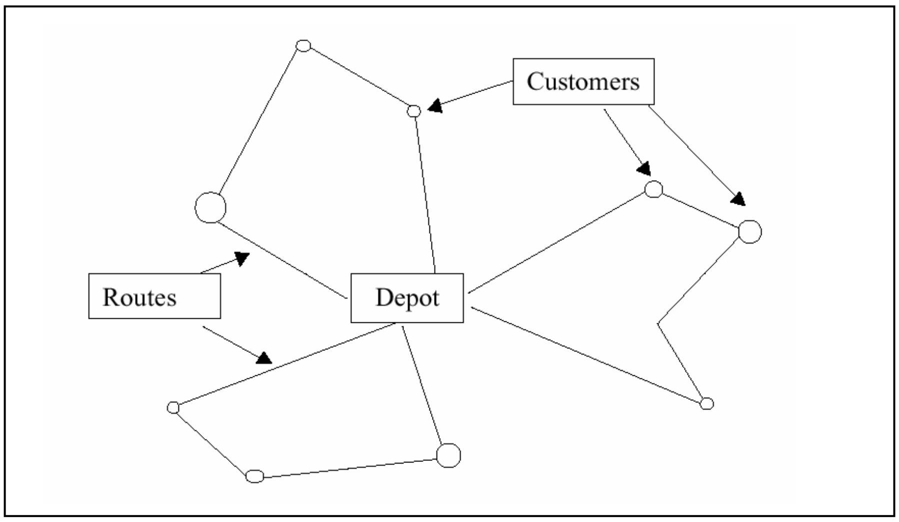

Figure 1.1 General representation of the Vehicle Routing Problem

In the literature, VRP is commonly formulated with capacity constraints, so the Vehicle Routing Problem generally has the same meaning with Capacitated Vehicle

Routing Problem (CVRP). Nevertheless, more realistic problems has various other constrains such as time windows, multiple depots/vehicles, etc.

There have been many papers proposing exact algorithms for solving the VRP. These algorithms are based on dynamic programming, Lagrangean relaxation, and column generation. On the other hand, as the VRP is known to be NP-hard, exact algorithms are not capable of solving problems for big numbers of customers.

Heuristics are thought to be more efficient for complex VRPs and have become very popular for researchers. There are three types of heuristics in the literature: construction algorithms, improvement algorithms, and metaheuristics. Since metaheuristic approaches are very efficient for escaping local optimum values while searching for better solutions they give competitive results. That is why the recent publications are all based on metaheuristic approaches such as genetic algorithms, tabu search, simulated annealing, ant systems.

In this thesis, an ant system (AS) based heuristic for the VRPs is proposed. AS was first introduced for solving the TSP. Since then many implementations of AS have been proposed for a variety of combinatorial optimization problems such as quadratic assignment problem (QAP), job shop scheduling problem, and VRP.

AS is based on the way that real colonies of ants behave in order to find shortest path between their nest and food sources. It simulates the behavior of real ants to solve combinatorial optimization problems with artificial ants. Artificial ants find solutions in parallel processes using a constructive mechanism guided by artificial pheromone and a greedy heuristic known as visibility. The amount of pheromone deposited on arcs is proportional to the quality of the solution generated and increases at run-time during the computation. In addition, the artificial ants are enabled to use local search heuristic in an attempt to improve the solution quality.

In this study, we propose an AS approach to Vehicle Routing Problem with Time Window (VRPTW) and Vehicle Routing Problem with Simultaneous Delivery and Pick-up (VRPSDP) that produces comparable results to those that exist in the literature. Chapter 2 includes a comprehensive literature review on the ant algorithms where a detailed definition of the algorithm is given, and major studies on this subject are explained.

Chapter 3 includes a detailed definition of the VRPTW and a literature review on the problem. It also describes the proposed ant system based approach for solving the VRPTW and reports the computational study on it. A benchmark study between the proposed approach and the best known results in the literature based on the test problems of Solomon (1987).

Chapter 3 includes a detailed definition of the VRPSDP and a literature review on the problem. It also describes the proposed ant system based approach for solving the VRPSDP and reports the computational study on it. The proposed approach has been tested on the benchmark problem instances of Min (1989) and Dethloff (2001).

The last chapter provides a discussion of the results achieved and concluding remarks. It also gives directions for future research.

2. ANT ALGORITHMS

Ant algorithms are one of the examples of swarm intelligence in which scientists study the behavior patterns of bees, termites, ants, and other social insects in order to simulate processes. Ant algorithms were first proposed by Dorigo *et al.* (1991) as an approach to solve combinatorial optimization problems like the TSP and QAP. Then, they have been applied to various other problems.

In this chapter, first general characteristics of ant algorithms and the ant colony optimization heuristic will be described. Then, applications of ant algorithms to various combinatorial optimization problems will be explained. Finally, a review of the improvements to ant algorithms will be given.

2.1. Basic Idea of Ant Algorithms

Understanding how blind animals like ants could establish shortest paths from their nests to feeding sources was one of the problems studied by ethnologists. Then, it was discovered that pheromone trails are used to communicate among individuals regarding paths and to decide where to go.

Ant algorithms are based on the way that real ant colonies behave in order to find shortest path between their nests and food sources. While walking ants leave aromatic essence, called pheromone, on the path they walk. Other ants can sense the existence of pheromone and choose their way according to the level of pheromone. Greater level of pheromone on a path will increase the probability that ants will follow that path. The level of pheromone laid is based on the length of the path and the quality of the food source. The level of pheromone on a path will increase when the number of ants following that path increases. In time all ants will follow the shortest path.

Choosing the shortest path can be explained in terms of *autocatalytic* behavior (i.e. positive feedback) that the more are the ants following a trail the more that trail becomes attractive for being followed. The most interesting aspect of autocatalytic process is that finding the shortest path around the obstacle is the result of the interaction between the obstacle shape and ants distributed behavior. Although all ants move at approximately the same speed and deposit a pheromone trail at approximately the same rate, it takes longer to go on their longer side than on their shorter side of obstacles. This makes the pheromone trail accumulate quicker on the shorter side.

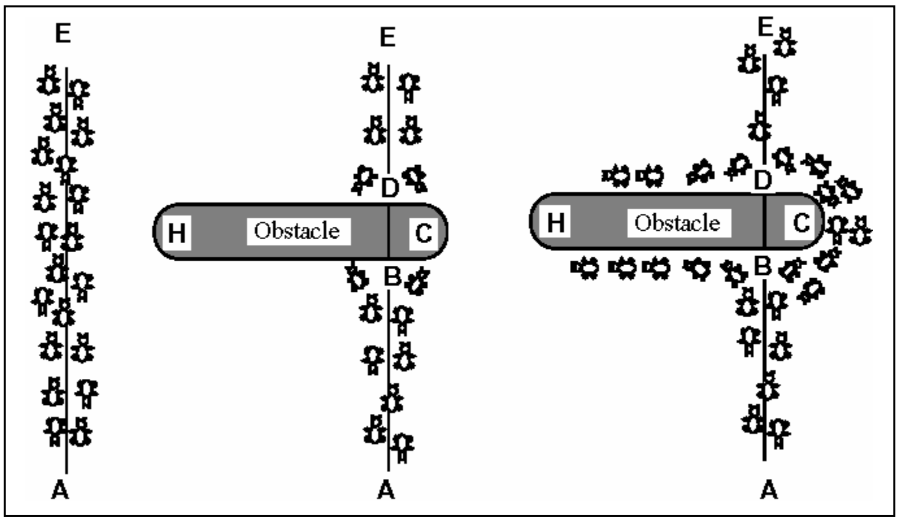

Figure 2.1 An example of the behavior of the real ants

Consider for example the experimental setting shown in Figure 2.1. There is a path along which ants are walking (for example from food source A to the nest E and vice versa). Suddenly an obstacle appears and the path is cut off. So at position B the ants walking from A to E (or at position D those walking in the opposite direction) have to decide whether to turn right or left. The choice is influenced by the intensity of the pheromone trails left by preceding ants. A higher level of pheromone on the right path gives an ant a stronger stimulus and thus a higher probability to turn right. The first ant reaching point B (or D) has the same probability to turn right or left (as there was no previous pheromone on the two alternative paths). Because path BCD is shorter than BHD the first ant following it will reach D before the first ant following path BHD. Shorter paths will receive pheromone reinforcement more quickly as they will be completed earlier than longer ones. The result is that an ant returning from E to D will find a stronger trail on path DCB, caused by the half of all the ants that by chance decided to approach the obstacle via DCBA and by the already arrived ones coming via BCD: they will therefore prefer path DCB to path DHB. As a consequence, the number of ants following path BCD per unit of time will be higher than the number of ants following BHD. This causes the quantity of pheromone on the shorter path to grow faster than on the longer one. Thus, the probability that any single ant chooses the path to follow is quickly biased towards the shorter one. The final result is that very quickly all ants will choose the shorter path (Dorigo and Colorni, 1996).

In what follows is the description of how ant system simulates the behavior of real ants to solve combinatorial optimization problems with artificial ants.

Consider the example in Figure 2.2, which is a possible AS interpretation of Figure 2.1 (Dorigo *et al*, 1991). The distances between D and H, between B and H, and between B and D are equal to 1. C is positioned in the middle of D and B. 30 new ants come to B from A and 30 to D from E at each time unit. Each ant walks at a speed of 1 per time unit and lays down a pheromone trail of intensity 1 at time *t*. Evaporation occurs in the middle of the successive time interval (*t*+1, *t*+2).

At *t*=0 30 ants are in B and 30 in D. As there is no pheromone trail they randomly choose the way to go. Thus, approximately 15 ants from each node will go toward H and 15 toward C.

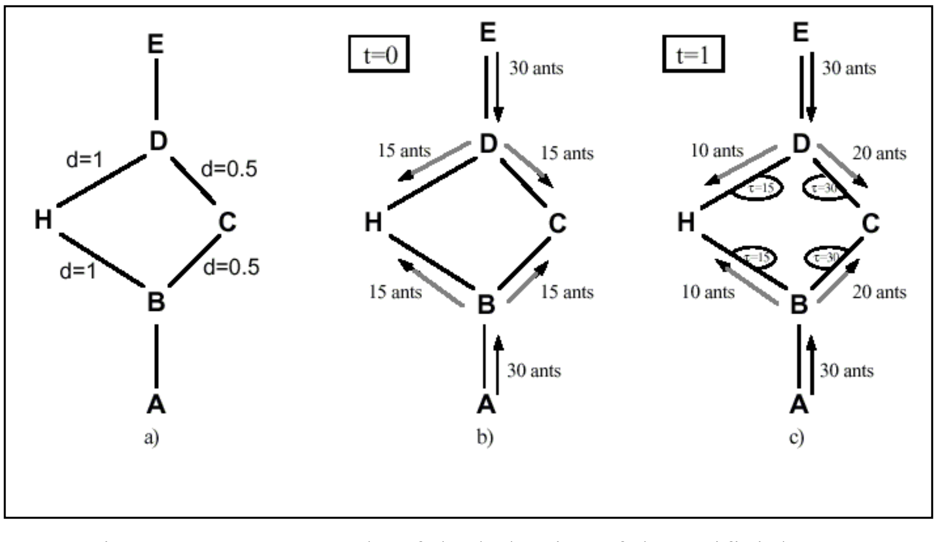

Figure 2.2 An example of the behavior of the artificial ants

At *t*=1 30 new ants come to B from A. They sense a trail of intensity 15 on the path that leads to H, laid by the 15 ants that went through B-H-D. They also sense a trail of intensity 30 on the path to C, obtained as the sum of the trail laid by the 15 ants that went through B-C-D and by the 15 ants that went through D-C-B. The probability of choosing a path is therefore biased. The expected number of ants going toward C will be the double of those going toward H: 20 versus 10, respectively. The same is true for the new 30 ants in D which came from E. This process continues until all of the ants eventually choose the shortest path.

In brief, if an ant has to make a decision about which path to follow it will most probably follow the path chosen heavily by preceding ants, and the more is the number of ants following a trail, the more attractive that trail becomes for being followed.

In the ant meta-heuristic, a colony of artificial ants cooperates in finding good solutions to discrete optimization problems. Artificial ants have two characteristics. On the one hand they imitate the following behavior of real ants:

- *Colony of cooperating individuals:* Like real ant colonies, ant algorithms are composed of entities cooperating to find a good solution. Although each artificial ant can find a feasible solution, high quality solutions are the result of the cooperation. Ants cooperate by means of the information they concurrently read/write on the problem states they visit.
- *Pheromone trail:* While real ants lie pheromone on the path they visit, artificial ants change some numeric information of the problem states. This information takes into account the ant's current performance and can be obtained by any ant accessing the state. In ant algorithms pheromone trails are the only communication channels among the ants. It affects the way that the problem environment is perceived by the ants as a function of the past history. Also an evaporation mechanism, similar to real pheromone evaporation, modifies the pheromone. Pheromone evaporation allows the ant colony to slowly forget its past history so that it can direct its search towards new directions without being over-constrained by past decisions.

- *Shortest path searching and local moves:* The aim of both artificial and real ants is to find a shortest path joining an origin to destination sites. Like real ants artificial ants move step-by-step through adjacent states of the problem.
- *Stochastic state transition policy:* Artificial ants construct solutions applying a probabilistic decision to move through adjacent states. As for real ants, the artificial ants only use local information in terms of space and time. The information is a function of both the specifications and pheromone trails induced by past ants.

On the other hand, they are enriched with the following capabilities.

- Artificial ants can determine how desirable states are.
- Artificial ants have a memory that keeps the ants' past actions.
- Artificial ants deposit an amount of pheromone which is a function of the quality of the solution found.
- The way that artificial ants lies pheromone is dependent on the problem.
- Ant algorithms can also be enriched with extra capabilities such as local optimization, backtracking, and so on, that cannot be found in real ants.

2.2. The Ant Colony Optimization Heuristic

In Ant Colony Optimization (ACO), a number of artificial ants with the described characteristics search for good quality solutions to the discrete optimization problem. If $G = (C, L)$ is assumed as the graph of a discrete optimization problem, ACO can be used to find to find a solution to the shortest path problem defined on the graph G. A solution is described in terms of paths through the states of the problem in accordance with the problems' constraints. For example, in the TSP, C is the set of cities, L is the set of arcs connecting cities, and a solution is a closed tour.

Each ant is assigned to an initial state based on problem criteria. The start state is usually defined as a unit length sequence. Artificial ants find solutions in parallel processes using an incremental constructive mechanism to search for a feasible solution.

It starts from the initial state and move to feasible neighbor states. Moves are made by applying a stochastic search policy guided by ants' memory, problem constraints, pheromone trail accumulated by all the ants from the beginning of the search process and problem-specific heuristic information (visibility). The ants' memory keeps information about the path it followed. It can be used to carry useful information to compute the goodness of the generated solution and/or the contribution of each executed move. It also provides the feasibility of the solutions. While building its own solution, each ant also collects information on the problem characteristics and its performance. It uses this information to modify the representation of the problem, as seen by the other ants. The information collected by ants is stored in pheromone trails. Visibility measures the attractiveness of the next node to be selected. Visibility value represents a priori information about the problem instance definition. A solution is constructed by moving through a sequence of neighbor states.

The decisions about when the ants should release pheromone on the environment and how much pheromone should be deposited depend on the problem. Ants can release pheromone while building the solution, or after a solution has been built, or both. In addition, pheromone trails can be associated with all problem arcs or some of them.

Probabilistic tables that are function of the pheromone trail and heuristic values guide the ants' search. The stochastic component of the decision policy and the pheromone evaporation mechanism prevents a rapid drift of all the ants towards the same part of the search space.

After building a solution the ant deposits additional pheromone information on the arcs of the solution. In general, the amount of pheromone deposited is proportional to the goodness of the solution. If a move generates a high-quality solution its pheromone will be increased proportionally to its contribution. After an ant constructs a solution and deposits pheromone information it dies.

Although a single ant can find a solution high quality solutions are only found as a result of the global cooperation among all ants. Communication among ants is mediated by information stored in pheromone trail values.

```
procedure ACO heuristics()
        While (termination condition not met)
            schedule activities
                ants generation and activity();
                pheromone evaporation();
                daemon actions();
            end schedule activities
        end while
end procedure

procedure ants generation and activity()
        While (available resources)
            new active ant();
        end while
end procedure

procedure new active ant();
        initialize ant();
        M=update ant memory ();
        While (current memory ≠complete solution)
            A=read local ant routing table();
            P=compute transition probabilities;
            next state=apply decision policy;
            move to next state(next state);
            if (local pheromone update)
                deposit pheromone on the visited arc();
                update ant routing table();
            end if
            M=update internal state();
        end while
        if (global pheromone update)
            foreach visited arc do
                deposit pheromone on the visited arc();
                update ant routing table();
            end foreach
        end if
        die();
end procedure
```

Figure 2.3 The ACO heuristic

In brief, a colony of ants concurrently moves through feasible adjacent states of the problem by applying a stochastic decision process. By moving, ants incrementally build solutions to the optimization problem. During the solution construction process or/and after the solution is constructed, the ants evaluate the (partial) solution and update pheromone trail values. Figure 2.3 provides the pseudo code of the ACO heuristic developed by Dorigo and Caro (1999).

Beside ants' generation and activity described above, ACO algorithm has two more procedures: *pheromone trail evaporation* and *daemon actions*. Pheromone evaporation is the process by which the pheromone trail values on the arcs decrease overtime. This prevents the convergence of the algorithm to a sub-optimal solution and enables the generation of new solutions. Daemon action is an optional process by which the solutions are observed and the extra pheromone is deposited on the arcs used by the shortest path.

Ants generation and activity, pheromone trail evaporation, and daemon actions of ACO need synchronization. In general, a strictly sequential scheduling of the activities is particularly suitable for non-distributed problems, where the global knowledge is easily accessible at any instant and the operations can be conveniently synchronized. On the contrary, some form of parallelism can be easily and efficiently exploited in distributed problems like routing in telecommunications networks (Dorigo *et al.*, 1998).

2.3. Ant System

In this section, general characteristics of the ant algorithms are described through Ant System (AS) approach, as it is the first study on ACO and most of the ant algorithms proposed are strongly inspired by AS. In addition, the first application of an ACO algorithm was done using the TSP, and TSP is the prototypical representative of NP-hard combinatorial optimization problems (Garey and Johnson, 1979). Therefore, AS is introduced with its application to the TSP. Then, its applications to solve other optimization problems will be explained.

2.3.1. TSP Application

The TSP is the problem of finding a minimal length closed tour that visits all cities of a given set exactly once. Artificial ants find solutions to the TSP in parallel processes using a constructive mechanism.

While solving the TSP, first all *m* artificial ants are randomly placed on cities and initial pheromone trail intensities are set on edges. Then, each artificial ant moves from one city to another. It chooses the city to move using a probabilistic function based on intensity of pheromone trail on edges and a heuristic function. Intensity of pheromone trail gives information about how many ants in the past have chosen that edge. The heuristic function is called visibility and is used to increase the probability of going to a closer city. In the earliest approaches, it was usually chosen as a function of the edges length. Artificial ants probabilistically choose closer cities with a lot of pheromone trail. Each time an ant makes a move the trail it leaves on path (i, j) is collected and used to compute the new values for path trails.

Each artificial ant has a memory called tabu list. The tabu list forces the ant to make legal tours. It saves the cities already visited and forbids the ant to move already visited cities until a tour is completed.

After all cities are visited, the tabu list of each ant will be full. The shortest path found is computed and saved. Then, tabu lists are emptied. This process is iterated for a user-defined number of cycles.

Suppose there are n cities and b_i is the number of ants at city i. Consider the following notation:

$m = \sum_{i=1}^{n} b_i$	: Total number of ants
N	: Set of cities to be visited
$tabu_k$	: Tabu list of the k-th ant
$tabu_k(s)$	: s-th city visited by the k-th ant in the tour
$\tau_{ij}(t)$	: Intensity of trail on edge between city i and city j at time t
η_{ij}	: Visibility of edge between city i and city j

η_{ij} is usually assumed as the inverse of the distance between city i and city j (d_{ij}) Thus, $\eta_{ij} = 1/ d_{ij}$.

After m artificial ants are randomly placed on cities, the first element of each ant's tabu list is set to be equal to its starting city. Then, they move to unvisited cities. The probability of moving from city i to city j for the k-th ant (p_{ij}^{k}) is defined as:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum\limits_{k \in allowed_k} [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta}} & , \quad j \in allowed_k \\ 0 & , \quad \text{otherwise} \end{cases} \tag{2.1}$$

where *allowed*$_k$ = {*N* – *tabu*$_k$}, α and β are parameters that control the relative importance of pheromone trail versus visibility.

Each time an ant moves from city *i* to city *j*, the pheromone trail on the edge (*i, j*) is modified. This is called as *local trail updatin*g. This prevents an edge to become dominant, and to be chosen by all the ants. Local trail updating is applied using the following formula:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot\tau_{0} \tag{2.2}$$

where τ_0 is a parameter representing the initial pheromone value on each edge and ρ is a coefficient such that (1 - ρ) represents the evaporation of trail.

After all the ants have completed their tours, the ant that made the shortest tour modifies the edges belonging to its tour. This is called as *global trail updatin*g and is applied using the following formula:

$$\tau_{ij} = \rho\cdot\tau_{ij} + \tau\Delta_{ij} \tag{2.3}$$

$$\tau\Delta_{ij} = \sum_{k=1}^{m} \tau\Delta_{ij}^{k}$$

where $\tau\Delta_{ij}^{k}$ is the quantity per unit of length of pheromone trail laid on path (*i*, *j*) by the *k*-th ant and is given by:

$$\tau\Delta_{ij}^{k} = \begin{cases} \dfrac{Q}{L_k} & \text{, if } k-\text{th ant uses path } (i,j) \text{ in its tour} \\ 0 & \text{,otherwise} \end{cases}$$

where *Q* is a constant and L_k is the tour length of the *k*-th ant.

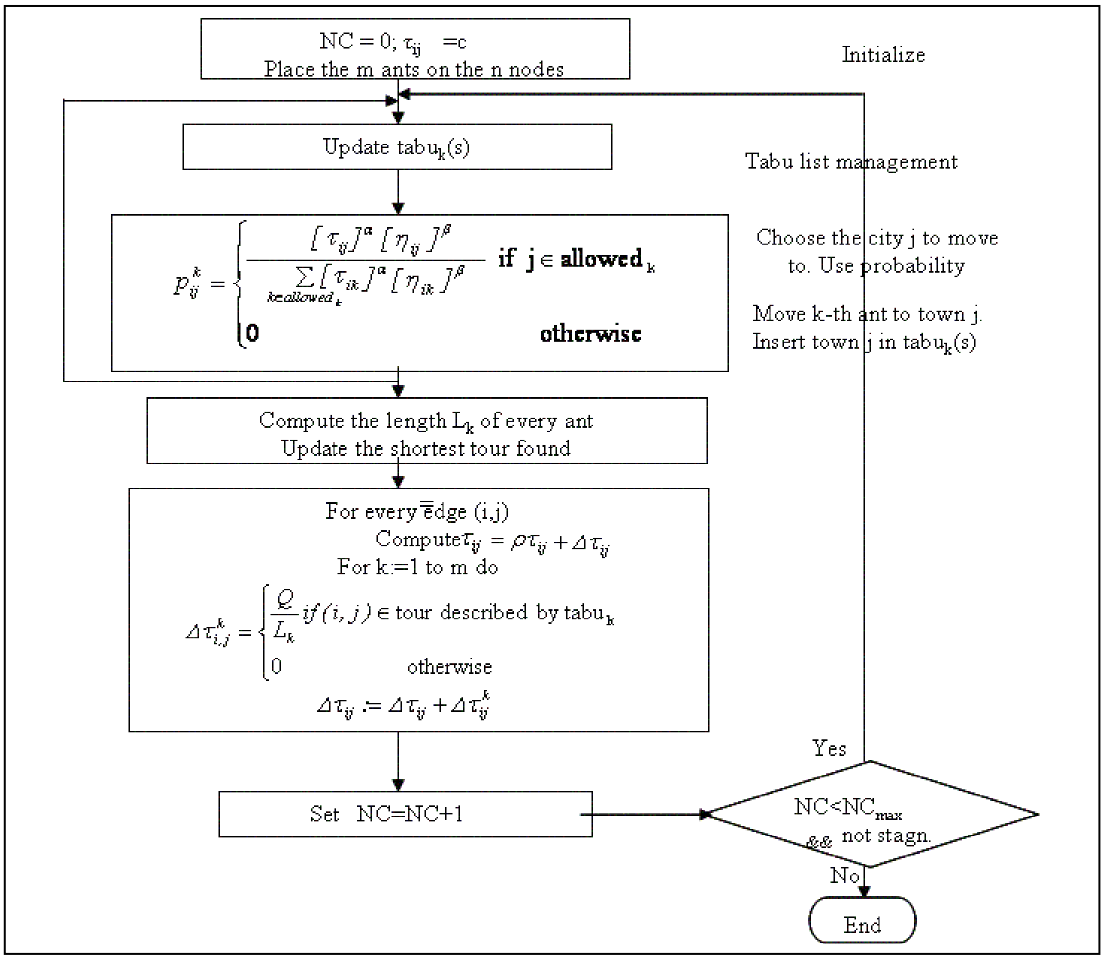
NC = 0; τ_{ij} =c
Place the m ants on the n nodes
Initialize
Update tabu$_k$(s)
Tabu list management
$p_{ij}^k = \begin{cases} \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{k \in allowed_k} [\tau_{ik}]^\alpha [\eta_{ik}]^\beta} & \text{if } j \in \text{allowed}_k \\ 0 & \text{otherwise} \end{cases}$
Choose the city j to move to. Use probability
Move k-th ant to town j. Insert town j in tabu$_k$(s)
Compute the length L_k of every ant
Update the shortest tour found
For every edge (i,j)
Compute $\tau_{ij} = \rho\tau_{ij} + \Delta\tau_{ij}$
For k:=1 to m do
$\Delta\tau_{i,j}^k = \begin{cases} \frac{Q}{L_k} & if (i,j) \in \text{tour described by tabu}_k \\ 0 & \text{otherwise} \end{cases}$
$\Delta\tau_{ij} := \Delta\tau_{ij} + \Delta\tau_{ij}^k$
Set NC=NC+1
Yes
NC<NC$_{max}$ && not stagn.
No
End

Figure 2.4 Solving the TSP using ACO

AS algorithm defined above is called *ant-cycle*. Two other algorithms of the AS *ant-density* and *ant-quantity* algorithms are also proposed. They differ in the way the trail is updated. In the ant-density, a quantity *Q* of trail is left on path (*i*, *j*). In the ant-quantity a quantity Q/d_{ij} of trail is left on edge (*i*, *j*) every time an ant goes from *i* to *j*.

In the ant-density:

$$\tau\Delta_{ij}^k = \begin{cases} Q & \text{, if } k-\text{th ant uses path } (i,j) \text{ in its tour} \\ 0 & \text{, otherwise} \end{cases}$$

In the ant-quantity:

$$\tau\Delta_{ij}^{k} = \begin{cases} \dfrac{Q}{d_{ij}} & \text{, if } k-\text{th ant uses path } (i,j) \text{ in its tour} \\ 0 & \text{, otherwise} \end{cases}$$

Finally, the shortest route is saved, the tabu lists of all ants are emptied, and the ants are free again to construct new tours. The process as described in Figure 2.4 continues until the tour counter reaches the maximum number of cycles, NC_{max}, or stagnation (all ants construct the same tour).

In general, all the ACO algorithms for the TSP follow a specific algorithmic scheme, which is outlined in Figure 2.5 (Stützle and Dorigo, 1999). After the initialization of the pheromone trails and some parameters a main loop is repeated until a termination condition. In the main loop, first, the ants construct feasible tours, then the generated tours are improved by applying local search, and finally the pheromone trails are updated.

```
procedure ACO algorithm for TSPs
    Set parameters, initialize pheromone trails
    While (termination condition not met)
        ConstructSolutions
        ApplyLocalSearch      % optional
        UpdateTrails
    end
end ACO algorithm for TSPs
```

Figure 2.5 An Algorithmic skeleton for ACO algorithm applied to the TSP

2.3.2. Other Applications

As ant algorithm is versatile, it can be applied to different variants of a problem. For example, it can also be used to solve the Asymmetric TSP (ATSP). Solving ATSP is similar to solving TSP. The only differences are in the distance and trail matrices that are not symmetric.

AS is also a robust heuristic that can be applied to various other combinatorial optimization problems such as VRP, QAP, the job-shop scheduling problem (JS), sequential ordering problem (SOP), graph coloring, routing in communications networks, and so on. (Dorigo *et al.*, 1991).

Assigning n facilities to n locations so that the cost of the assignment is minimized is an example of QAP. Since QAP is the generalization of the TSP, AS was first applied to QAP after the TSP.

Let, $\mathbf{D} = \{d_{ij}\}$, where d_{ij} is the distance between location i and location j and
$\mathbf{F} = \{f_{hk}\}$, where f_{hk} is the flow between facility h and facility k
A permutation π is interpreted as an assignment of facility $h = \pi(i)$ to location i, for each $i=1,..,n$. The problem is to identify a permutation π of both row and column indexes of the matrix **F** that minimizes the total cost:

$$\min Z = \sum_{i,j=1}^{n} d_{ij} . f_{\pi(i)\pi(j)}$$

An AS approach similar to AS approach of the TSP is used to solve the QAP. As AS requires the objective function represented on the basis of a single matrix, first the QAP objective function was expressed by a combination of the "potential vectors" of distance and flow matrices. The potential vectors, D and F, are the row sums of each of the two matrices as follows:

$$\mathbf{D} = \begin{bmatrix} 0 & 1 & 3 \\ 1 & 0 & 5 \\ 2 & 4 & 0 \end{bmatrix} \rightarrow D = \begin{bmatrix} 4 \\ 6 \\ 6 \end{bmatrix} \qquad \mathbf{F} = \begin{bmatrix} 0 & 10 & 20 \\ 10 & 0 & 20 \\ 20 & 10 & 0 \end{bmatrix} \rightarrow F = \begin{bmatrix} 30 \\ 30 \\ 30 \end{bmatrix}$$

From these two potential vectors, a third matrix **S** is obtained, where each element is computed as $s_{ih} = d_i . f_h$.

$$\mathbf{S} = \begin{bmatrix} 120 & 120 & 120 \\ 180 & 180 & 180 \\ 180 & 180 & 180 \end{bmatrix}$$

Visibility is used as the inverse of the values of **S**.

Various ACO algorithms for the QAP have been introduced. The interested reader is referred to Stützle and Dorigo (1999) for an overview of these approaches.

The JSP can be described as the following: A set of M machines and a set of J jobs are given. The j-th job (j=1, ..., J) consists of an ordered sequence of operations from a set $O=\{\ldots\ o_{jm}\ \ldots\}$. Each operation $o_{jm} \in O$ belongs to job j and has to be processed on machine m for d_{jm} consecutive time instants. $N=|O|$ is the total number of operations. The problem is to assign the operations to time intervals in such a way that no two jobs are processed at the same time on the same machine and the maximum of the completion times of all operations is minimized (Graham *et al.*, 1979).

To solve JSP by AS, first the problem is represented as a directed weighted graph $Q=(O',A)$ where $O'=O-\{o_0\}$ and A is the set of arcs that connect o_0 with the first operation of each job and that completely connect the nodes of O except for the nodes belonging to a same job. Nodes belonging to a same job are connected in sequence. Node o_0 specifies the job scheduled first. Therefore, there are N+1 nodes and ($N.(N-1)/2+|J|$) arcs. Each arc is weighted by intensity of trail (τ_{ij}) and the visibility η_{ij}. Visibility can be calculated as a function of the processing time or the completion time.

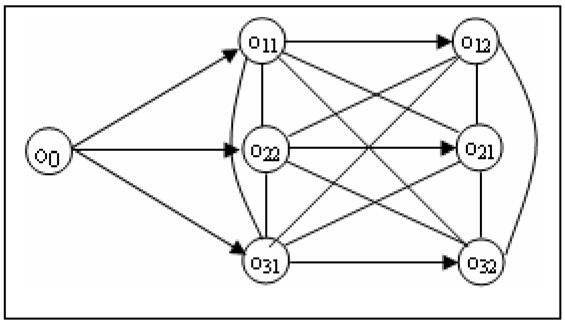

Figure 2.6 A graph for JSP with 3 jobs and 2 machines

First, all ants are placed on o_0. Then, at each step a feasible permutation of the remaining nodes have to be identified. In order to obtain a feasible permutation, the set of allowed nodes must be defined according to both the tabu list, and the problem characteristic. For each ant k, let G_k be the set of all the nodes still to be visited and S_k the set of the nodes allowed at the next step. Transition probabilities are computed on the basis of Equation 2.1, where the set of allowed nodes is equal to S_k. When a node is chosen, it is deleted from both G_k and S_k. If the chosen node is not the last job then its immediate successor is added to S_k. In this way, feasible solutions are provided. The process continues until G_k is emptied. The trails are computed as in the case of TSP. However, results are not competitive.

The SOP is closely related to the ATSP, but additional precedence constraints between the nodes have to be satisfied. Gambardella and Dorigo (1997) extended the AS approach used to solve the ATSP and enhanced it by a local search algorithm. Then, they obtained excellent results and were able to improve the best known solutions for many benchmark instances.

2.4. Improvements to Ant System

AS is the first study which uses ACO algorithm to solve *NP*-hard combinatorial optimization problems. However, its performance compared to other approaches is rather poor. Therefore, several ACO algorithms have been proposed in order to increase the performance of AS. Improved versions have been applied to various optimization problems. Examples include the VRP (Bullnheimer *et al.*, 1997; Gamberdella *et al.*, 1999), sequential ordering (Gamberdella and Dorigo, 1997), single machine tardiness (Bauer *et al.*, 1999), multiple knapsack (Leguizamon and Michalewicz, 1999), etc.

2.4.1. Elitist Strategy

A first improvement on the AS is called the elitist strategy, and is introduced in Dorigo *et al.*(1996). The global best tour is denoted by L_{gb} and a strong additional reinforcement to the arcs belonging to that tour is given. When the pheromone trails are updated, pheromone value equal to $e.1/L_{gb}$ is added to the arcs of that tour, where e is the number of elitist ants.

2.4.2. Ant Colony System

Dorigo and Gambardella (1996) proposed the ACS which has two types. In the first type, after all the ants have built a solution, pheromone trails on the arcs used by the ant that found the best tour so far are updated. In the second, after all the ants have built a solution, a local search procedure based on 3-opt is applied to improve the solutions and then pheromone trails on the arcs used by the ant that found the best tour so far are updated. The pheromone trail update rule is as follows:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho.\tau\Delta_{ij} \qquad (2.4)$$

where $\tau\Delta_{ij}$ = (length of the shortest tour)$^{-1}$

A different decision rule, called pseudo-random-proportional rule, is used in the ACS. The pseudo-random-proportional rule P_{ij}^{k}, used by ant k in node *I* to choose the next node *j* is the following:

$$P_{ij}^{k} = \begin{cases} \arg\max\limits_{j\in allowed_i} \left\{\tau_{ij}\cdot\left[\eta_{ij}\right]^{\beta}\right\} & , \quad \text{if } q \le q_0 \\ p_{ij}^{k} & , \quad \text{otherwise} \end{cases} \qquad (2.5)$$

where q is a random variable uniformly distributed over [0, 1], and q_0 Є [0, 1] is a parameter.

While using the probabilistic choice of the components to construct a solution is called exploration, choosing the component that maximizes a blend of pheromone trail values and heuristic evaluations is called exploitation.

$$p_{ij}^{k} = \begin{cases} \dfrac{\left[\tau_{ij}\right]\cdot\left[\eta_{ij}\right]^{\beta}}{\sum\limits_{k\in allowed_k}\left[\tau_{ik}\right]\cdot\left[\eta_{ik}\right]^{\beta}} & , j \in allowed_k \\ 0 & , \text{otherwise} \end{cases} \qquad (2.6)$$

An ant moving from city *i* to city *j* updates the pheromone trail on arc (*i, j*).

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi.\tau_0 \qquad (2.7)$$

where $0 < \varphi \leq 1$

$$\tau_0 = (n.L_{nn})^{-1}$$

where n is the number of cities and L_{nn} is the length of a tour produced by the nearest neighbor heuristic.

Last, ACS exploits a data structure called candidate list which provides additional local heuristic information. A candidate list is a list of preferred cities to be visited from a given city. In ACS, when an ant is in city, instead of examining all the unvisited neighbors, it chooses the city to move to among those in the candidate list. Other cities are examined only if no candidate list city has unvisited status. The candidate list of a city contains d cities ordered by non-decreasing distance (d is a parameter) and the list is scanned sequentially and according to the ant tabu list to avoid already visited cities (Dorigo *et al.*, 1998).

There are other versions of ACS. These differ from the ACS described above:

(i) in the way local pheromone update applied, such as setting $\tau_0 = 0$,

(ii) in the way the decision rule are made

(iii) in the type of solution used for global pheromone updating, such as adding the pheromone only to arcs belonging to the best solution found

2.4.3. Ant –Q

Ant-Q has the same characteristics as ACS. The only difference is in the value of τ_0. Pheromone trails are updated with a value which is a prediction of the value of the next state. In Ant-Q, an ant k applies global pheromone updates by the following equation:

$$\tau_{ij} = (1-\varphi)\cdot\tau_{ij} + \varphi.\gamma.\max_{l \in allowed_j} \tau_{jl} \tag{2.8}$$

Unfortunately, it was later found that setting the complicate prediction term to a small constant value, as it is done in ACS, resulted in approximately the same performance. Therefore, although having a good performance, Ant-Q was abandoned for the equally good but simpler ACS (Dorigo *et al.*, 1998).

2.4.4. MAX-MIN Ant System

Stutzle and Hoos (1997) proposed the MAX-MIN Ant System (MMAS). The solutions are constructed in the same way as in AS. The main modifications are the followings:

- The allowed range for intensity of pheromone trails are in an interval [τ_{min} ,τ_{max}]. This indirectly limits the probability p_{ij} of selecting a city j when an ant is in city i to an interval [p_{min}; p_{max}], with $0 < p_{min} \leq p_{ij} \leq p_{max} \leq 1$.
- Initial pheromone values are set equal to τ_{max}. This increases the exploration of tours at the start of the algorithm, since the relative differences between the pheromone trail values are less pronounced.
- After each iteration, only the pheromone levels of the arcs used by the best ant are increased using the formula (2.3).
- To avoid stagnation that may occur in case some pheromone trails are close to $\tau_{\max}$ while most others are close to τ_{min}, pheromone trails are updated such that:

$$\Delta\tau_{ij} = \tau_{\max} - \tau_{ij}$$

Better solutions are obtained using MMAS.

In Stutzle and Hoos (1999), MMAS using the pseudo-random-proportional action choice rule of ACS is considered. Using that choice rule, very good solutions could be found faster but the final solution quality achieved was worse.

MMAS applied to the flow shop scheduling problem (FSP) outperforms earlier proposed Simulated Annealing algorithms and performs comparably to Tabu Search algorithms (Stützle,1997)

MMAS has been applied to the generalized assignment problem by Lorençou and Serra (1998). It found optimal and near optimal solutions.

2.4.5. AS$_{rank}$

Bullnheimer *et al.*(1997) proposed AS$_{rank}$ where after all *m* ants construct their tours, the ants are sorted by their tour lengths ($L_1 \leq L_2 \leq \ldots \leq L_m$). The trail levels on the arcs visited by the best σ -1 ants are updated. Contribution of an ant to the trail level update is proportional to the rank μ of the ant. In addition, extra emphasis is given to the best route found so far. When the trail levels are updated this path is treated as if a certain number of ants, namely the σ elitist ants, had chosen the path. The amount of pheromone on the arc (*i*, *j*) is updated according to the following formula:

$$\tau_{ij} = \rho.\tau_{ij} + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \tag{2.9}$$

$$\Delta\tau_{ij} = \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu}$$

$$\Delta\tau_{ij}^{\mu} = \begin{cases} (\sigma - \mu)\dfrac{1}{L_{\mu}}, & \text{if the } \mu\text{ - th best ant move on arc}(i, j) \\ 0, & \text{otherwise} \end{cases}$$

$$\Delta\tau_{ij}^{*} = \begin{cases} \sigma\dfrac{1}{L^{*}}, & \text{if arc}(i, j) \text{ is on the so far found best solution} \\ 0, & \text{otherwise} \end{cases}$$

μ	: Ranking index
$\Delta\tau_{ij}^{\mu}$	: Increase of trail level on arc (i, j) by the μ - th best ant
L_{μ}	: Tour length of the μ - th best ant
$\Delta\tau_{ij}^{*}$	: Increase of trail level on arc (i, j) by the elitist ants
σ	: Number of elitist ants
L^{*}	: Tour length of the best solution found so far

2.4.6. Local Search

Local search starts from some initial assignment and repeatedly tries to improve the current assignment by local changes. If a better tour *T* is found, it replaces the current tour and the local search is continues from *T*. The most widely known improvement algorithms are 2-opt (Croes, 1958) and 3-opt (Lin, 1965). They test whether the current solution can be improved by replacing 2 or 3 arcs, respectively.

Local search algorithms with *k* >3 arcs to be exchanged are not used commonly due to the high computation times.

2.4.7. Candidate List

A candidate list contains a given number of potential customers to be visited for each customer *i*. Many AS procedures use a candidate list in order to reduce run-times of larger instances. Generally, candidate set strategies have only been used as a part of local search procedure applied to the solutions constructed by ACO. However, in improvements of ACS, candidate set strategies were applied as part of the construction process. An ant first chooses the next customer to be visited from the candidate list corresponding to the current customer. After all the states in the candidate list have been visited, one of the remaining states is considered. Candidate lists are usually formed using nearest neighborhood when TSPs are solved. A candidate list consists of a fixed number of cities for each city in the order of non-decreasing distances.

Stützle and Hoos (1996) proposed a candidate set strategy that requires to be regenerated throughout the search process. Randall and Montgomery (2002) proposed two types of candidate set for ACO: elite candidate set and evolving set. In the elite candidate set, the candidate set is formed by selecting the best *k* states based on their probability values. Then, this set is used for the next *l* iterations. In the evolving set, states with low probability values are eliminated temporarily and these states are not used for the next *l* iterations.

3. VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

In this chapter, first the VRPTW will be explained, and a linear integer programming formulation of it will be given. Then, a detailed review of the VRPTW from the literature is given. Finally, an ACO based approach is proposed and applied to VRPTW.

3.1. Mathematical Formulation of the VRPTW

The simplest type of the VRP is the capacitated vehicle routing problem (CVRP). In the CVRP, each customer i (i = 1…n) has a demand q_i of goods and each vehicle with a capacity Q is available to deliver goods. A solution to CVRP is a set of tours where each customer visited exactly once, each vehicle must start and end its tour at the depot, and the total tour demand is at most Q.

Mathematically, CVRP is described by a set of homogenous vehicles V, a set of customers C, and a directed graph G (N, A, d). $N = \{0,\ldots,n+1\}$ denotes the set of vertices. The graph consists of $|C|+2$ vertices where the customers are denoted by 1, 2,...,n and the depot is represented by the vertices 0 and $n+1$. $A = \{(i, j): i\neq j\}$ denotes the set of arcs that represents connections between the depot and the customers and among the customers. No arc terminates at vertex 0 and no arc originates from vertex $n+1$. A cost(distance) c_{ij} is associated with each arc (i, j). Finally, Q, d_i, c_{ij} are assumed to be non-negative integers.

For each arc (i, j), where $i \neq j$; $i \neq n + 1$; $j \neq 0$, and each vehicle k, x_{ijk} is defined as

$$x_{ijk} = \begin{cases} 1 & , \quad \text{if vehicle } k \text{ uses arc } (i, j) \\ 0 & , \quad \text{otherwise} \end{cases}$$

The goal is to design a set of minimal total cost routes such that each customer is serviced exactly once and every route originates at vertex *0* and ends at vertex $n + 1$.

VRP can be stated mathematically as: (Larsen,1999)

$$\min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} . x_{ijk} \qquad (3.1)$$

s.t.

$$\sum_{k \in V} \sum_{i \in N} x_{ijk} = 1 \qquad \forall i \in C \qquad (3.2)$$

$$\sum_{i \in C} d_i \sum_{j \in N} x_{ijk} \leq Q \qquad \forall k \in V \qquad (3.3)$$

$$\sum_{j \in N} x_{0jk} = 1 \qquad \forall k \in V \qquad (3.4)$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0 \qquad \forall h \in C \ , \forall k \in V \qquad (3.5)$$

$$\sum_{i \in N} x_{in+1k} = 1 \qquad \forall k \in V \qquad (3.6)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in N \ , \ \forall k \in V \qquad (3.7)$$

In the model above, the objective function (3.1) aims to minimize the total travel distance. The constraint (3.2) assures visiting each customer exactly once and (3.3) states that no vehicle is loaded more than its capacity. The next three equations (3.4, 3.5, and 3.6) ensure that each vehicle leaves the depot *0*, after arriving at a customer the vehicle leaves that customer again, and finally arrives at the depot n+1. Constraints (3.7) are the binary constraints.

Most real world problems encountered in distribution have a time constraint within which distribution of goods or services can be made. In addition, customers' preferences, such as in restaurants where deliveries are only allowed before a certain time of the day, may also restrict the schedule of the vehicles involved. Normally, these issues are simplified and formulated as VRP; the solution to this relatively unconstrained problem may not be practical (Bodin, 1990). VRPTW generalizes VRP by involving additional constraints which restricts each customer to be served within a given time window.

VRPTW is a well-known NP-hard problem which is an extension of VRP, encountered very frequently in making decisions about the distribution of goods and services (Tan *et al.*, 2000). In VRPTW least cost routes from a given central depot to a set of geographically scattered customers with known demands are designed for a fleet of identical/non-identical vehicles with known capacities. The routes must originate and terminate at the depot. Moreover, each customer is visited only once by exactly one vehicle within a given time, and each route must satisfy capacity constraint.

Time window $[a_i, b_i]$ given for a customer is defined as follows: a_i and b_i are the earliest and the latest times, respectively, when the customer permits the start of the service. Service at customer i must not start before a_i and the vehicle must arrive at customer i before b_i. The vehicle may arrive before a_i but the customer cannot be serviced until a_i. The depot also has a time window $[a_0, b_0]$. A vehicle can leave the depot after a_0 and must return to the depot until b_0.

In VRPTW, allowable delivery times of the customers add complexity to the VRP because of the time feasibility constraint that must be satisfied for each customer. The following is set of decision variables and constraints added to the model to specify the times that services begin.

s_{ik} : Time that vehicle k starts to service customer i $\quad \forall i \in N, \forall k \in V$

Assuming $a_0 = 0$, $s_{0k} = 0 \quad \forall k \in V$

$$s_{ik} + t_{ij} - K(1 - x_{ijk}) \leq s_{jk} \quad \forall i, j \in N, \forall k \in V \tag{3.8}$$

$$a_i \leq s_{ik} \leq b_i \quad \forall i \in N, \forall k \in V \tag{3.9}$$

Constraints (3.8) state that vehicle k going from i to j can not arrive at j before $s_{ik} + t_{ij}$. K in this constraint is a very large number. Constraints (3.9) ensure the observations of time windows.

In some cases, vehicles are allowed to start service just at the time they arrive to the customer site. So, in these types of problems, there are no waiting times for the vehicles at the customer sites.

3.2. Complexity of VRPTW

The problem of finding the route for only one vehicle/person that has to visit a set of customers is called the TSP. TSP is a well-known NP-hard problem. The VRP is the generalization of the TSP, as the TSP is the VRP with one vehicle and without any constraints, such as customer demand or vehicle capacity. As an *m*-TSP, VRP is more complicated than TSP. Adding time windows constraints to the VRP results in a more complicated VRP than the VRP without time windows. Furthermore, Savelsbergh (1985) had shown that even finding a feasible solution to the VRPTW when the number of vehicles is fixed is itself an NP-Complete problem. Although optimal solutions to VRPTW can be obtained using exact methods, the computational time required to solve the VRPTW to optimality is prohibitive (Desrochers *et al.,*1992). Therefore, the development of approximation algorithms or heuristics for this problem has been of primary interest to many researchers.

3.3. Optimal Algorithms for VRPTW

The first exact algorithm for the VRPTW was proposed by Kolen *et al.* (1987). Since then various researchers have studied on exact algorithms for the VRPTW. Exact algorithms in the literature are based on principles of dynamic programming, lagrangean relaxation, and column generation.

3.3.1. Dynamic Programming

Kolen *et al.* (1987) is inspired from Christofides *et al.* (1981) and presented the first paper on dynamic programming for the VRPTW. In this paper, branch-and-bound approach was used in order to retrieve optimal solutions. There are three nodes in the branch-and-bound algorithm, each of which corresponds to three sets: The set of fixed feasible routes starting and finishing at the depot, partially built route starting at the depot, and customers that are not allowed to be next on partially built route starting at the depot. Branching is done by selecting a customer that is not forbidden and that does not appear in any route. At each branch-and-bound node, dynamic programming is used to calculate a lower bound on all feasible solutions.

3.3.2. Lagrangean Relaxation-Based Methods

There are many studies that use Lagrangean relaxation based methods for solving VRPTW. Variable splitting followed by Lagrangean decomposition was used by Jörnsten *et al.* (1986), Madsen *et al.* (1988) and Halse (1992). Jörnsten *et al.* (1986) presented variable splitting for the first time, but no computational results were given. Madsen *et al.* (1988) also presented four different decomposition approaches without any computational results. Then, Halse (1992) offered three approaches and gave the computational results of one of these approaches.

Fisher *et al.* (1997) used K-*tree* approach followed by Lagrangean relaxation. They formulate the VRPTW as finding a K-*tree* with degree $2K$ on the depot, degree 2 on the customers and subject to capacity and time constraints. This representation becomes equal to K routes.

Finally Kohl *et al.* (1997) relax the constraints that ensure each customer must be visited exactly once and add a penalty term to the objective function. The model is decomposed into one sub-problem for each vehicle. The resulting problem is a shortest path problem with time window and capacity constraints.

3.3.3. Column Generation

Column generation is used when a linear program contains too many variables to be solved explicitly. The linear program is initialized with a small subset of variables and all other variables are set to 0. Then, a solution to that reduced linear program is computed. Afterwards, it is checked if the addition of one or more variables, not in the linear program, might improve the LP-solution.

Desrochers *et al.* (1992) used the column generation approach for solving the VRPTW for the first time. They add feasible columns as needed by solving a shortest path problem with time windows and capacity constraints using dynamic programming. The LP solution obtained provides a lower bound that is used in a branch-and-bound algorithm to solve the integer set-partitioning formulation.

Kohl (1995) solves more instances using a more effective version of the same model as Desrochers *et al.* (1992) with the addition of valid inequalities.

3.4. Approximation Algorithms for the VRPTW

Since the VRPTW is an NP-hard problem, many approximation algorithms have been proposed in the literature. These algorithms can be classified into three groups: construction algorithms, improvement algorithms, and metaheuristics.

3.4.1. Construction Algorithms

Construction algorithms are used to build an initial feasible solution for the problem. They build a feasible solution by inserting unrouted customers iteratively into current partial routes according to some specific criteria, such as minimum additional distance or maximum savings, until the route's scarce resources (e.g. capacity) are depleted (Cordeau *et al.*, 1999). These types of algorithms are classified as either sequential or parallel algorithms. In a sequential algorithm routes are built one at a time whereas in a parallel algorithm many routes are constructed simultaneously.

3.4.1.1 Sequential Construction Algorithms

Sequential construction algorithms are mostly based on the *Sweep Heuristic* (Gillet and Miller, 1974) and the *Savings Heuristic* (Clarke and Wright, 1964). In the sweep heuristic, routes are constructed as an angle sweeps the location of nodes on a 2D space. In the savings heuristic, first routes are constructed in a predefined quantity and then new nodes are added to available nodes in order to obtain maximum savings.

Baker and Schaffer (1986) proposed the first sequential construction algorithm. The algorithm is based on savings heuristic, and starts with all possible single customer routes in the form of depot – *i* – depot. Then two routes with the maximum saving are combined at each iteration. The saving between customers *i* and *j* is calculated as:

$$s_{ij} = d_{i0} + d_{0j} - G.d_{ij} \tag{3.10}$$

where G is the route form factor and d_{ij} is the distance between nodes i and j.

Solomon (1987) proposed *Time Oriented Nearest Neighborhood Heuristic*. Every route is initialized with the customer closest to the depot. At each iteration unassigned customer that is closest to the last customer is added to the end of the route. When there is no feasible customer, a new route is initialized.

Solomon (1987) also proposed *Time-Oriented Sweep Heuristic*. First, customers are assigned to different clusters and then TSPTW problem is solved using the heuristics proposed by Savelsbergh (1985).

3.4.1.2 Parallel Construction Algorithms

Solomon (1987) proposed a *Giant-Tour Heuristic*. In this heuristic, first of all, a giant route is generated as a traveling salesman tour without considering capacity and time windows. Then, it is divided into number of routes.

Potvin and Rousseau (1993) proposed parallelization of the *Insertion Heuristics*. Each route is initialized by selecting the farthest customer from the depot as a center customer. Then, the best feasible insertion place for each not yet visited customer is computed. Customers with the largest difference between the best and the second best insertion place are inserted to the best feasible insertion place. Parallel algorithm in Foisy and Potvin (1993) also constructs routes simultaneously using the Insertion Heuristics to generate the initial center customers.

Antes and Derigs (1995) proposed another parallel algorithm based on the Solomon's heuristic. Offers comes to the customers from the routes, unrouted customers send a proposal to the route with the best offer, and each route accepts the best proposal.

3.4.2. Improvement Algorithms

Improvement algorithms try to find an improved solution starting from a considerably poorer solution. Almost all improvement algorithms for the VRPTW use an exchange neighborhood to obtain a better solution. Exchange of neighborhood can be intra or inter route (Thangian and Petrovic, 1998). While *k*-opt procedure operates within a route, the relocate, exchange, and cross operators operate between routes.

Croes (1958) introduced *k*-opt approach for single vehicle routes. In this heuristic, a set of links in the route are replaced by another set of *k* links.

The Or-Opt exchange originally proposed for TSP by Or (1976) removes a chain of at most three consecutive customers from the route and tries to insert this chain at all feasible locations in the routes.

In 1-1 exchange procedure connectors between nodes are replaced by connectors between nodes either in the same or in different route. 1-0 exchange move transfers a node from its current position to another position in either the same or a different route.

Christofides and Beasley (1984) proposed the *k*-node interchange for the first time to take time windows into account. In this heuristic, sets M_1 and M_2 are identified for each customer *i*. M_1 denotes the customer *i* and its successor *j*. M_2 denotes two customers that are closest to *i* and *j* on a different route than *i* and *j*. The elements of the sets M_1 and M_2 are removed and inserted in any other possible way.

Osman and Christofides (1994) introduced λ-interchange local search that is a generalization of the relocate procedure. λ, the parameter, denotes the maximum number of customer nodes that can be interchanged between routes.

Potvin and Rousseau (1995) present two variants of 2-Opt and Or-Opt. For the 2-Opt, they proposed the consideration of every pair of links in different routes for removal. For the Or-Opt, every sequence of three customers is considered and all insertion places are also considered for each sequence.

Schulze and Fahle (1999) proposed shift-sequence algorithm. A customer is moved from one route to another checking all possible insertion positions. If an insertion is feasible after the removal of another customer, that customer is removed.

3.4.3. Metaheuristics

In order to escape local optima and enlarge the search space, metaheuristic algorithms such as simulated annealing, tabu search, genetic algorithm, and ant colony algorithm have been used to solve the VRPTW (Bräysy and Gendreau, 2001).

3.4.3.1. Simulated Annealing

Simulated Annealing (SA) is a stochastic relaxation technique. It is based on the annealing process of solids, where a solid is heated to a high temperature and gradually cooled in order to crystallize (Bräysy and Gendreau, 2001). During the SA search process, the temperature is gradually lowered. At each step of the process, a new state of the system is reached. If the energy of the new state is lower than the current state, the new solution is accepted. But if the energy of the new state is higher, it is accepted with a certain probability. This probability is determined by the temperature. SA continues searching the set of all possible solutions until a stopping criterion is reached.

Thangiah *et al.* (1994) used λ-interchange with λ=2 to define the neighborhood and decrease the temperature after each iteration. In case the entire neighborhood has been explored without finding and accepting moves the temperature is increased.

Chiang and Russell (1996) proposed three different SA methods. First one uses modified version of the *k*-node interchange mechanism and second uses λ-interchange with λ=1. The third is based on the concept of tabu list of Tabu Search.

Tan *et al.* (2001) proposed an SA heuristic. They defined a new cooling schedule. Thus, ehen the temperature is high, the probability of accepting the worse is high, when the temperature is decreased according to function given above; the probability of accepting worse is reduced.

Finally, Li and Lim (2003) proposed an algorithm that finds an initial solution using Solomon's insertion heuristic and then starts local search from initial solution using proposed tabu-embedded simulated annealing approach.

3.4.3.2. Tabu Search

Tabu search (TS) presented by Glover (1986) is a memory based local search heuristic. In TS, the solution space is searched by moving from a solution *s* to the best solution in its neighborhood *N*(*s*) at each iteration. In order to avoid from a local optimum, the procedure does not terminate at the first local optimum and the solution may be deteriorated at the following iteration. The best solution in the neighborhood is selected as the new solution even if it is poorer. Solutions having the same attributes with the previously searched solutions are put into tabu list and moving to these solutions is forbidden. This usually prevents making a move to solutions obtained in the last *t* iterations. TS can be terminated after a constant number of iterations without any improvement of the over all best solution or a constant number of iteration.

Garcia *et al.* (1994) applied TS to solve VRPTW for the first time. They generate an initial solution using Solomon's insertion heuristic and search the neighborhood using 2-opt and Or-opt. Garcia et al. (1994) also parallelized the TS using partitioning strategy. One processor is used for controlling the TS while the other is used for searching the neighborhood.

Thangiah et al. (1994) proposed TS with λ-interchange improvement method. They also combined TS with SA to accept or reject a solution.

Potvin *et al.* (1995) proposed an approach similar to Garcia et al. (1994) based on the local search method of Potvin and Rousseau (1995).

Badeau *et al.* (1997) generated a series of initial solutions. Then, they decomposed them into groups of routes and performed TS for each group using the exchange operator. Their tabu list contains penalized exchanges that are frequently performed.

Chiang and Russell (1997) used a parallel version of Russell (1995) to generate the initial solution and then applied λ-interchange. They penalize frequently performed exchanges and dynamically adjust parameter values based on the current search.

De Backer and Furnon (1997) used the savings heuristic to generate the initial solution and search the neighborhood using 2-opt and Or-opt .

Schulze and Fahle (1999) propose a parallel TS heuristic where initial solutions are generated using the savings heuristic and the neighborhood is searched using route elimination and Or-opt. The search penalizes frequently performed exchanges. All routes generated are collected in a pool. To obtain a new initial solution for the TS heuristic, a set covering heuristic is applied to the routes in the pool.

Tan *et al.* (2000) generate the initial solution using modified Solomon's insertion heuristic and search the neighborhood using λ-interchange and 2-opt. A candidate list is used to save elite solutions found during the search process.

Lau *et al.* (2002) introduce a concept of holding list containing the not yet serviced customers. All customers are put into the holding list at the beginning. Relocate and exchange operators are then used to transfer customers back and forth to the holding list.

3.4.3.3. Genetic Algorithms

The Genetic Algorithm (GA) is based on the Darwinian concept of evolution. Solutions to a problem are encoded as chromosomes and based on their fitness; good properties of solutions are propagated to a next generation (Vacic and Sobh, 2002). The creation of the next generations involves four major phases:

1. Representation: The significant features of each individual in the population are encoded as a chromosome.
2. Selection: Two parent chromosomes are selected from the population.
3. Reproduction: Genetic information of selected parents is combined by crossover and two offspring of the next generation are generated.
4. Mutation: The gene sequence of small number of newly obtained is randomly swapped.

A new generation is created by repeating the selection, reproduction, and mutation phases until a specified set of new chromosomes have been created. Then the current population is set to the new population of chromosomes.

Thangiah *et al.* (1991) applied the GA to VRPTW for the first time. GA is proposed to find good clusters of customer. The routes within each cluster are then constructed with a cheapest insertion heuristic and λ–interchange are applied.

Thangiah *et al.* (1995) generate initial population by clustering the customers randomly into groups and applying the cheapest insertion heuristic for each group. Then, 2-point crossover is used.

GA of Potvin and Bengio (1996) is performed on chromosomes of feasible solutions. Parents are randomly selected and two types of crossover are applied to these parents. The reduction of routes is obtained by two mutation operators. The routes are improved using Or-Opt at every *k* iterations.

Homberger and Gehring (1999) generated initial population using a modified savings heuristic and a precedence relationship among the genes in a chromosome. The difference of their algorithm is in the representation and the role of the mutation. The representation includes a vector of evolutionary strategy in addition to the solution vector and both components are evolved via crossover and mutation operators. The search is mainly driven by mutation, based on 2-opt, Or-opt, λ-interchanges, and special Or-opt for route elimination. Crossover is used to modify the initially randomly created mutation codes. Only one offspring is created through crossover. Fitness values are used to select predetermined number of offspring among created offsprings.

Tan *et al.* (2001) propose a GA approach in which the genetic operators are applied directly to solutions, represented as integer strings. The differences of the algorithm lie in the determination of customers served by different routes and the crossover. The basic grouping is determined by the Solomon's insertion heuristic and λ-interchanges are used to create alternative groupings. Crossover includes randomly choosing two cut points and performing a series of swapping operations.

3.4.3.4. Miscellaneous Algorithms

Rochat and Taillard (1995) used a probabilistic local search method based on intensifying the solution, which is in some ways similar to the SA approach. First, with the proposed local search, *I* different solutions are generated. The generation of *I* initial solutions creates a set *T* of tours. Second, good tours are extracted. The extraction of tours, followed by the optimization with the local search and the insertion of the new tours in *T* is repeated until a stopping criterion is met.

Kilby *et al.* (1999) used a memory-based metaheuristic, Guided Local Search (GLS). In GLS, the cost function is modified by adding a penalty term, that is, escaping form local optima is done by penalizing solution features. As local search neighborhoods, they use 2-opt exchanges.

In Potvin and Robillard (1999), a combination of a competitive neural network and a GA is described. They use a competitive neural network to cluster the customers. For every vehicle, a weight vector is defined. Initially, all weight vectors are placed randomly close to the depot. Then, customers are selected. For each cluster, the distance to all weight vectors are calculated and closest weight vector is updated by moving it closer to the customer.

Braysy *et al.* (2000) describe a two-step evolutionary algorithm based on the hybridization of a GA consisting of several local searches and route construction heuristics inspired form the studies of Solomon (1987). At the first step, a GA based on the studies of Braysy (1999) and Berger *et al.* (1998) is used. The second step consists of an evolutionary metaheuristic that picks every pair of routes in random order and applies randomly one of the four local search operators or route construction heuristics.

Tan *et al.* (2001) propose an artificial intelligence heuristic which can be interpreted as the hybrid combination of SA and TS. During the process, if a move is not a tabu and satisfies the SA criterion, it will be accepted and then the search is restarted from the beginning of a new current solution after updating the tabu list and SA parameters.

3.5. Ant Colony Based Approaches

In this section, ACO approaches for solving the VRPs will be discussed.

3.5.1. ACO for CVRP

Bullneheimer *et al.* (1998) applied the AS to the VRP with one central depot and identical vehicles for the first time (Figure 3.1). They set the number of ants (*m*) equal to the number of cities (*n*). Initially, each ant is placed at each customer. Then, ants construct vehicle routes by successively selecting cities, until all cities have been visited. When there is no feasible city to visit, the depot is selected and a new route is started. City *j* is selected after city *i* according to following random-proportional rule:

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha}.[\eta_{ij}]^{\beta}.[\mu_{ij}]^{\gamma}.[\kappa_{ij}]^{\lambda}}{\sum\limits_{k \in allowed_k}[\tau_{ik}]^{\alpha}.[\eta_{ik}]^{\beta}.[\mu_{ik}]^{\gamma}.[\kappa_{ik}]^{\lambda}} & , j \in allowed_k \\ 0 & , \text{otherwise} \end{cases}$$

$$\eta_{ij} = 1/d_{ij}$$

μ_{ij}: Savings of visiting customer *j* after customer *i*

$$\mu_{ij} = d_{i0} + d_{0j} - d_{ij}$$

κ_{ji}: Capacity utilization through the visit of customer *j* after customer *i*

$$\kappa_{ij} = \frac{Q_i + q_j}{Q}$$

Q_i = Total capacity used including the capacity requirement of customer *i*

γ: Relative influence of the savings

λ: Relative influence of κ_{ji}

After routes are constructed using the proposed approach, 2-opt heuristic is applied to each route. Then, pheromone trail on arc (*i, j*) is updated according to:

$$\tau_{ij} = \rho.\tau_{ij} + \sum_{k=1}^{m}\Delta_{ij}^{k} + \sigma.\Delta_{ij}^{*}$$

If arc (*i, j*) is used by the *k*-th ant, the pheromone trail on that are increased by $\Delta_{ij}^{k} = 1/L_k$. In addition, if arc (*i, j*) is on the so far best route, it is emphasized as if σ elitist ants used it. Each elitist ant increases the pheromone trail by $\Delta_{ij}^{*} = 1/L_*$

```
STEP I : Initialize
STEP II: For $I^{max}$ iterations do:
        a)  For each ant $k = 1,...,m$ generate a new solution
        b)  Improve all vehicle routes using the 2-opt heuristic
        c)  Update the pheromone trail
```

Figure 3.1 An algorithmic skeleton for ACO algorithm applied to the CVRP

Bullneheimer *et al.* (1999) introduced an improved ACO algorithm for the VRP with one central depot and identical vehicles (Figure 3.1). Differences of this approach from Bullneheimer *et al.* (1998) are in random proportional rule and pheromone trail update. They calculated the random proportional rule using equation 2.1. However, following parametrical savings function is used for the visibility:

$$\eta_{ij} = d_{i0} + d_{0j} - gd_{ij} + f\left|d_{i0} - d_{0j}\right| = s_{ij} - (g-1)d_{ij} + f\left|d_{i0} - d_{0j}\right|$$

After an artificial ant has constructed a feasible solution, ants are ranked according to solution quality. Only the best ranked and elitist ants are used to update the pheromone trails. This update is done using equation 2.9. They also used candidate lists for the selection of customers. Candidate lists are formed using nearest neighborhood.

Bell and McMullen (2003) used ant colonies to solve the CVRP. Differences of this approach from Bullneheimer *et al.* (1998) are in selection the next customer and pheromone trail update. Candidate lists are also formed using nearest neighborhood. Selection of the next customer j is made using ACS approach. Thus, using equations 2.5 and 2.6., each ant may either follow the most favorable path or randomly select a path to follow based on a probability distribution. Trail updating includes local updating of trails after each selection and global updating of the best solution route after all routes are constructed. These are respectively done with the following equations:

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

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

Doerner *et al.* (2001) proposed the savings based ant system approach (SbAS). The basic structure is identical to Bullneheimer *et al.* (1999), but they use the savings algorithm to calculate visibility. The attractiveness is calculated by:

$$\xi_{ij} = \left[s_{ij}\right]^{\beta}\left[\tau_{ij}\right]^{\alpha}$$

where s_{ij} is the savings of visiting customer j after customer i. Initially attractiveness values are sorted in non-increasing order and k-best combinations are considered at each decision step. If $allowed_k$ denotes the set of k feasible combinations (i, j) yielding the largest ξ_{ij}, the decision rule is given by:

$$p_{ij} = \begin{cases} \dfrac{[\xi_{ij}]}{\sum_{k \in allowed_k} [\xi_{ik}]} & , j \in allowed_k \\ 0 & ,\text{otherwise} \end{cases}$$

After solutions are constructed, only the best ranked and elitist ants are used to update the pheromone trails. This update is done using equation 2.9. Then ξ_{ij}s are calculated and sorted in non-increasing order.

Marc Reimann *et al.* (2004) presented D-Ants for solving large scale VRPs. This approach is based on the fact that the VRP is a generalization of the TSP and built on the SbAS proposed by Doerner *et al.* (2001). D-Ants decomposes the set of tours that constitute the complete problem into a number of smaller sets of tours and solves these subproblems using the SbAS (Figure 3.2).

```
procedure D-Ants {
    Read the input data;
    Initialize the system (parameters and global pheromone matrix);
    repeat {
        for a prespecified number of iterations {
            Solve the complete problem using the SbAS; (Step I)
        }
        for the best solution found so far {
            Compute the center of gravity of each route
            according to the modified Miehle algorithm; (Step II)
        }
        Decompose the best solution into a prespecifed number of subproblems
        by applying the Sweep algorithm to the centers of gravity; (Step III)
        for each subproblem {
            for a prespecified number of iterations {
                Solve the subproblem with the SbAS
                by the relevant part of global pheromone matrix locally; (Step IV)
                If applicable, update best solution; (Step V)
            }
        }
        Update the global memory (global pheromone matrix);
    }until a stopping criterion is met;
}
```

Figure 3.2 An Algorithmic skeleton for D-Ants algorithm

D-Ants consists of five main steps. First, the initial solutions are generated by the SbAS. After finding the best solution, a decomposition leading to subproblems with geographically close tours is found. Closeness of two tours is computed by the polar angle between the centers of gravity of the tours and the depot. The sweep algorithm is used to cluster the centers of gravity. Thus, in the second step the center of gravity for each vehicle route of the best solution found so far is computed. Distances are weighted by the demands and the coordinates of the center are iteratively adjusted until the change in the weighted distance of all customers to the center is minimal. Third, the sweep algorithm is applied to cluster the nodes corresponding to the centers of gravity. Fourth, subproblems are solved using the SbAS. Finally, pheromone information is changed locally after iterations of subproblem and after finding the best solution.

3.5.2. ACO for VRPTW

Gambardella *et al.* (1999) presented Multiple Ant Colony System for Vehicle Routing Problem with Time Windows (MACS-VRPTW), an ACO based approach for solving the VRPTW. MACS-VRPTW has a multiple objective function and both objectives are optimized simultaneously by the coordination of two ant colonies. The first colony, ACS-VEI, reduces the number of vehicles used while the second, ACS-TIME, optimize the travel times of the feasible solutions found by ACS-VEI. However, they use independent pheromone trail values.

Initially, a feasible VRPTW solution, ψ^{gb}, is found using a nearest neighbor heuristic. ACS-VEI tries to find a feasible solution with one vehicle less than the number of vehicles used in ψ^{gb}. ACS-TIME tries to minimize the total travel time of ψ^{gb} that use as many vehicles as vehicles used in ψ^{gb}. ψ^{gb} is updated each time one of the colonies finds an improved fesible solution. When the improved solution contains fewer vehicles than ψ^{gb}, the process is restarted with the reduced number of vehicles.

Before constructing routes, MACS-VRPTW makes the VRP similar to the TSP by duplicating the depot a number of times equal to the number of vehicles and setting distances between copies of the depot to zero.

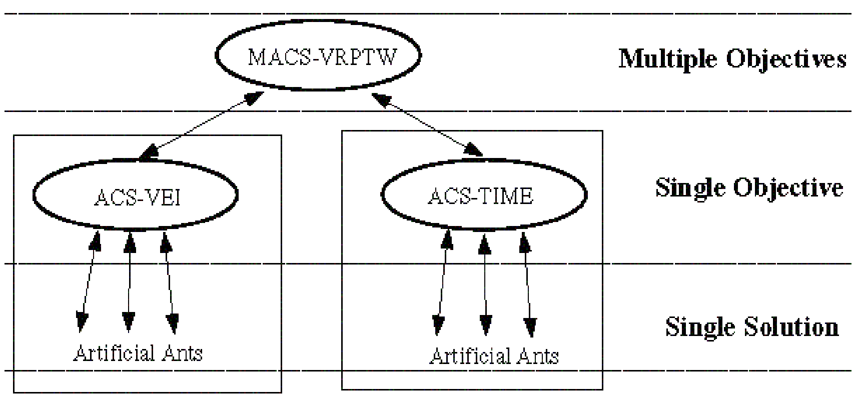

Figure 3.3 Structure of the MACS-VRPTW

ACS-VEI and ACS-TIME use a similar constructive procedure to ACS 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 city that does not violate time window constraints and vehicle capacities. An ant positioned at city *i* chooses probabilistically the next city *j* to be visited by using exploration and exploitation mechanisms. η_{ij} is computed by taking into account the traveling time t_{ij} between city *i* and *j*, the time window $[b_j,e_j]$ of city *j* and the number of time customer *j* has not been inserted in a route IN_j. *IN* are set to zero in ACS-TIME. If N_i^k is the set of feasible cities for city *i*, then $\forall j \in N_i^k$ η_{ij} is calculated by:

$$
\begin{array}{lll}
delivery\_time_j & \leftarrow & \max(current\_time_k + t_{ij}, b_j) \\
delta\_time_{ij} & \leftarrow & delivery\_time_j - current\_time_k \\
distance_{ij} & \leftarrow & delta\_time_{ij} * (e_j - current\_time_k) \\
distance_{ij} & \leftarrow & \max(1.0, (distance_{ij} - IN_j)) \\
\eta_{ij} & \leftarrow & 1/\,distance_{ij}
\end{array}
$$

Each time an ant moves from one city to another, a local update of the pheromone trail is executed.

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

In ACS-TIME *m* artificial ants construct routes $\psi^1,...,\psi^m$. If a better solution than ψ^{gb} is found, ψ^{gb} is updated. Then, the global updating is performed by:

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

ACS-VEI can produce infeasible solutions in which some customers are not visited. The solution with the highest number of visited customers is stored in $\psi^{ACS\text{-}VEI}$. A better solution is found when the number of visited customers is increased. ACS-VEI uses a vector *IN* of integers for favoring the customers that are less frequently included in the routes. The entry IN_j contains the number of time customer *j* has not been inserted in a route. In ACS-VEI, pheromone trails are globally updated by both $\psi^{ACS\text{-}VEI}$ and ψ^{gb}.

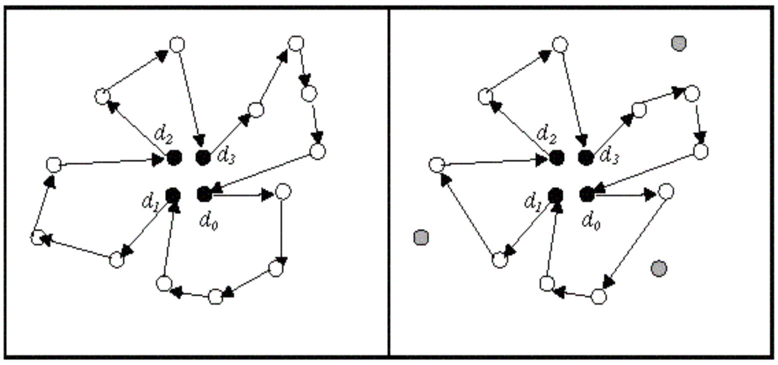

Figure 3.4 Feasible and infeasible solutions for a VRP with four duplicated depots and four vehicles

If the solution is incomplete at the end of the constructive phase, all non visited customers sorted in decreasing delivery quantities are inserted to the best feasible location (Figure 3.4).

Ellabib *et al.* (2002) proposed another AS based approach for solving VRPTW. The basic idea is to let the ACS perform its search in the space of local minima rather than in the search space of all feasible tours. The VRPTW is transformed to the TSP as proposed by Gamberdella *et al.* (1999).

The approach starts by applying a tour construction heuristic for creating a good initial solution and then let the ACS operate on the search space of local optima to guide search toward the global optimum. The ant constructive procedure is similar to the ACS constructive procedure designed for the TSP in Dorigo and Gamberdella (1997). In this procedure, each ant starts from a randomly chosen depot and moves to the feasible unvisited customer based on the transition rule until it finishes all the remaining unvisited customers. At each step, exploration and exploitation mechanism is applied for the diversification and intensification balance, visibility is computed for the transition rule, and the pheromone of the selected edge is updated locally. The global update rule is update at the end of all ant tours in which the pheromone of the best solution edges is updated. However, the amount of pheromone updated does not only depend on the length of the tour as considered in TSP but on the number of vehicles. Insertion and the nearest neighbor heuristics are applied to generate the initial solution for the ACS.

The insertion heuristic considers the insertion of an unvisited customer *u* between two adjacent customer i_{p-1} and i_p in a partially finished route. It is focused on the most effective Solomon the sequential insertion heuristic called (I1) (Solomon, 1987).This heuristic applies two criteria one for selecting the best position of the unvisitied customer and the other for the customer who has the best cost. The cheapest insertion cost and the associated insertion place for each unvisited customer are calculated using the following equations:

$C_0 = l_i + d_{0i}$

$C_{11} = d_{iu} + d_{uj} - \mu.d_{ij}$

$C_{12} = b_{ju} - b_j$

$C_1 = \alpha_1.C_{11} + \alpha_2.C_{12}$

$C_2 = \lambda.d_{ou} - C_1$

where, $\alpha_1 + \alpha_2 = 1$, $\mu \geq 0$, and $\lambda \geq 0$

C_0 : Cost of the first customer inserted in a new route

C_1 : Cost of the best position

C_2 : Cost of best customer

l_i : Latest service time of customer *i*

d_{0i} : Distance from the depot to customer *i*

d_{ij} : Distance between the customer *i* and *j*

b_j and b_{ju} : Beginning of service before and after the insertion

Nearest Neighbor algorithm starts every route by finding the closest unvisited customer. Three types of the cost functions are presented. The inverse of the cost is used as the visibility measure.

1. First visibility function is introduced by Solomon (1987). It is calculated considering the distance between customers *i* and *j*, the difference between the completion time of service at customer *i* and the beginning time of service at customer *j*, and the urgency of delivering to customer *j*.

2. The second function is introduced by Gamberdella *et al.* (1999). It is computed by multiplying the difference between the completion time of service at customer *i* and the beginning time of service at customer *j* by the urgency of delivery to customer *j*.

3. The third function is based on the difference between the position angle of the current customer and the candidate customer is introduced.

In order to solve Solomon benchmark problem instances different combinations of initial solution heuristics and the visibility functions are used. The solution quality is based on minimizing the number of routes followed by the total distance.

3.5.3. ACO for Dynamic VRP

Montemanni *et al.* (2003) proposed an algorithm for the Dynamic VRP (DVRP). The algorithm is based on the decomposition of the DVRP into static VRPs. In this algorithm, event manager receives new orders and keeps track of the served orders and vehicle. The working day is divided into time slices. For each of them a static VRP is created. New orders received during a time slice are considered at the end of that slice. At each time slice, customers whose service time starts in that time slice are assigned to the vehicles. A vehicle will wait at its last committed customer until all the customers are served or all vehicle capacity used. A new static problem is then considered. The method similar to Gambardella *et al.* (1999) is applied to solve static VRPs. The only difference is that visibility is calculated as the inverse of the distance. Once a time slice is over and the relative static problem has been solved, pheromone trail on each arc (*i, j*) is updated by the following equation:

$$\tau_{ij} = (1-\gamma_r)\tau_{ij} + \gamma_r \tau_o$$

where γ_r is a new parameter introduced to adjust pheromone conservation.

3.6. A Revised Ant Colony System Approach to the VRPTW

In this section, a revised ant colony algorithm (RACS) for the VRPTW is proposed. It is influenced by the classical ACS approach of Dorigo *et al.* (1997) for the TSP.

3.6.1. Candidate List

A candidate list is used in order to reduce the computation time and tour length. The candidate list of each customer is formed as follows: In the ACS, visiting customer *j* after the current customer *i* is based on the amount of both the pheromone trails τ_{ij} and the visibility η_{ij} on arc (i, j). Therefore, at each customer *i* candidate set $\Omega(i)$ is formed by taking *k* feasible customers with the largest attractiveness $\varphi_{ij} = \tau_{ij} . [\eta_{ij}]^{\beta}$.

When forming the initial candidate lists, it is assumed that service at customer *i* starts at time a_i and then η_{ij} are calculated for each feasible customer *j*. As some arcs are reinforced through the local and global update of the pheromone information, the attractiveness values φ_{ij} change. Thus, pheromone values that was initially high but not on the arcs of good solutions will decrease, while arcs with initially low values that appeared in good solutions will become more attractive. On the other hand, forming candidate lists after each local update is time consuming. Therefore, after each global pheromone trail update, the candidate list of each customer is formed again.

3.6.2. Initial Pheromone Trails

In most of the ant colony based algorithms to VRP, initial pheromone trails τ_0 is set equal to the inverse of the best known route distances found for the particular problem. However, it was found that $\tau_0=1/nL_{initial}$, where $L_{initial}$ is the length of the initial solution and *n* is the number of customers, can generate the shortest routes. In this way, solution made independent from the previous solutions.

When the initial route is constructed, it is started at the depot and the customer with the highest φ_{0j} value is selected as the first customer to be visited. Then, the tour is constructed by selecting the not yet visited feasible customer with the highest φ_{ij} at each time. A customer is infeasible if it violates either the capacity or the time window constraints. If no feasible customer is available then it is returned to the depot and a new route is started. This process continues until all customers are visited. The result of this is a set of tours through all customers.

3.6.3. Visibility

In TSP, the tour length is determined only by the distance between two customers. So, the visibility is calculated as the inverse of the distance between customers. However, in VRPTW, not only the distance between two customers but also customers' distance to the depot and the time window associated with the customer to whom the ant is considered to move are the essential characteristics of the tour length. Hence, these three characteristics are considered in calculating the visibility.

Savings measure proposed in Clarke *et al.* (1964) is used in order to consider the customers' distances to depot. Savings measure s_{ij} is calculated by:

$$s_{ij} = d_{i0} + d_{0j} - d_{ij} \tag{3.11}$$

where d_{ij} (d_{i0}) denotes the distance between customers *i* and *j* (the depot). Thus, s_{ij} contains the savings of serving customer *i* and *j* on the same route instead of serving them on different tours.

The higher savings value favors visiting customer *j* after customer *i* while the longer distance value prevents it. Thus, the savings per unit distance traveled between customers measures the attractiveness of visiting customer j after customer i and is calculated as follows:

$$\mu_{ij} = \begin{cases} (d_{i0} + d_{0j} - d_{ij}) / d_{ij}, & \text{if } d_{i0} + d_{0j} - d_{ij} \geq 1 \\ 1/d_{ij}, & \text{otherwise} \end{cases} \tag{3.12}$$

Since a high value of μ_{ij} indicates that visiting customer j after customer i is a desired choice the tour length is expected to be shorter if the probability of moving from customer i to customer j increases with μ_{ij}.

Furthermore, as VRPTW is a time window restricted problem, the tendency to visit a customer j with the smaller earliest and latest service starting times is more important. Thus, higher priority is given to that customer. The priority rule κ_{ij} is calculated as follows:

$$\kappa_{ij} = t_w (b_j - t_c) \tag{3.13}$$

where t_w is the waiting time and t_c the current time. t_w is obtained as follows:

$$t_w = \begin{cases} t_{ij}, & \text{if } t_c + t_{ij} \geq a_j \\ a_j - t_c, & \text{otherwise} \end{cases} \tag{3.14}$$

The visibility of selecting customer j after customer i is computed by:

$$\eta_{ij} = \begin{cases} \dfrac{\mu_{ij}}{\kappa_{ij}}, & \text{if } \dfrac{\kappa_{ij}}{\mu_{ij}} \geq 1 \\ 1, & \text{otherwise} \end{cases} \tag{3.15}$$

3.6.4. Route Construction Process

It is assumed that the number of ants is equal to the number of customers and initially, each ant is positioned at each customer. Then, each ant constructs its own tour by successively selecting a not yet visited feasible customer. The choice of the next customer to visit is based on the information of both the pheromone trails and the visibility of that choice given in equation (3.16):

$$\varphi_{ij} = \tau_{ij}\left[\eta_{ij}\right]^{\beta} \tag{3.16}$$

τ_{ij} denotes the amount of pheromone on arc $(i\ ,\ j)$ and β is power weighting parameter that weights the consistency of arc $(i\ , j)$.

Using the following equations (3.17) and (3.18) each ant may either visit the most favorable customer or randomly select a customer to visit based on a probability distribution $p(\ i\ , j)$ (Dorigo *et al.*, 1997)

$$p(i,j) = \begin{cases} \arg\max_{j\in\Omega(i)} \varphi_{ij}, & \text{if } q \le q_0 \\ P(i,j) \quad , & \text{otherwise} \end{cases} \tag{3.17}$$

$$P(i,j) = \begin{cases} \dfrac{\varphi_{ij}}{\sum\limits_{h\in\Omega(i)} \varphi_{ih}}, & \text{if } j \in \Omega(i) \\ 0 \quad , & \text{otherwise} \end{cases} \tag{3.18}$$

where q_0 ($0 \le q_0 \le 1$) is a parameter that determines the relative importance of exploitation versus exploration.

In order to reduce the probability of repeatedly selected customer, each time an ant moves from one customer to another the amount of pheromone on the chosen arc is reduced by applying a local updating rule given in equation (3.19). Otherwise, some arcs become dominant and same routes are constructed at all iterations.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\tau_0 \tag{3.19}$$

where ρ ($0 \le \rho \le 1$) is the trail persistence parameter.

If no feasible customer is available due to either the time window or the vehicle capacity constraint then the depot is chosen and a new route is started. This process is executed until all customers have been visited.

3.6.5. Global Pheromone Update

Once all ants construct their tours, the best λ tours are chosen. Because of the computation time, 2-opt procedure is only applied to these best tours to improve solutions (Croes, 1958). Then the global updating rule based on ranked based version is applied as follows (Bullnheimer, 1999):

$$\tau_{ij} = (1-\rho).\tau_{ij} + \sum_{r=1}^{\lambda-1} \Delta\tau_{ij}^{r} + \lambda.\Delta\tau_{ij}^{*} \tag{3.20}$$

If an arc is used by the r^{th} best ant, the pheromone value on arc (*i, j*) will be increased by $\Delta\tau_{ij}^{r} = (\lambda - r)/L_r$, where L_r is the tour length of the r^{th} best ant. Also, the best solution found so far is increased if λ ants had traversed it by an amount $\Delta\tau_{ij}^{*} = 1/L^{*}$, where L^{*} is the length of the best solution obtained so far.

As mentioned in the section 3.6.1, after the global pheromone update, the attractiveness values φ_{ij} are calculated with the new pheromone information as in equation 3.16, and new candidate lists are formed.

3.7. Computational Study

In this section, experimental results of applying the proposed approach to solve VRPTW are presented.

3.7.1. Benchmark Problems

Solomon's (1987) problems are used to test the performance of the proposed algorithm since they provide a common benchmark for the majority of algorithms on the literature.

There are 56 problems made up of 100 customers located in 100*100 unit plane. The benchmark set contains six different subsets called R1, R2, RC1, RC2, C1, and C2. Vehicle capacity, customer time windows, service time, and coordinates vary so as to cover all configurations as thoroughly as possible. Thus, customers are randomly distributed in R1 and R2, clustered in C1 and C2. For groups RC1 and RC2, the clustered and random distributions are mixed. Problem sets R1, C1, and RC1 have a short scheduling horizon, narrow time windows, and low vehicle capacity. On the other hand, problem sets R2, C2, and RC2 have large scheduling horizon, wide time windows, and high vehicle capacity. In these data sets, travel times correspond to Euclidean distances.

3.7.2. Experiments on Solomon's Data Instances

The algorithm is coded in Visual C++. Firstly, the parameters were initialized. First parameter is the number of iterations. Its value affects the solution quality and computational time. Bigger number of iterations increases the probability of reaching better solutions at the expense of higher computational time. By applying several runs for various problem instances, it has been observed that there is almost no improvement after the 5000th iterations. So, the number of iterations is set as 5000.

By applying experimental runs to different problems, it turns out that very small evaporation rates (like 0.0001 and 0.001) do not guarantee diversifying the solution to a new point. So, the search may not escape from the local optima using small evaporation rate. Based on the initial runs, ρ is set to 0.1. Also, it has been found out from the initial runs that setting $k = 15$, $q_0 = 0.75$, $\beta = 2$, and 6 elitist ants generate better solutions.

In order to test the performance and solution quality of the algorithms proposed, the results have been compared with the best known results of the Solomon instances in the literature. The best published results were obtained from the web page of Marius M. Solomon [63]. Table 3.1 reports the best results found by our algorithm and the best published results. In this table, NV means number of vehicles used and TD means travel distance. Gaps are the deviations from the best known.

Table 3.1 Comparison of the results of the RACS with the best known

	Best of		**Best Known**		**Gap**		**Best of**		**Best Known**		**Gap**
	TD	**NV**	**TD**	**NV**			**TD**	**NV**	**TD**	**NV**	
c101	828,937	10	828,94	10	0	**c201**	591,557	3	591,56	3	0
c102	851,27	10	828,94	10	0,0269	**c202**	591,557	3	591,56	3	0
c103	873,337	10	828,06	10	0,0547	**c203**	600,206	3	591,17	3	0,0153
c104	841,527	10	824,78	10	0,0203	**c204**	591,557	3	590,6	3	0,0016
c105	828,937	10	828,94	10	0	**c205**	588,876	3	588,88	3	0
c106	832,268	10	828,94	10	0,004	**c206**	588,49	3	588,49	3	0
c107	832,25	10	828,94	10	0,004	**c207**	588,286	3	588,29	3	0
c108	832,25	10	828,94	10	0,004	**c208**	588,32	3	588,32	3	0
c109	859,91	10	828,94	10	0,0374						
r101	1715,79	20	1645,79	19	0,0425	**r201**	1276,1	5	1252,37	4	0,0189
r102	1556,11	19	1486,12	17	0,0471	**r202**	**1169,19**	5	1191,7	3	-0,019
r103	1326,92	15	1292,68	13	0,0265	**r203**	1001,37	4	939,54	3	0,0658
r104	1052,18	11	1007,24	9	0,0446	**r204**	**787,421**	4	825,52	2	-0,046
r105	1431,65	15	1377,11	14	0,0396	**r205**	1068,75	4	994,42	3	0,0747
r106	1287,64	13	1251,98	12	0,0285	**r206**	982,841	3	906,14	3	0,0846
r107	1158,24	11	1104,66	10	0,0485	**r207**	923,024	3	893,33	2	0,0332
r108	1021,85	10	960,88	9	0,0635	**r208**	778,429	3	726,75	2	0,0711
r109	1231,91	12	1194,73	11	0,0311	**r209**	975,093	4	909,16	3	0,0725
r110	1153,83	12	1118,59	10	0,0315	**r210**	1007,77	4	939,34	3	0,0728
r111	1131,86	11	1096,72	10	0,032	**r211**	**851,125**	3	892,71	2	-0,047
r112	1007,06	10	982,14	9	0,0254						
rc101	**1679,38**	15	1696,94	14	-0,01	**rc201**	**1361,04**	6	1406,91	4	-0,033
rc102	**1548,31**	14	1554,75	12	-0,004	**rc202**	**1207,43**	5	1367,09	3	-0,117
rc103	1318,92	11	1261,67	11	0,0454	**rc203**	1056,62	5	1049,62	3	0,0067
rc104	1184,48	11	1135,48	10	0,0432	**rc204**	866,45	3	798,41	3	0,0852
rc105	**1594,3**	15	1629,44	13	-0,022	**rc205**	**1278,68**	6	1297,19	4	-0,014
rc106	1425,99	15	1424,73	11	0,0009	**rc206**	1187,3	4	1146,32	3	0,0357
rc107	1313,29	12	1230,48	11	0,0673	**rc207**	1126,95	4	1061,14	3	0,062
rc108	1168,16	11	1139,82	10	0,0249	**rc208**	908,516	3	828,14	3	0,0971

On all of the 56 problem instances, the proposed approach achieved nine shorter travel distances and matched eight best-known solutions. The shorter distances are reported in boldface.

Table 3.2 compares the mean number of vehicles (MNV) and the mean travel distance (MTD) obtained by proposed algorithm to the best known solutions.

Table 3.2 Comparisons of averages on all data sets

Data Set	Proposed	Best Known
C1 MNV	10	10
C1 MTD	842.298	828.38
C2 MNV	3	3
C2 MTD	591.1061	589.8588
R1 MNV	13.25	11.92
R1 MTD	1256.253	1209.887
R2 MNV	3.82	2.73
R2 MTD	983.7375	951.9073
RC1 MNV	13	11.5
RC1 MTD	1427.469	1408.496
RC2 MNV	4.5	3.25
RC2 MTD	1124.123	1119.353

In general, the algorithm does not perform very well for problem set R1 but is efficient for problem set C2. It can be observed that the algorithm generated good results when compared to the best known in the literature. It is worth noting here that the best known solutions are obtained using various algorithms and an efficient algorithm that performs well across all problem sets does not exist.

The computational time is not the main focus of this study. Although the computational time slightly changes from problem to problem is approximately 35 minutes.

3.7.3. Comparison with Other Heuristics

In order to test the performance of the algorithms, comparison with some competing heuristics is provided in Table 3.3. Benchmark heuristics are as follows:

- Potvin and Bengio (1996): Genetic algorithm (GA)
- Tan *et al.* (2001): Tabu search (TS)
- Li and Lim (2003): Simulated annealing-like restarts (SA)

Table 3.3 Comparisons of average travel distances of heuristics on all data sets

Data Set	Proposed	GA	TS	SA
C1	842.298	838.11	870.87	828.38
C2	591.1061	589.9	634.85	589.86
R1	1256.253	1296.83	1266.37	1215.06
R2	983.7375	1117.70	1080.23	953.55
RC1	1427.469	1446.2	1458.16	1385.57
RC2	1124.123	1360.60	1293.38	1142.48

The results of all three algorithms proposed are obviously better than the genetic algorithm proposed by Potvin and Bengio and TS proposed by Tan *et al.*(2001). However, the results are relatively worse than the simulated annealing-like restarts of Li and Lim (2002).

The proposed algorithm performs better than TS in all problem sets. It provides competitive results to GA in problem sets C1 and C2 and significantly out performs in R1, R2, RC1 and RC2. The results are slightly worse than those of SA in C1, C2, R1, and R2. On the other hand, it performs SA in problem set RC2.

4. VEHICLE ROUTING PROBLEM WITH SIMULTANEOUS PICK-UP AND DELIVERY

The VRP with Pick-ups and Deliveries (VRPPD) is an extension to the VRP where the vehicles are not only required to deliver goods to customers but also to pick some goods up at customer locations. Customers receiving goods are called linehauls, while customers sending goods are called backhauls. The objective function of the VRPPD is either to minimize the total distance traveled by the vehicles or the number of vehicles used, subject to maximum distance and maximum capacity constraints on the vehicles (Nagy and Salhi, 2004).

VRPPD is classified into three groups:

Delivery First, Pick-up Second VRPPD: Vehicles pick up goods after they have delivered their goods.

Mixed Pickups and Deliveries: Linehauls and backhauls can occur in any sequence on a vehicle route.

Simultaneous Pick-ups and Deliveries: Vehicles simultaneously deliver and pick-up goods.

Delivery-first pickup-second and mixed VRPPD problems are jointly referred to as the vehicle routing problem with backhauling (VRPB). Each customer has either a pick-up or a delivery demand to be satisfied. Products to be delivered are loaded at the depot while picked up products are transported back to the depot. A set of vehicle routes has to be designed so that all customers are serviced exactly once and no "pick-up customer" is visited before any other "delivery customer" on the same route. In the VRP with backhauls and time windows (VRPBTW) each customer also must be served during her service time interval.

In this chapter, first the VRPSDP will be explained, and a linear integer programming formulation of it will be given. Then, a detailed review of the VRPSDP from the literature is given. Finally, an ACO based approach is proposed and applied to VRPSDP.

4.1. Mathematical Formulation of the VRPSDP

The problem deals with a single depot distribution/collection system servicing a set of customers by means of a homogeneous fleet of vehicles. Each customer requires two types of service: a pick-up and a delivery. The critical feature of the problem is that both activities have to be carried out simultaneously by the same vehicle. Products to be delivered are loaded at the depot and products picked up are transported back to the depot. The objective is to find the set of routes servicing all the customers at the minimum cost (Angelelli and Mansini, 2001).

From a practical point of view VRPSDP models situations such as distribution of soft drinks, laundry service of hotels where the customers are typically visited only once but for a double service, grocery stores where reusable specialized pallets/containers are used for the transportation of merchandise. Also, regulations force companies to take responsibility for their products throughout their lifetime.

Mathematically, VRPSDP is described by a set of homogenous vehicles V, a set of customers C, and a directed graph G (N, A). $N = \{0,...,n+1\}$ denotes the set of vertices. Each vehicle has capacity Q and each customer i has delivery and pick-up requests d_i and p_i, respectively. The graph consists of $|C|+2$ vertices where the customers are denoted by $1,2,\dots,n$ and the depot is represented by the vertices 0 and $n+1$. $A = \{(i, j): i \neq j\}$ denotes the set of arcs that represents connections between the depot and the customers and among the customers. No arc terminates at vertex 0 and no arc originates from vertex $n+1$. A cost/distance c_{ij} is associated with each arc (i, j). Finally, Q, d_i, p_i, c_{ij} are assumed to be non-negative integers.

If P is assumed as an elementary path in G, $P = \{0 = i_0, i_1, \dots, i_p, i_{p+1} = n + 1\}$, a feasible solution for our problem can be represented by a set of disjoint elementary paths originating from 0 and ending at $n + 1$. These paths visit every customer exactly once while satisfying the capacity constraints. Thus, the pick-up demands already collected plus the quantities to be delivered must not exceed the vehicle capacity. The objective is to minimize the total distance traveled by all the vehicles.

For each arc *(i, j),* where $i \neq j$, $i \neq n+1$, $j \neq 0$, and each vehicle *k*, x_{ijk} is defined as

$$x_{ijk} = \begin{cases} 1, & \text{if vehicle } k \text{ uses arc}\,(i,j) \\ 0, & \text{otherwise} \end{cases}$$

D_{ik} is the amount of the remaining deliveries carried by vehicle *k* when departing from customer *i* and P_{ik} is the amount of the collected pick-up quantities carried by vehicle *k* when departing from customer *i*. The mathematical problem is formulated as follows:

$$\min \sum_{k \in V} \sum_{i \in N} \sum_{j \in N} c_{ij} x_{ijk} \tag{4.1}$$

s.t.

$$\sum_{k \in V} \sum_{i \in N} x_{ijk} = 1 \qquad \forall j \in C \tag{4.2}$$

$$\sum_{i \in N} x_{ihk} - \sum_{j \in N} x_{hjk} = 0 \qquad \forall h \in C \;, \forall k \in V \tag{4.3}$$

$$\sum_{j \in N} x_{0jk} \leq 1 \qquad \forall k \in V \tag{4.4}$$

$$\sum_{i \in N} x_{in+1k} = \sum_{j \in N} x_{0jk} \qquad \forall k \in V \tag{4.5}$$

$$D_{ik} + P_{ik} \leq Q \qquad \forall k \in V, \forall i \in C \tag{4.6}$$

$$D_{n+1k} = 0 \qquad \forall k \in V \tag{4.7}$$

$$D_{0k} = \sum_{i \in N} \sum_{j \in N} x_{ijk} d_i \qquad \forall k \in V \tag{4.8}$$

$$P_{n+1k} = \sum_{i \in N} \sum_{j \in N} x_{ijk} p_i \qquad \forall k \in V \tag{4.9}$$

$$P_{0k} = 0 \qquad \forall k \in V \tag{4.10}$$

$$x_{ijk}(P_{ik} + p_j - P_{jk}) = 0 \qquad \forall i,j \in C, \forall k \in V \tag{4.11}$$

$$x_{ijk}(D_{ik} + d_j - D_{jk}) = 0 \qquad \forall i,j \in C, \forall k \in V \tag{4.12}$$

$$D_{ik} \geq 0 \qquad \forall i \in C, \forall k \in V \tag{4.13}$$

$$P_{ik} \geq 0 \qquad \forall i \in C, \forall k \in V \tag{4.14}$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j \in N, \forall k \in V \tag{4.15}$$

In the model above, the objective function (4.1) aims to minimize the total travel distance. Constraints (4.2) assure servicing each customer exactly once. Constraints (4.3) guarantee that if a vehicle arrives at a customer then the same vehicle leaves from it. The constraints (4.4) and (4.5) ensure that each vehicle is used at most once. The constraint set (4.6) introduces limits for vehicle loads. The constraints (4.8) and (4.10) establish that each vehicle leaves the depot fully loaded with the products to be distributed while the pick-up load is null. Conversely, the constraint sets (4.7) and (4.9) guarantee that when vehicles return back to the depot, they have distributed all their deliveries and are fully loaded with the picked up quantities. The non-linear sets of equations (4.11) and (4.12) establish that if arc (*i, j*) is visited by vehicle *k* then the quantity to be delivered by the vehicle has to decrease by d_j while the quantity picked up has to increase by p_j. Finally, (4.13) and (4.14) are nonnegative constraints.

4.2. Complexity of VRPSDP

Anily (1996) proved that the VRPB is *NP*-hard as in the following way: If $P_j = 0$ ($j \in J$),or even $P_j \leq D$ ($j \in J$) then the problem reduces to the VRP which is known to be *NP*-hard. VRPB is also *NP* –hard. As the VRPB can be considered as the special case of the VRPSDP where either the delivery demand D_j or the pick-up demand P_j of each customer equals zero. VRPSDP is also *NP* –hard.

4.3. Optimal Algorithms for the VRPSDP

To our knowledge no exact algorithms have been proposed for the VRPSDP, except some suggestions in Halse (1992) and the algorithm introduced for the VRPSDP with time windows by Angelelli and Mansini (2001).

Angelelli and Mansini (2001) implemented a branch and price approach based on a set covering formulation of the master problem. A relaxation of the elementary shortest path problem with time windows and capacity constraints is used as the pricing problem. Branch and bound is applied to obtain integer solutions. Different branching strategies and some variants of a pricing algorithm are implemented in order to test their efficiency for this problem.

4.4. Approximation Algorithms for the VRPSDP

The problem was first introduced by Min(1989). He solved a practical problem faced by a public library. In his study, there was a central depot that is responsible for supplying remote libraries with ordered books and recollecting previously delivered books from them. There were two trucks with capacity of 10500 pounds. While solving this problem, the customers are first clustered into groups. Then, the TSPs in each group are solved. The infeasible arcs are penalized by setting their lengths to infinity and TSPs are solved again.

Halse (1992) studied a number of VRP versions including a special case of the VRPSDP. He used a cluster-first routing-second approach for solving the problems. In the first stage the assignment of customers to vehicles is performed, then a routing procedure based on 3-opt is used.

Gendreau *et al.* (1999) studied the VRPSPD for a single vehicle case. First, the TSP is solved without regard to pick-ups and deliveries. Then, the order of pick-ups and deliveries on the TSP-tour is determined.

Casco *et al.* (1988) developed a load-based insertion procedure where the insertion cost for backhaul customers is based on the load still to be delivered. Salhi and Nagy (1999) modified this method by allowing backhauls to be inserted in clusters, not just one by one. This procedure is also capable of solving simultaneous problems.

Dethloff (2001) modified the approaches of Casco *et al.* (1988) and Salhi and Nagy (1999) and developed a construction algorithm based on the cheapest-insertion concept. In this approach, customers are successively inserted into routes that are constructed consecutively. First, one customer is chosen as the seed customer. Then, a route from the depot to the seed customer and back to the depot is built. For all remaining unrouted customers the value of an insertion criterion for all possible insertion positions is computed and the best of the feasible (with respect to the vehicle capacity) insertions is carried out. The insertion criterion consists of the extra travel distance, the distance of customers to the depot, and the remaining vehicle capacity after a potential insertion. This phase is repeated until no further customer can be inserted into the route. Then, the next route is built with an arbitrarily chosen seed customer of the still unrouted customers. Again, insertions are performed as described until no more insertions are feasible. This route building and inserting procedure is repeated until all customers are routed.

Nagi and Salhi (2004) proposed integrated heuristic to the VRPSDP. It consists of four phases. First, a weakly feasible initial solution is generated. Then, the generated solution is improved by using some of the improvement heuristics such as reverse, 2-opt , 3-opt, exchange, combine, split heuristics. In the third phase, routes are made feasible. In the final step, solution quality is attempted to improve.

4.5. Ant System Based Appraches

To our knowledge, there is no ant colony based approach to the VRPSDP. However, there is a number of ant colony based approaches for the VRPB. As VRPSDP is the generalization of the VRPB, in this section, various approaches for solving the VRPBs will be discussed.

4.5.1. VRPBTW

Reimann *et al.* (2002) proposed ant system based approach to the VRPBTW. In this algorithm an insertion procedure based on Solomon (1987) is used to construct solutions. The routes are constructed one by one. First, the furthest customer from the depot is selected as the seed customer for the current route. Sequentially other customers are inserted into this route until no more insertions are feasible. The customer that will be inserted is selected by the following probability:

$$P_i = \frac{\eta_i}{\sum_{h|\eta_h > 0} \eta_h}$$

For each unrouted customer, visibility is calculated as follows:

$$\eta_i = \max\left\{0, \max_{j \in R_{l_i}} \left[\left(\alpha d_{0i} - \beta(d_{ji} + d_{is_j} - d_{js_j}) - (1-\beta)(b^i_{s_j} - b_{s_j}) + \gamma.type_i \right) \frac{\tau_{ij} + \tau_{is_j}}{2\tau_{is_j}} \right] \right\}$$

where s_j is the customer visited after customer j, $b^i_{s_j}$ is the arrival time at customer s_j, if i is inserted between customers j and s_j, b_{s_j} is the arrival time at customer s_j before the insertion of customer i and R_{l_i} denotes the set of customers assigned to the current tour after which customer i could feasibly be inserted. α and β are parameters. According to this formula, a customer far from the depot is more likely to be chosen than a customer close to the depot.

After routes are constructed, swap and move procedures are applied to improve the solution. The swap operator exchanges a customer i with a customer j. The move operator ejects a customer i from its current position and insert it at another position. Only global pheromone updating is applied, and the pheromone trails are updated according to the AS_{ranked} (see section 2.4.5).

4.5.2. ACO Approach for the Mixed VRPB

Wade and Salhi (2003) used AS based approach to solve the mixed vehicle routing problem with backhauls

An approach that considers the number of customers in the neighborhood of each customer is proposed to form the candidate list. The minimum number of customers (M) contained in the candidate list is calculated by:

$$M = \{N/4, 10\}$$

Average distances of each customer i to all other customers ($\overline{d}_i$) and the average of these distances (D) are respectively calculated.

$$D = \frac{\sum_{k=1}^{N} \overline{d}_k}{N} \quad \text{where } \overline{d}_k = \frac{\sum_{j \neq i} d_{ij}}{N-1}$$

For each customer k, average distance to all other customers that fall within the range (C_k) is calculated and the average of all these distances (R) is computed:

$$R = \frac{\sum_{k=1}^{N} C_k}{N} \quad \text{where } C_k = \frac{\sum_{\{j:d_{ij}<D\}} d_{ij}}{\left|\{j : d_{ij} < D\}\right|}$$

The candidate list is constructed for each customer i by,

$$E_i = \{j = 1...m \quad \text{s.t. } d_{ij} \leq R\}$$

Also, if $|E_i| < M$ then the nearest $M - |E_i|$ customers not contained in E_i are selected.

The whole region originated from the depot is divided into a given number of sectors so that each contains an equal number of customers. If this is not possible, the remaining customers are allocated to the final sector. An ant is placed at the closest and farthest customer to the depot within each sector. A given number of ants are then placed randomly on remaining nodes in each sector.

Two different visibility functions, the choice of which depends on the remaining capacity on vehicle, are used. If the vehicle is nearly full then it would be more efficient to visit a customer that is between the current customer and the depot if possible. Thus, if the unused capacity on the vehicle is less than a given parameter then the visibility is calculated by:

$$\eta_{ij} = \frac{1}{d_{ij}.d_{j0}}$$

If the unused capacity on the vehicle is larger than a given parameter, the next customer *j* is considered with the nearest customer *k* in relation to *j* in the visibility. Feasibility of customer *k* is not checked.

$$P_{ij}^{k} = \begin{cases} \arg\max\limits_{j \in allowed_i} \left\{ [\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta} [\kappa_{ij}]^{\phi} \right\}, & \text{if } q \leq q_0 \\ p_{ij}^{k}, & \text{otherwise} \end{cases} \tag{4.16}$$

$$p_{ij}^{k} = \begin{cases} \dfrac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta} [\kappa_{ij}]^{\phi}}{\sum\limits_{k \in allowed_k} [\tau_{ik}]^{\alpha} \cdot [\eta_{ik}]^{\beta} [\kappa_{ik}]^{\phi} \cdot} & , j \in allowed_k \\ 0 & , \text{otherwise} \end{cases} \tag{4.17}$$

where $\kappa_{ij} = \dfrac{1}{d_{ij}.d_{jk}}$ and $\eta_{ij} = \dfrac{1}{d_{ij}}$

In order to reduce the possibility that arcs are selected repeatedly and to encourage exploration of the search space, a frequency based local trail update is used. That is, if an arc has been selected a greater number of times than a given percentage, of the number of solutions that have been generated since the beginning of the algorithm, then the trail value on that arc is updated according to following equation:

$$\tau_{ij} = (1 - \gamma.v)\tau_{ij} + v.\tau_{o}$$

where γ is a pheromone decay parameter in the range (0 < γ < 1) and *v* is an adjustment factor (*v* > 1). Otherwise, γ is set equal to 1

The global best solution is used to update pheromone trail values together with a maximum number of iteration best solutions, *λ*. Considering each of the *λ* iteration best solutions, only if the solution is within a given percentage of the global best solution, *θ,* is the route used to update trail values. The trail values are updated according to equation (2.9) as given by Bullnheimer et al. (1997) except that *λ* is adjusted as follows:

$$G = \left\{ s = 1,\ldots,\lambda \quad \text{s.t.} \quad \frac{(\cos(s) - \cos(best))}{\cos(best)}.100 < \theta \right\}$$

To improve the solutions obtained 2-opt and 3-opt and the shift heuristics are used. In the 2-opt and 3-opt procedure only customers which belong to the same route are considered and each route is improved independently. In the shift heuristic, one or two customers between routes are exchanged.

4.6. Computational Study

In this section, the proposed ant system based approach for VRPSDP is tested on the benchmark problem instance(s) of Min (1989) and Dethloff (2001).

4.6.1. Benchmark Problems

The first instance used for testing is the real-life problem given by Min (1989). In that problem, there are 22 customers and a depot. The vehicle capacity is 10.500, the total delivery amount equals 20.300, and the total pick-up amount is 19.950.

The second problem set used for testing is given by Dethloff (2001). Random test instances with 50 customers are generated where two different geographical scenarios are examined: In scenario SCA, the coordinates of the customers are uniformly distributed over the interval [0,100]. Half of the customers in scenario CON are distributed in the same way as in SCA while the coordinates of the other half are uniformly distributed over the interval [100/3,200/3]. Distances are measured using the Euclidean metric in both cases.

The delivery demand D_j of the customers are uniformly distributed over the interval [0,100].The pick-up demand P_j is computed by using a random number r_j that is uniformly distributed over the interval [0,1] such that P_j =(0.5+ r_j) D_j. Instances with different vehicle capacities are generated by choosing the minimal number of vehicles μ. Then, the corresponding capacity is $C = \sum_{s \in J} D_s / \mu$ where μ is chosen to be 3 or 8.

4.6.2. Experiments on Dethloff's Data Instances

The Algorithm is coded in Visual C++. By applying experimental runs to different problems, the parameters were set. Considering both the solution quality and the computational time, the number of iterations is set as 5000. In addition, to escape from the local optima, ρ is set to 0.1. Also, initial runs suggest that $k = 12$, $q_0 = 0.75$, $\beta = 1$ and 5 elitist ants generate solutions.

Min (1989) reported the objective value of his problem as 94. Dethloff (2001) reported the best solution for Min's problem as 91 with a computation time of 0.27seconds. Dethloff (2001) also reported that after 100 hours of computing time on a Pentium III 500 Mhz processor the best known solution for Min's problem was found to be 89. Our proposed algorithm obtained the solution as 89 in approximately 17 seconds on a Intel Xeon 2 GHz processor.

In order to test the solution quality of the RACS, the results have been compared with those of Dethloff's problems. For each of the Dethloff's data 10 experiments are performed similar to Dethloff (2001). Table 4.1 reports the results found using RACS in comparison to Dethloff's results. Dethloff only published the average travel distances of each data set after the 10 experiments. Therefore, we compare the average travel distances obtained by RACS with those of Dethloff's.

RACS achieves shorter away travel distances in 38 out of the 40 problem instances. Only the average travel distances of SCA 8-2 and CON 8-6 are longer than the average travel distance found by Dethloff. However, they are approximately within %1 deviation.

The results of the RACS are presented in Appendix C in detail. In the table of appendix, BTD means the best travel distance obtained after 10 experiments, NV means number of vehicles used in the best result, and MCT means mean computational time in seconds.

In sum, the algorithm performs very well for the data instances of Min (1989) and Dethloff (2001). It can be observed from the results that the RACS give better results when compared with the Dethloff.

Table 4.1 Comparison of the results found with the RACS with the Dethloffs'

Data Set	RACS	Dethloff	Deviation
SCA3-0	666,0416	689	-0,03332
SCA3-1	738,8839	765,6	-0,0349
SCA3-2	692,8085	742,8	-0,0673
SCA3-3	708,8335	737,2	-0,03848
SCA3-4	719,9005	747,1	-0,03641
SCA3-5	721,1812	784,4	-0,0806
SCA3-6	670,7185	720,4	-0,06896
SCA3-7	680,4808	707,9	-0,03873
SCA3-8	759,779	807,2	-0,05875
SCA3-9	693,2852	764,1	-0,09268
SCA8-0	1019,075	1132,9	-0,10047
SCA8-1	1129,507	1150,9	-0,01859
SCA8-2	1107,837	1100,8	**0,006393**
SCA8-3	1052,831	1115,6	-0,05626
SCA8-4	1171,683	1235,4	-0,05158
SCA8-5	1160,57	1231,6	-0,05767
SCA8-6	1028,7721	1062,5	-0,03174
SCA8-7	1100,279	1217,4	-0,09621
SCA8-8	1192,269	1231,6	-0,03193
SCA8-9	1103,998	1185,6	-0,06883
CON3-0	633,9779	672,4	-0,05714
CON3-1	567,6955	570,6	-0,00509
CON3-2	530,394	534,8	-0,00824
CON3-3	599,7221	656,9	-0,08704
CON3-4	600,0444	640,2	-0,06272
CON3-5	588,2873	604,7	-0,02714
CON3-6	516,6378	521,3	-0,00894
CON3-7	596,1911	602,8	-0,01096
CON3-8	523,8241	556,2	-0,05821
CON3-9	588,3128	612,8	-0,03996
CON8-0	916,76282	967,3	-0,05225
CON8-1	771,2668	828,7	-0,06931
CON8-2	746,0453	770,2	-0,03136
CON8-3	866,5003	906,7	-0,04434
CON8-4	875,343	876,8	-0,00166
CON8-5	827,2182	866,9	-0,04577
CON8-6	757,7431	749,1	**0,011538**
CON8-7	889,3294	929,8	-0,04353
CON8-8	814,6108	833,1	-0,02219
CON8-9	854,9211	877,3	-0,02551

5. CONCLUSION

The purpose of this study was to develop ant system based approach for solving VRPs. The proposed approach basically differ form the other ant system heuristics in the way that it forms the candidate list, and it calculates the initial pheromone values and the visibility function.

Most of the ant system based heuristics forms the candidate lists at the beginning, and do not update them. On the other hand, attractiveness of arcs depends on the pheromone values on them. As pheromone values on arcs are updated, some arcs that are not on the candidate lists may become attractive. Thus, in this study candidate lists are updated after global pheromone update procedure.

In most of the ant colony based algorithms to VRP, initial pheromone trails is calculated based on the best known route distances found for the particular problem. However, in this study it is calculated based on the feasible solution found.

Finally, visibility of an arc is calculated as a function of distance between two customers, customers' distance to the depot and the time window associated with the customer to whom the ant is considered to move.

The proposed approach has been tested for VRPTW and VRPSDP. Solomon (1987) instances are used as a benchmark for the VRPTW. The results are compared with some known heuristics and the best known results of the problems. The results of the proposed approach are generally good when compared with the benchmark heuristics, but they are not that competitive with the best published results.

This is the first study that uses ACO to solve VRPSDP. The proposed approach produce competitive results in relatively small competition time when compared to Min (1989) and Dethloff (2001).

Future work in this topic may focus on the visibility and local search method on the entire solution. Visibility has a significant importance on the solution quality. A revised heuristic will include capacity constraint or modified savings function. In this paper, the 2–opt algorithm is only applied in the route. Nevertheless, an application of a local search heuristic between routes may improve the solution quality. Although computational efficiency is not of primary concern in this study, the algorithm may be run on parallel computers to improve computational time. Also, other types of the VRP may be addressed using the same approach with little modification.

6. REFERENCES

1. Anily, S., “The vehicle-routing problem with delivery and back-haul options,” *Naval Research Logistics*, vol.43, pp.415 –434, 1996.

2. Backer, B., Furnon, V., Kilby P., Prosser P., Shaw, P., “Solving Vehicle Routing Problems using Constraint Programming and Metaheuristics,” *Journal of Heuristics*, vol.6, no.4, pp. 501-525, 2000.

3. Badeau, P.,Gendreaou, M., Guertin,F., Potvin J., Taillard E.D. “A parallel tabu search heuristic for the vehicle routing problem with time windows,” *Transportation Research,*(55), pp.109-122, 1997.

4. Braysy, O., “Genetic algorithms for the vehicle routing problem with time windows,” Technical Report, Department of Mathematics and Statistics, University of Vaasa, Finland, 1999.

5. Braysy, O., Berger, J., Barkaoi, M., “A new hybrid evolutionary algorithm for the vehicle routing problem with time windows,” Presented at the *Route 2000-Workshop,* Skodsborg, Denmark, 2000.

6. Braysy, O. and Gendreau, M., “Metaheuristics for the Vehicle Routing Problem with Time Windows,” Sintef Technical Report STF42 A01025, Department of Mathematics and Statistics, University of Vaasa, Finland, 2001

7. Bullnheimer, B., Hartl, R.F., Strauss, C., “A new ranked based version of the ant system,” Working Paper, Vienna University of Economics and Business Administration, Austria, 1997.

8. Bullnheimer, B., Hartl, R.F., Strauss, C., “Applying ant system to the vehicle routing problem,” Presented at the 2nd *International Conference on Metaheuristics*, Sophia, France, July 21-24, 1997.

9. Bullnheimer, B., Hartl, R.F., Strauss, C., “An improved ant system algorithm for the vehicle routing problem,” *Annals of Operations Research,* vol.89, pp.319-328,1999

10. Bullnheimer, B., Hartl, R.F., Strauss, C., "Applying the ant system to the vehicle routing problem," *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer, Boston 1998.

11. Chiang, W., Russell, R. "Simulated annealing metaheuristics for the vehicle routing problem with time windows," *Annals of Operations Research*, (63), pp.3-27,1996.

12. Chiang, W., Russell, R. "A reactive tabu search metaheuristic for the vehicle routing problem with time windows", *INFORMS Journal on Computing*, (9), pp.417-430, 1997.

13. Clarke, G., Wright W., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations Research*, (12), pp.568-581, 1964

14. Colorni A., M. Dorigo, V. Maniezzo, "Distributed Optimization by Ant Colonies," *Proceedings of the First European Conference on Artificial Life,* Paris, France, pp.134-142., 1992.

15. Croes, G.A., "A method for solving the traveling salesman problems," *Operations Research*, (6), pp.791-812, 1958.

16. Deitel, H.M., Deitel, P.J., *C++, How to Program*, Prentice Hall, New Jersey, 2001.

17. Desrochers, M., Desroiers, J.,Solomon, M.M., "A new optimization algorithm for the vehicle routing problem with time windows," *Operation Research*, vol.40, pp.342-354, 1992.

18. Dethloff, J., "Vehicle routing and reverse logistics: The vehicle routing problem with simultaneous delivery and pick-up", *OR Spektrum*, vol.23, pp.79-96, 2001.

19. Doerner, K.F., Hartl, R.F., Reimann, M., "Ants solve time constrained pick-up and delivery problems with full truckloads," Technical Report, Lehrstuhl für Produktion und Logistik Institut für BWL, Wien, 2000.

20. Dorigo, M., Maniezzo, V. Colorni, A., "The ant system: Optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, vol.26, pp.29-41, 1996.

21. Fisher,M.L., Jörnsten, K.O., Madsen, O.B.G., "Vehicle routing with time windows: two optimization algorithms," *Operations Research,* vol.45,no.3,1997

22. Fogel, M., *How to Solve it: Modern Heuristics*, Springer, New York, 2000.

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

24. Gambardella, L.M., Taillard, E., Agazzi, G., “MACS-VRPTW: A multıple ant colony system for vehıcle routıng problems wıth tıme wındows,” Technical Report, IDSIA, Lugano, Swıtzerland, 1999.

25. Garcia, B.D., Potvin J., Rousseau J., “A parallel implementation of the tabu search heuristic for vehicle routing problems with time window constraints,” *Computers & Operations Research,* vol.21, no.9, pp.1025-1033, 1994.

26. Gillet, E., Miller, L.R., “A heuristic algorithm for the vehicle routing dispatch problem,” *Operational Research,* 22, pp.340-349,1974.

27. Glover, F., Laguna, M., “Tabu search,” *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell, Oxford, pp.76-150, 1993.

28. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., “Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey,” in *Annals of Discrete Mathematic*s, vol.5, pp.287-326, 1979.

29. Homberger, J., Gehring, H., “Two evolutionary metaheuristics for the vehicle routing problem with time windows,” *INFOR*, (37), pp.297-318, 1999.

30. Jörnsten, K.O., Madsen, O.B.G., Sorensen, B., “Exact solution of the vehicle routing and scheduling problem with time windows by variable splitting,” Technical Report, Department of Mathematical Modeling, Technical University of Denmark, 1986.

31. Halze, K., “Modeling and solving complex vehicle routing problems,” Phd. Thesis, Department of Mathematical Modeling, Technical University of Denmark, 1992.

32. Kilby, P., Prosser,P., Shaw, P., “Guided local search for the vehicle routing problems with time windows,” *Metaheuristics: Advances and Trends in Local Search for Optimization,* Kluwer Academic Publishers, pp.473-486, Boston, 1999.

33. Kohl, N., “Exact methods for time constrained routing and scheduling problems,” Phd. Thesis, Department of Mathematical Modeling, Technical University of Denmark, 1995.

34. Kohl, N., Madsen O., “An optimization algorithm for the vehicle routing problem with time windows based on Lagrangean Relaxation,” *Operations Research*, (45), pp.395-406, 1997.

35. Kolen, A., Rinnooy A., Trienekens, H., “Vehicle routing with time windows,” *Operations Research*, (35), pp.266-273, 1987

36. Larsen, J., “Parallelization of the vehicle routing problem with time Windows,” PhD. Thesis, Technical University of Denmark, Lyngby, 1999.

37. Li, H., Lim, A., “Local Search with annealing-like restarts to solve the VRPTW,” *European Journal of Operational Research*, Corrected Proof, Article in Press., 2003.

38. Lin, S., “Computer Solutions for the Traveling Salesman Problem”, *Bell Systems Technology Journal*, vol.44, pp.2245-2269, 1965.

39. Madsen, O.B.G., “Variable splitting and vehicle routing problem with time windows,” Technical Report 1A/1988, Department of Mathematical Modeling, Technical University of Denmark, 1988.

40. Min, H., “The multiple vehicle routing problem with simultaneous delivery and pick-up points,” *Transportation Research*, vol.23-A, pp.377-386, 1989.

41. Randall, M., Montgomery J., “Candidate set strategies for ant colony optimization,” Technical Report, School of Information Technology, Bond University, QLD 4229, Australia, 2002.

42. Reimann, M., Doerner, K., Hartl, R. F., “Insertion based ants for vehicle routing problems with backhauls and time windows,”*Ant Algorithms*, Springer LNCS 2463, Berlin, pp.135–147, 2002

43. Reimann, M., Doerner, K., Hartl, R. F., “Analyzing a Uni.ed Ant System for the VRP and Some of Its Variants,” *EvoWorkshops 2003*, LNCS 2611, pp.300-310, 2003.

44. Montemanni, R., Gambardella, L.M., Rizzoli, A.E., Donati, A.V., “A new algorithm for a Dynamic Vehicle Routing Problem based on Ant Colony System,”

*ODYSSEUS 2003: Second International Workshop on Freight Transportation and Logistic*s, Palermo, Italy, 27- 30 May 2003.

45. Potvin,J., Bengio, S., “The vehicle routing problem with time windows-part II: genetic search,” *INFORMS Journal on Computing*, (8), pp.165-172, 1996.

46. Potvin, J., Kervahut, T., Garcia, B.L., Rousseau, J.M., “The vehicle routing problem with time windows; part I: tabu search,” *INFORMS Journal on Computing,* (8), pp.158-164, 1995.

47. Reimann, M., Doerner, K., Hartl, R.F., “Insertion based ants for vehicle routing problems with backhauls and time windows,” ANTS:2002, LNCS 2463, pp.136-148,2002

48. Reimann, M., Doerner, K., Hartl, R.F., “ D-Ants: Savings based ants divide and conquer the vehicle routing problem,” *Computers and Operations Research*, vol.31, pp.563-591, 2004.

49. Rochat, Y., Taillard, E.D., “Probabilistic Diversification and intensification in local search for vehicle routing,” *Journal of Heuristics*, (1), pp.147-167, 1995.

50. Savelsbergh, M.W.P., “Local search for routing problems with time windows,” Annals of *Operations Research*, (4), pp.285-305, 1985.

51. Schulze,J., Fahle, T., “A parallel algorithm for the vehicle routing problem with time window constraints,” *Combinatorial Optimization: Recent Advances in Theory and Praxis*, *Special volume of Annals of Operations Research*, 86, pp.585-607,1999.

52. Smith, R., Osman, I.H., Reeves, C.R., Smith, G.D., *Modern Heuristic Search Methods,* Wiley, New York, 1996.

53. Solomon, M.M., “Algorithms for the vehicle routing and scheduling problems with time window constraints,” *Operations Research*, vol.35, no.2, pp.254-265, 1987

54. Stützle, T., Dorigo, M., “ACO Algorithms for the quadratic qssignment problem,”, *New Ideas in Optimization*, Mc Graw-Hill, 1999a.

55. Stützle, T., Hoos, H H.., “Improving the Ant System: A Detailed Report on the MAX-MIN Ant System,” Technical Report AIDA-96-12 - Revised version,

Darmstadt University of Technology, Computer Science Department, Intellectics Group., 1996.

56. Stützle, T., Hoos, H.H., "MAX-MIN ant system and local search for combinatorial optimization problems," *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, pp. 313-329, Boston, 1999.

57. Tan, K.C., Lee, L.H., Zhu, Q.L., Ou, K., "Heuristic methods for vehicle routing problem with time windows," *Artificial Intelligence in Engineering*, (15), pp.281-295, 2001.

58. Tan, K.C., Lee, L.H., Zhu, Q.L., Ou, K., "Artificial intelligence heuristics in solving vehicle routing problems with time windows," *Engineering Applications of Artificial Intelligence*, (14), pp.825-837, 2001.

59. Thangiah, S., "Vehicle routing with time windows using genetic algorithms," Technical Report, SR4-CPSC-TR-93, 23, Computer Science Department, Slippery Rock University, Slippery Rock, PA, 1993.

60. Thangiah, S., Osman, I.H., Sun, T., "Hybrid genetic algorithm, simulated annealing and tabu search method for vehicle routing problems with time windows," Technical Report, UCK/OR94/4, Institute of Mathematics and Statistics, University of Kent, Canterbury, UK, 1994.

61. Toth, P.,Vigo, D., "Models, relaxations and exact approaches for the capacitated vehicle routing problem," *Discrete Applied Mathematics*, vol.123, no.1-3, pp.487-512, 2002.

62. Wade, A., Salhi, S., "An ant System Algorithm for the Mixed vehicle routing problem with backhauls," Kluwe Academic Publishers, Netherlands, 2003.

63. http://web.cba.neu.edu/~msolomon/problems.htm, July, 2004.

7. APPENDICES

Appendix A: Pseudo-Code for the RACS to VRPTW

Appendix B: Computational Results of the RACS for VRPSDP

	BTD	NV	MCT
SCA3-0	653,869	4	146,8
SCA3-1	721,256	4	149,8
SCA3-2	685,299	4	150,4
SCA3-3	701,922	4	151,6
SCA3-4	709,299	4	152,2
SCA3-5	716,147	4	146,3
SCA3-6	660,864	4	149,6
SCA3-7	660,78	4	148,7
SCA3-8	754,053	4	148,6
SCA3-9	683,573	4	145,7
SCA8-0	1004,87	9	165,3
SCA8-1	1098,17	9	166,4
SCA8-2	1068,35	9	168,1
SCA8-3	1027,73	9	167,3
SCA8-4	1142,25	9	164,6
SCA8-5	1140,02	9	169,2
SCA8-6	998,621	9	174,4
SCA8-7	1065,2	9	177,3
SCA8-8	1173,15	9	165,5
SCA8-9	1089,58	9	170
CON3-0	627,409	4	151,2
CON3-1	559,551	4	150,7
CON3-2	525,428	4	159,6
CON3-3	597,61	4	155,8
CON3-4	589,322	4	163,2
CON3-5	583,279	4	171,6
CON3-6	508,668	4	172,7
CON3-7	578,184	4	154,7
CON3-8	523,676	4	147
CON3-9	579,487	4	155,1
CON8-0	893,619	9	173,2
CON8-1	756,416	9	175,4
CON8-2	732,986	9	173,6
CON8-3	858,633	9	178,9
CON8-4	848,95	9	178,1
CON8-5	808,083	9	175,2
CON8-6	742,962	9	181,9
CON8-7	875,204	9	181,1
CON8-8	804,81	9	178,9
CON8-9	839,992	9	182,3