Applying Two-Stage Ant Colony Optimization to Solve the Large Scale Vehicle Routing Problem

Chia-Ho CHEN
Assistant Professor
Department of Global Marketing and Logistics
MingDao University
369 Wen-Hua Road, Peetow, Changhua 52345, Taiwan, R.O.C.
Fax: +886-4-887-9013
E-mail: chench@mdu.edu.tw

Ching-Jung TING
Associate Professor
Department of Industrial Engineering and Management
Yuan Ze University
135 Yuan-Tung Road, Chungli, Taiwan 320, R.O.C.
Fax: +886-3-4638907
E-mail: ietingcj@saturn.yzu.edu.tw

Abstract: The vehicle routing problem (VRP) is an important problem in the field of logistics management. As an NP-hard problem, the VRP real world sized instances cannot be solved to optimality within reasonable times. This research aims to develop a two-stage ant colony optimization (TACO) algorithm, which possesses a two-stage solution construction rule, to solve the large scale vehicle routing problem (LSVRP). In the first stage of the solution construction rule, the VRP is decomposed into several sub-problems (traveling salesman problem). Then each sub-problem is solved by an improved ant colony system (ACS) in the second stage of the construction rule. The performance of TACO is tested by 14 VRP and 20 LSVRP benchmark instances and compared with other meta-heuristic approaches in the literature. The computational results show that our TACO can obtain good solutions for large scale VRP instances and its performance is competitive with other well-known meta-heuristic algorithms.

Key Words: *logistics management, vehicle routing problem, two-stage ant colony optimization*

1. INTRODUCTION

Physical distribution is the set of activities concerned with efficient movement of finished goods from the end of the production operation to the consumer and contributes approximately 20% to the total costs of an average product (Reimann et al., 2004). Most problems in the field of physical distribution can be regarded as the Vehicle Routing Problem (VRP). The VRP is described as a weighted graph $G = (V, E)$, where the vertices are represented by $V = \{v_0, v_1, ..., v_n\}$ and the edges are represented by $E = \{(v_i, v_j): i \neq j\}$. In this graph model, v_0 is the central depot and the other vertices are the n customers to be served. Each customer has a non-negative demand q_i and a non-negative service time s_i. Each edge (v_i, v_j) is associated with a value d_{ij} representing the traveled distance (traveled time) between v_i and v_j. A fleet of m identical vehicles delivers goods to customers, and each vehicle route starts from and terminates at the depot v_0. Each customer v_i must be visited exactly once, and the quantity of goods to be delivered on a route should never exceed the vehicle capacity Q. The total duration of any route can not exceed an upper limit L. The objective of the problem is to minimize the total traveled distance (traveled time).

The interest in VRP comes from its practical relevance as well as from considerable difficulty to solve them exactly. The VRP belongs to NP-hard problems. Its computational time increases dramatically with the increase of problem size. Hence, the task of finding the best set of vehicle routes by solving the optimization model is computational prohibitive for real world applications. Different types of heuristic algorithms are widely used for solving VRP.

Many meta-heuristic approaches have been applied to the VRP successfully, such as Genetic Algorithms (Baker and Ayechew, 2003; Jaszkiewicz and Kominek, 2003; Prins, 2004; Wang and Lu, 2009), Simulated Annealing (Robuste et al., 1990; Alfa 1991; Osman, 1993; Breedam, 1995), Tabu Search (Osman, 1993; Taillard, 1993; Gendreau et al., 1994; Rochat and Taillard, 1995; Xu and Kelly, 1996; Rego and Roucairol, 1996; Rego, 1998; Barbarrosoglu and Ozgur, 1999), and Ant Colony Optimization (Bullnheimer et al., 1999a; Bullnheimer et al., 1999c; Gambardella et al., 1999; Reimann et al, 2004; Chen and Ting, 2006). Moreover, in the recent decade, many researchers have begun to investigate the large scale Vehicle Routing Problem (LSVRP). Golden et al. (1998) observed that most real world VRP applications involve at least 200 stops per day. They generated a set of 20 LSVRP instances, whose customer numbers are between 200 and 483. A number of researchers have tested their algorithms using these instances (Toth and Vigo, 2003; Prins, 2004; Reimann et al., 2004; Li et al., 2005; Tarantilis, 2005; Kytöjoki et al., 2007; Mester and Bräysy, 2007; Créput and Koukam, 2008).

The Ant Colony Optimization is a meta-heuristic in which a colony of artificial ants cooperates in finding good solutions to difficult discrete optimization problems. The first ACO algorithm, Ant System, was introduced by Dorigo et al. (1996). Since then several ACO algorithms have been proposed, including Ant System (Dorigo et al., 1996), Elitist Ant System (Dorigo et al, 1996), Rank-Based Ant System (Bullnheimer et al., 1999b), Max-Min Ant System (Stützle and Hoos, 2000), Approximate Nondeterministic Tree-Search procedure (Maniezzo, 1999), Hyper-Cube Framework for ACO (Blum et al., 2001), Ant Colony System (Dorigo and Gambardella, 1997), and Ant-Q (Gambardella and Dorigo, 1995). ACO algorithms have been extensively applied to combinatorial optimization problems, including Routing Problems (e.g., Traveling Salesman Problem, Vehicle Routing Problem and Sequential Ordering), Assignment Problems (e.g., Quadratic Assignment Problem, Graph Coloring Problem, Generalized Assignment, Frequency Assignment and University Course Timetabling), Scheduling Problems (e.g., Job Shop, Open Shop, Flow Shop and Project Scheduling) and Subset Problems (e.g., Multiple Knapsack, Max Independent Set, Redundancy Allocation and Set Covering). Detail descriptions of ACO algorithms and related literature review can be obtained in Dorigo and Stützle (2004). The recent trends of application for ACO were reported by Blum (2005).

If taking the central depot as the nest and customers as the food, the VRP is very similar to food-seeking behaviors of ant colonies in nature. This makes the coding of an ACO for the VRP is simple. Among the earliest studies was that of Bullnheimer et al. (1999a) who presented a hybrid ant system algorithm with the 2-opt and the saving algorithm for the VRP. Subsequently, a number of ACO algorithms were proposed to solve the VRP. The objective of this research is to develop a novel ACO, called Two-Stage Ant Colony Optimization (TACO), to solve the LSVRP. The ACO is a constructive algorithm; it has to reconstruct the initial solutions based on a probabilistic construction rule in each iteration. Different from conventional ACO, the TACO constructs the solutions using a two-stage rule. Each route of VRP can be considered as a Traveling Salesman Problem (TSP). Therefore, the main idea is to decompose large problems (LSVRP) into a number of smaller sub-problems (TSP) that can be solved both more effectively and more efficiently using an improved Ant Colony System (IACS), which was proposed by Chen and Ting (2006). The remainder of this paper is organized as follows. In the next section, we introduce the procedures of IACS for TSP. The framework and procedures of TACO are described in section 3. Section 4 presents the computational results of benchmark instances. Finally, conclusions are followed in section 5.

2. IMPROVED ANT COLONY SYSTEM

The basic idea of ACO comes from the foraging behavior observed by the colony of ants. Real ants communicate with each other using an aromatic essence, called pheromone, while they lay down on the path they traversed. The pheromone will be accumulated when more and more ants pass through the same path. Nevertheless, the pheromone will be evaporated if no ants continue to pass. The selection of the pheromone trail reflects the length of the paths as well as the quality of the food source found.

The framework of the improved Ant Colony System (IACS) for TSP is based on the Ant Colony System proposed by Dorigo and Gambardella (1997). In the classical ACS, *m* ants are initially positioned on vertices that are chosen according to some a priori assignment procedure (e.g., randomly). Each ant builds a tour by repeatedly applying a probabilistic nearest neighbor heuristic. While constructing a tour, an ant modifies the pheromone levels on the visited edges by applying a local updating rule. When all ants have completed their tours, all tours are improved by a local search method. Finally, the pheromone level on each edge is modified again by applying the global updating rule which favors the edges associated with the best tour found from the start. The IACS modifies the global pheromone updating rules and only applies local search method on one ant. The IACS mainly consists of the iteration of the following three steps:

Step 1: Each ant constructs the solution by the state transition rule and local pheromone information is updated according to each constructed solution immediately.

Step 2: Apply local search method to improve the iteration-best solution (the best one among all constructed solutions).

Step 3: Update the global pheromone information based on the global-best solution (best solution so far) and the iteration-best solution.

2.1 Solution Construction of IACS

In the IACS, when located at vertex *i*, ant *h* moves to a vertex *j* according to the so-called state transition rule, given by

$$j = \begin{cases} \arg \max\limits_{j \in U_{ig}^h} \left[\tau_{ij}^g \left(\eta_{ij}^g \right)^\beta \right] & q \le q_0 \\ V & q > q_0 \end{cases} \tag{1}$$

$$V : P_{ij}^h = \frac{\tau_{ij}^g \left(\eta_{ij}^g \right)^\beta}{\sum\limits_{j \in U_{ig}^h} \tau_{ij}^g \left(\eta_{ij}^g \right)^\beta} \tag{2}$$

where $U_{ig}{}^h$ is the set of vertices that are not visited by ant *h* at vertex *i* at iteration *g*, $\tau_{ij}{}^g$ is the pheromone of edge (*i*, *j*), $\eta_{ij}{}^g$ is the heuristic information for the ant visibility measure. Here the $\eta_{ij}{}^g$ is defined as the savings of combining two vertices *i* and *j* on one tour as opposed to serving them on two different tours. The $\eta_{ij}{}^g$ is calculated as follows:

$$\eta_{ij}^g = d_{i0} + d_{0j} - d_{ij} \tag{3}$$

where d_{ij} is the distance between vertices *i* and *j*, 0 denotes the depot.

β is the parameter that determines the relative influence of pheromone versus heuristic

information ($\beta > 0$). Moreover, q is a random number uniformly distributed in [0, 1], and q_0 is a pre-defined parameter ($0 \leq q_0 \leq 1$). If $q \leq q_0$, then the best next vertex is chosen according to eq. (1). On the contrary, the next vertex is chosen according to V which is a random variable selected according to the probability distribution given in eq. (2). In other word, with probability q_0 the ant makes the best possible move as indicated by the learned pheromone trails and the heuristic information (in this case, the ant is exploiting the learned knowledge), while with probability $(1 - q_0)$ it performs a biased exploration of the edges. Tuning the parameter q_0 allows modulation of the degree of exploration and the choice of whether to concentrate the search of the system around the best-so-far tour or to explore other tours.

2.2 Local Search of IACS

In the typical ACS, after the ants have constructed their solutions but before the pheromone is local updated, the solution of each ant is improved by applying a local search technique. However, local search is a time-consuming procedure of ACS. To save the computational time, the IACS only applies local search to the best solution among all solutions (iteration-best solution) constructed in current iteration. The idea here is that better solution may have better chance to find a local optimum via local search. In this research, the 2-opt, which is to improve a single route by replacing two of its edges by two other edges, is adopted.

2.3 Pheromone Updating Rule of IACS

The pheromone updating of IACS includes global and local updating rules. The local updating rule in eq. (4) is applied to change pheromone level of edges while constructing a solution.

$$\tau_{ij}^{g+1} = \tau_{ij}^{g} + \rho\tau_0 \quad \text{if } \{edge(i, j) \in S_h\} \tag{4}$$

where S_h is the solution constructed by ant h, $0 \leq \rho \leq 1$ is the pheromone decay parameter, and $\tau_0 = (n*L_{nn})^{-1}$ is initial pheromone level of edges, n is the number of vertices and L_{nn} is the tour length obtained by the Nearest Neighbor heuristic.

For the global pheromone update, IACS adopts an elitist strategy. The idea of the elitist strategy in the context of the IACS is to give extra emphasis to the best path found so far. Such a strategy will direct the search of all the other ants in probability toward a solution composed by some edges of the best tour itself. In the IACS, the best elitist ants' tours of the iteration, including the global-best tour (S_b) and the iteration-best tour (S_l), are allowed to lay pheromone on the edges belong to them. The idea here is to balance between exploitation (through emphasizing the global-best ant) as well as exploration (through the emphasis to the iteration-best ant). The global updating rule is described as follow:

$$\tau_{ij}^{g+1} = (1-\rho)\tau_{ij}^{g} + \rho\Delta\tau_{ij}^{g} \quad \text{if } \{edge(i, j) \in S_b \text{ or } S_l\} \tag{5}$$

where

$$\Delta\tau_{ij}^{g} = \begin{cases} [(L_w - L_b) + (L_w - L_l)] / L_w & \text{if } \{edge(i, j) \in S_b \text{ or } S_l\} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

where L_b and L_l denote the tour length of the global-best solution and the iteration-best solution, respectively, and L_w is the tour length of the worst solution at current iteration. Edges do not belong to the global-best solution and the iteration-best solution just lose pheromone at

the rate ρ, which constitutes the trail evaporation. This choice is intended to make ants searching in the neighborhood of the two best tours instead of the globally best tour to avoid the algorithm being trapped in a local optimum without finding very good solutions.

2.4 Overall Procedure of IACS

The procedures of IACS for TSP are described as follows:

```
Procedure IACS ()
Begin
    //Set parameters: B (number of ants), G (maximum number of iterations), β, ρ, q0
    Set_Parameters(B, G, β, ρ, q0);
    //Initialize the pheromone matrix A
    Nearest_Neighbor_heuristic(Lnn);
    τ0 := 1/n*Lnn;
    for i := 1 to n do
       for j := 1 to n do
          A[i, j] := τ0;
       end for
    end for
    //global-best cost (Lb)
    Lb := ∞
    for g := 1 to G do
       //iteration-best cost (Ll)
       Ll := ∞
       for h := 1 to B do
          //Construct TSP solution by ant h based on state transition rule
          TSP_Solution_Construction(Sh);
          //Implement local pheromone update based on the solution constructed by ant h
          Local_Pheromone_Update(A, Sh);
          //Record the iteration-best solution (Sl)
          if Cost(Sh) < Ll then
            Ll := Cost(Sh);
            Sl := Sh;
          end if
       end for
    //Apply 2-opt to improve the iteration-best solution
     2-opt(Sl);
    //Record the global-best solution (Sb)
    if Cost(Sl) < Lb then
       Lb := Cost(Sl);
       Sb := Sl;
    end if
    /*Implement global pheromone update based on the global-best and the iteration-best
      solutions*/
    Global_Pheromone_Update(A, Sb, Sl);
    end for
    //Output global-best solution and global-best cost (Lb)
    Return Lb, Sb;
End
```

3. TWO-STAGE ANT COLONY OPTIMIZATION

Different from typical ACO algorithms, the Two-Stage Ant Colony Optimization (TACO) adopts a two-stage solution construction rule. In the first stage (customer assignment stage), the customers are assigned to each vehicle. For each vehicle, an independent TSP is solved by IACS in the second stage (route construction stage). Furthermore, two different pheromone matrices and pheromone updating rules are used to record pheromone information for each stage, respectively. The framework of TACO can be illustrated as figure 1 and procedures of TACO are described in the following sections.

3.1 Solution Construction of TACO

In the TACO, the first-stage construction rule is to assign customers to each vehicle, while the second-stage construction rule applies the IACS to solve TSP. The heuristic information of customer assignment stage is based on the shortest distance among all edges between customer *i* and customer *l*. The customer *l* belongs to the set of a vehicle *k* and those customers which have been assigned to vehicle *k*. The pheromone information of the stage is the pheromone level of edge (*i*, *k*). On the other hand, the heuristic information of route construction stage is based on the distance of edge (*i*, *j*). The pheromone information of the stage is the pheromone level of edge (*i*, *j*). Furthermore, the solution construction rule of TACO contains IACO for TSP, so it is a time-consuming procedure. Therefore, small number of ants of TACO is adopted in this research. These two stages of solution construction rule are described as follows:

Customer Assignment Stage

To begin with, we assign one customer to each vehicle randomly, and then the remainder customers are assigned according to the state transition rule. The customer assignment stage allocates customer *i* to vehicle *k* based on the rule which can be described as follows:

$$k = \begin{cases} \arg\max_{k \in W_{g'}^{h'}} \left[\xi_{ik}^{g'} \left(\psi_{ik}^{g'} \right)^{\beta'} \right], & \text{if } q' \le q_0' \\ K, & \text{otherwise} \end{cases} \tag{7}$$

$$K : P_{ik}^{h'} = \frac{\xi_{ik}^{g'} \left(\psi_{ik}^{g'} \right)^{\beta'}}{\sum_{k \in W_{g'}^{h'}} \xi_{ik}^{g'} \left(\psi_{ik}^{g'} \right)^{\beta'}} \tag{8}$$

where $W_{g'}^{h'}$ is the set of vehicles that is used by ant h' at iteration g', and the number of vehicles *m* is determined by the following equation.

$$m = \frac{\sum_{i \in V} q_i}{Q} \tag{9}$$

$\xi_{ik}^{g'}$ is the pheromone level between customer *i* and vehicle *k*, $\psi_{ik}^{g'}$ is the reciprocal of D_{ik}, which is given by

$$D_{ik} = \min_{l \in A_k} d_{il} \tag{10}$$

where A_k is the set of customers which have been assigned to vehicle k, and the d_{il} is the distance between customer i and l. β' is the parameter that determines the relative effect of $\xi_{ik}^{g'}$ versus $\psi_{ik}^{g'}$ ($\beta' > 0$). On the other hand, the parameter q_0' determines the relative importance of exploitation eq. (7) versus exploration eq. (8).

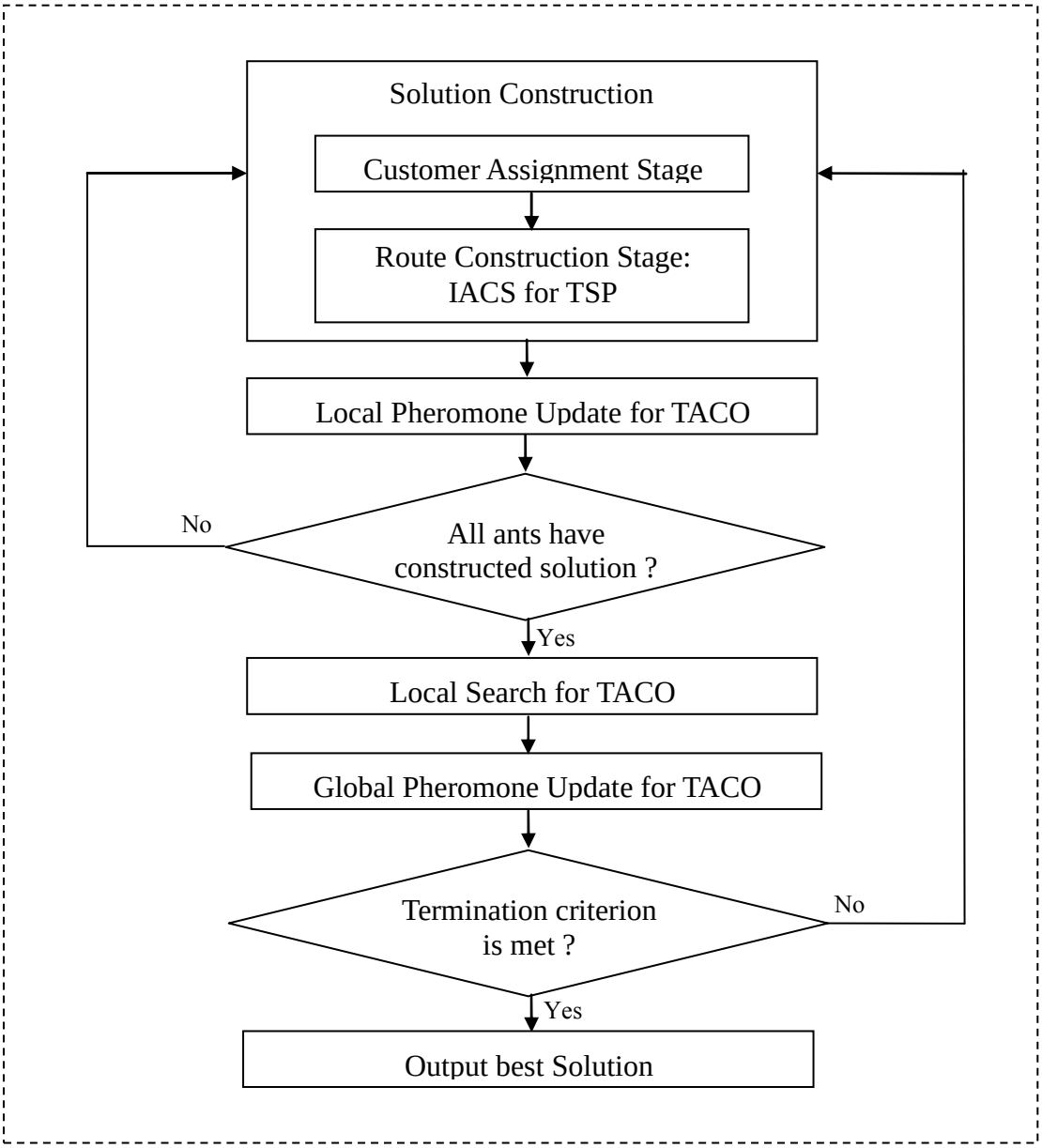

Figure 1 The framework of TACO

Route Construction Stage

After all customers are assigned to each vehicle, the construction of routes for each vehicle can be regarded as an independent TSP. Each TSP is solved by the IACS, which is described in section 2.

In this research, infeasible solutions are allowed during the search. Let Y denote the set of solutions that satisfy the VRP constraints. Each solution $y \in Y$ is represented by a set of routes such that every customer belongs to exactly one route. A solution y is evaluated by means of

the cost function $f(y) = c(y)+w_q q(y)+w_d d(y)$. $c(y)$ denotes the total traveled distance of routes, and $q(y)$ and $d(y)$ are the violation of capacity and duration constraints, respectively. w_q and w_d are positive penalty parameters. Hence, if y is feasible, the $f(y)$ is equal to the $c(y)$. Otherwise, $f(y)$ includes two penalty terms proportional to the total excess demand and excess duration of routes. Each violation is calculated as follows:

$$q(y) = \sum_{k \in W} [(\sum_{i \in V} \sum_{j \in V} q_i x_{ij}^k) - Q]^+ \tag{11}$$

$$d(y) = \sum_{k \in W} [(\sum_{i \in V} \sum_{j \in V} (d_{ij} + s_i) x_{ij}^k) - L]^+ \tag{12}$$

where $[a]^+ = \max\{0, a\}$. If vehicle k visits customer j immediately after customer i ($i \neq j$), x_{ij}^k is equal to 1; otherwise, x_{ij}^k is equal to 0.
In this research, the initial value of w_q and w_d are set to be 100. If the iteration-best solution is infeasible, the value of these two parameters will be doubled.

3.2 Local Search of TACO

After the initial solutions of the TACO are generated by the solution construction rule, two local search approaches are used to improve the solutions. The first one is insertion move between routes: remove a customer from one route, and then insert it into another route. The other one is swap move between routes: exchange two customers belonging to two different routes. In the route construction rule, the IACS has found the local optimum of TSP for each vehicle. Any neighborhood move between routes will easily violate constraints. These two move operators can change the customer assignments as well as generate less infeasible solutions.

3.3 Pheromone Updating Rule of TACO

In the TACO, the customer assignment stage and route construction stage use different pheromone matrices and pheromone updating rules, respectively. The pheromone updating rules of route construction stage is described in section 2.3. Here, we only state the pheromone updating rulers for customer assignment stage. The local pheromone updating rule of customer assignment stage is applied to change the level of pheromone matrix once, after customers are initially assigned by each ant. The local pheromone updating rule for customer assignment is as follow:

$$\tau_{ik}^{g'+1} = (1-\rho')\tau_{ik}^{g'} + \rho'\tau_0' \quad \text{if } \{edge(i,k) \in S_{h'}\} \tag{13}$$

where $S_{h'}$ is the solution constructed by ant h', $0 \leq \rho' \leq 1$ is the pheromone decay parameter, and τ_0' (a very small value) is the initial level of pheromone matrix.
The global pheromone updating rule (eqs. (14) and (15)) for customer assignment stage is only applied to the global-best solution ($T_{b'}$) and the iteration-best solution ($T_{l'}$).

$$\tau_{ik}^{g'+1} = (1-\rho')\tau_{ik}^{g'} + \rho'\Delta\tau_{ik}^{g'} \quad \text{if } \{\text{edge}(i,k) \in S_{b'} \text{ or } S_{l'}\} \tag{14}$$

where

$$\Delta\tau_{ik}^{g'} = \begin{cases} [(L_{w'} - L_{b'}) + (L_{w'} - L_{l'})] / L_{w'} & \text{if}\{\text{edge}(i,k) \in S_{b'} \text{ or } S_{l'}\} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $L_{b'}$ and $L_{l'}$ denote the objective function values of the global-best solution and the iteration-best solution, respectively; $L_{w'}$ is the objective function value of the worst solution at current iteration.

3.4 Overall Procedure of TACO

The procedures of our TACO are described as follows:

```
Procedure TACO ()
Begin
  //Set parameters: B' (number of ants), G' (maximum number of iterations), β', ρ', q0', τ0'
   Set_Parameters(B', G', β', ρ', q0'. τ0');
   //Initialize the pheromone matrix A';
   for i := 1 to n do
      for k := 1 to m do
         A'[i, k] := τ0';
     end for
   end for
   //global-best cost (Lb')
   Lb' := ∞
   for g' := 1 to G do
      //iteration-best cost (Ll')
      Ll' := ∞
      for h' := 1 to B do
          //Construct VRP solution by ant h' based on state transition rule
          VRP_Solution_Construction(Sh');
          //Implement local pheromone update based on the solution constructed by ant h'
          Local_Pheromone_Update(A', Sh');
          //Record the iteration-best solution (Sl');
          if Cost(Sh') < Ll' then
            Ll' := Cost(Sh');
            Sl' := Sh';
          end if
      end for
    //Apply insertion move and swap move to improve the iteration-best solution
     Insertion_Move (Sl');
     Swap_Move (Sl');
    //Record the global-best solution (Sb');
    if Cost(Sl') < Lb' then
       Lb' := Cost(Sl');
       Sb' := Sl';
    end if
    /*Implement global pheromone update based on the global-best and the iteration-best
      solutions*/
    Global_Pheromone_Update(A', Sb', Sl');
    end for
    //Output global-best solution and global-best cost (Lb')
    Return Lb', Sb';
End
```

```
Procedure VRP_Soluiton_Construction()
Begin
    //Assign first customer to each vehicle
    for k := 1 to m do
        do
            i := Random(n);
        while (customer i is assigned);
        Assign customer i to vehicle k;
    end for
    //Assign the remainder customers to each vehicle
    for i := 1 to n do
        if customer i is unassigned then
            //Assign customer i based on state transition rule;
            Customer_Assignment();
        end if
    end for
    for k := 1 to m do
        //Implement IACS procedure for vehicle k;
        IACS();
    end for
End
```

4. COMPUTATIONAL RESULTS

The proposed TACO for the VRP is coded in Borland C++ Builder 5.0, and run on a PC with an Athlon XP 2500+ (1.83 GHz) processor and 512 MB RAM, under the Windows XP operating system. We test the TACO by two different groups of benchmark instances, including 14 VRP instances in Christofides et al. (1979) and 20 large scale VRP instances proposed by Golden et al. (1998). According to the preliminary experiment, the parameters of TACO are set as follows: $B=n/m$, $B'=4$, $\beta=4$, $\beta'=1$, $\rho=0.1$, $\rho'=0.1$, $q_0=0.5$, $q_0'=0.5$, $G=100$ and $G'=25$.

4.1 Results for the VRP Benchmark Instances

The first group of benchmark instances, which includes 14 VRP instances, was proposed by Christofides et al. (1979). The number of customers in these instances ranges from 50 to 199. The customers in instances 1-10 are generated by a random uniform distribution, while they are clustered in instances 11-14. Instances 1-5 and 6-10 are identical, except that the total length of each vehicle route is limited for the latter problems. Instances 13-14 are the counterparts of instances 11-12 with additional route length restriction.

The information of Christofides et al.'s instances and computational results are summarized in table 1. Table 1 shows the problem size n, the vehicle capacity Q, the service time s, the maximum duration of vehicle T, the best-known solution (BKS), the best solution obtained by TACO, the relative percentage deviation over the BKS (gap (%)) and average run time (minute) over 10 runs for each instance. As shown in table 1, our TACO can achieve the BKS in 8 instances and the largest gap is 1.46%. Moreover, we compare the performance of the TACO with well-known meta-heuristic approaches in table 2. The compared methods include

the AS (Bullnheimer et al., 1998), IAS (Bullnheimer et al., 1999), SavingsAnts (Doerner et al, 2002), D-Ants (Reimann et al., 2004), SA (Osman, 1993), TABUROUTE (Gendreau et al., 1994), Granular TS (GTS) (Toth and Vigo, 2003) and IACS (Chen and Ting, 2006). As can be seen from table 2, the average gap (0.33%) of TACO is the lowest among all compared algorithms. TACO provides 10 best solutions in 14 instances and reaches 8 best known solutions. Based on the computational results, we can find that our TACO finds competitive solutions to other algorithms in terms of solution quality for small and medium VRP instances.

Table 1 The computational results of VRP instances

Pro.	*n*	*Q*	*s/T*	BKS	Best sol.	Gap (%)	CPU time (min.)
C1	50	160	0/∞	524.61	**524.61**[a]	**0.00**	0.21
C2	75	140	0/∞	835.26	837.36	0.25	0.70
C3	100	200	0/∞	826.14	**826.14**	**0.00**	1.04
C4	150	200	0/∞	1028.42	1030.08	0.16	3.01
C5	199	200	0/∞	1291.45	1310.27	1.46	7.00
C6	50	160	10/200	555.43	**555.43**	**0.00**	0.33
C7	75	140	10/160	909.68	**909.67**	**0.00**	1.19
C8	100	200	10/230	865.94	**865.94**	**0.00**	1.45
C9	150	200	10/200	1162.55	1176.61	1.21	4.49
C10	199	200	10/200	1395.85	1408.56	0.91	9.86
C11	120	200	0/∞	1042.11	**1042.12**	**0.00**	1.81
C12	100	200	0/∞	819.56	**819.56**	**0.00**	1.23
C13	120	200	50/720	1541.14	1550.70	0.62	3.22
C14	100	200	90/1040	866.37	**866.37**	**0.00**	1.76
Avg.					980.24	0.33	2.66

[a]: those numbers in bold are the solution reaches the BKS.

Table 2 Comparison of different algorithms for VRP instances

Pro.	AS	IAS	Savings Ants	D-Ants	SA	TABUROUTE	GTS	IACS	Our TACO
C1	**0.00**[a]	**0.00**	**0.00**	**0.00**	0.65	**0.00**	**0.00**	**0.00**	**0.00**
C2	4.23	1.08	0.40	0.64	0.40	0.06	0.40	**0.01**	0.25
C3	6.45	0.75	1.48	0.25	0.37	0.40	0.29	0.61	**0.00**
C4	11.57	3.22	1.21	0.89	2.88	0.75	0.47	0.84	**0.16**
C5	14.09	4.03	1.26	**1.20**	6.55	2.42	2.08	2.25	1.46
C6	1.35	0.87	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
C7	4.23	0.72	0.37	0.86	**0.00**	0.39	1.21	**0.00**	**0.00**
C8	2.34	0.09	0.48	**0.00**	0.09	**0.00**	0.41	**0.00**	**0.00**
C9	3.39	2.88	0.94	0.98	**0.14**	1.31	0.91	0.61	1.21
C10	7.80	4.00	3.07	1.41	1.58	1.62	2.86	0.99	**0.91**
C11	2.91	2.22	0.17	0.13	12.85	3.01	0.07	**0.00**	**0.00**
C12	0.05	**0.00**	**0.00**	**0.00**	0.79	**0.00**	**0.00**	1.50	**0.00**
C13	3.20	1.22	0.45	0.37	0.31	2.12	**0.28**	0.29	0.62
C14	0.40	0.08	**0.00**	**0.00**	2.73	**0.00**	**0.00**	**0.00**	**0.00**
Avg. gap(%)	4.43	1.51	0.70	0.48	2.09	0.86	0.64	0.51	0.33

[a]: those numbers in bold are the best solution among all algorithms.

Effects of computer performance are influenced by many factors such as CPU speed, memory capacity, operation system and programming language. Therefore, a fair transformation of computational time is difficult to establish. We used Dongarra (2008) tables to get a rough

idea of the relative speeds of the different computer. Because many types of computers were not included in these tables, we just use similar computers for rough estimate. The speed of the computers, measured in millions of floating-point operations per second (Mflop/s), as shown in table 3. We evaluate the scaled time to compare the computational speed of different algorithms. The scaled time is computed as follows: If T_a is the computational time and P_a is the computer power (Mflop/s) for one algorithm a, its scaled time is $(T_a/T_g)*(P_a/P_g)$, with g standing for TACO. If the scaled time of the compared algorithm is smaller than 1, the algorithm is faster than the TACO. Otherwise, it is slower than the TACO. For example, the scaled time of the AS is equal to 0.08, thus, the AS is approximate twelve times faster than the TACO. Although the computational speed is not an advantage for our TACO, its run time is acceptable.

Table 3 Comparison of computational speed of algorithms for VRP instances

Method	Reference	NBKS*	Computer	Mflop/s	Programming Language	CPU (Min)	Scaled Time
AS	Bullnheimer et al. (1998)	1	Pentium 100 MHz	10	-+	18.1	0.08
IAS	Bullnheimer et al. (1999)	2	Pentium 100 MHz	10	-	18.43	0.08
SavingsAnts	Doerner et al (2002)	4	Pentium 900 MHz	190	-	-#	-
D-Ants	Reimann et al. (2004)	5	Pentium 900 MHz	190	C	3.3	0.26
SA	Osman (1993)	2	VAX 8600 workstation	0.48	-	151.35	0.03
TABUROUTE	Gendreau et al. (1994)	5	Silicon Graphics work station	-	-	46.8	-
GTS	Toth and Vigo (2003)	4	Pentium 200 MHz	38	Fortran 77	3.8	0.06
IACS	Chen and Ting (2006)	6	Pentium 1 GHz	250	BCB 5	12.2	1.27
TACO	Ours	8	Athlon 2500+	900	BCB 5	2.66	1

*: The number of solutions which are equal to the BKS.
+: The programming language used is not provided.
#: The CPU time is not reported.

4.2 Results for the LSVRP Benchmark Instances

The second group of benchmark instances, which includes 20 large scale instances, was generated by Golden et al. (1998). The number of customers on these instances ranges from 200 to 483. Instances 1-8 have customers located in concentric circles around the depot and possess route length restriction. Instances 9-12 have customers located in concentric squares with the depot located in one corner. Instances 13-18 have customers located in concentric squares around the depot. Table 4 presents the information of LSVRP instances and computational results of TACO. The format of the table is the same as that in table 1. The computational results show that our TACO can find near-BKS for many LSVRP instances within a reasonable run time. The average gap is 0.82% and the largest gap is 1.92%. In table 5, the computational results of LSVRP instances obtained by TACO is compared with nine competitive results, including the Granular TS (Toth and Vigo, 2003), hybrid GA (Prins, 2004), D-Ants (Reimann et al., 2004), VRTR (Li et al., 2005), SEPAS (Tarantilis, 2005), TS+PR (Ho and Gendreau, 2006), VNS (Kytöjoki et al., 2007), VLNS (Pisinger and Ropke, 2007) and Memetic SOM (Créput and Koukam, 2008). As shown in table 5, the gap (%) of our TACO is only larger than those of D-ants, SEPAS and VLNS among nine compared

algorithms. Overall, the computational results reveal that our TACO is effective and competitive for LSVRP.

Table 4 The computational results of LSVRP instances

Pro.	*n*	*Q*	*T*	BKS	Best sol.	Gap (%)	CPU time (min.)
K1	240	550	650	5627.54	5655.57	0.50	7.89
K2	320	700	900	8447.92	8476.43	0.34	15.38
K3	400	900	1200	11036.22	11106.62	0.64	26.24
K4	480	1000	1600	13624.52	13658.26	0.25	40.68
K5	200	900	1800	6460.98	**6460.98**[a]	**0.00**	4.07
K6	280	900	1500	8412.8	8414.74	0.02	9.55
K7	360	900	1300	10181.75	10271.07	0.88	19.21
K8	440	900	1200	11663.55	11887.90	1.92	34.19
K9	255	1000	∞	583.39	588.70	0.91	8.24
K10	323	1000	∞	741.56	748.37	0.92	14.52
K11	399	1000	∞	918.45	923.85	0.59	24.28
K12	483	1000	∞	1107.19	1122.79	1.41	38.36
K13	252	1000	∞	859.11	867.45	0.97	10.37
K14	320	1000	∞	1081.31	1095.07	1.27	18.31
K15	396	1000	∞	1345.23	1357.34	0.90	29.81
K16	480	1000	∞	1622.69	1641.34	1.15	47.33
K17	240	200	∞	707.79	709.27	0.21	8.57
K18	300	200	∞	998.73	1008.75	1.00	16.23
K19	360	200	∞	1366.86	1381.78	1.09	25.55
K20	420	200	∞	1820.09	1847.00	1.48	39.18
Avg.						0.82	21.90

[a]: those numbers in bold are the solution reaches the BKS.

Table 5 Comparison of different algorithms for LSVRP instances

Pro.	GTS	GA	D-Ants	VRTR	SEPAS	TS+PR	VNS	VLNS	Memetic SOM	Our TACO
K1	1.93	0.34	**0.29**[a]	0.69	0.88	3.96	4.27	0.42	4.11	0.50
K2	1.24	**0.00**	0.01	0.25	0.14	4.16	0.34	0.25	0.43	0.34
K3	3.32	**0.00**	**0.00**	0.99	**0.00**	6.16	0.07	0.10	2.18	0.64
K4	9.44	**0.00**	0.55	0.98	0.10	2.56	0.05	0.08	3.05	0.25
K5	3.66	**0.00**	**0.00**	0.26	**0.00**	1.32	**0.00**	0.09	0.42	**0.00**
K6	6.54	**0.00**	**0.00**	1.51	0.02	4.07	0.03	0.04	0.83	0.02
K7	3.59	0.14	0.14	1.06	0.34	5.25	1.14	**0.00**	1.59	0.88
K8	3.20	1.42	1.42	2.20	2.34	4.35	1.79	**0.43**	3.10	1.92
K9	1.71	1.40	0.60	0.83	0.35	1.03	6.39	**0.30**	3.18	0.91
K10	1.36	1.33	1.24	1.07	**0.67**	1.01	5.83	0.99	2.54	0.92
K11	1.92	1.59	0.96	0.81	0.51	1.04	7.44	**0.46**	2.74	0.59
K12	3.61	2.40	3.04	1.88	2.10	1.56	9.20	**1.07**	5.01	1.41
K13	1.13	1.87	0.69	0.71	0.69	1.86	7.76	**0.65**	3.65	0.97
K14	1.38	0.46	1.15	1.52	**0.44**	2.24	6.83	1.30	3.49	1.27
K15	1.80	1.65	0.96	1.20	**0.65**	1.82	8.64	1.09	3.27	0.90
K16	1.83	1.74	0.77	0.79	**0.74**	1.57	7.41	1.01	3.79	1.15
K17	0.46	0.37	0.14	0.56	**0.13**	0.83	2.57	0.16	2.23	0.21
K18	1.81	1.61	**0.01**	1.16	0.82	1.89	7.89	0.37	2.80	1.00
K19	2.49	0.70	**0.02**	1.15	0.30	2.22	5.68	0.54	2.23	1.09
K20	5.26	1.45	**0.16**	1.69	0.97	2.41	6.48	0.59	3.29	1.48
Avg. gap (%)	2.88	0.92	0.61	1.07	0.61	2.57	4.49	0.50	2.70	0.82

[a]: those numbers in bold are the best solution among all algorithms

Concerning the computational speed, table 6 presents the scaled time of algorithms for LSVRP. The VNS is the fastest among compared algorithms, but its solution quality is the

worst among all algorithms. Among all algorithms, our TACO is only faster than TS+PR and Memetic SOM. This is because the local search used by TACO is a very time-consuming procedure for LSVRP. Another observation is that single solution search algorithms are faster than population-based search algorithms in terms of scaled time. Overall, our TACO is not a fast algorithm, but its run time for the LSVRP is reasonable.

Table 6 Comparison of computational speed of algorithms for LSVRP instances

Method	Reference	NBKS*	Computer	Mflop/s	Programming Language	CPU (Min)	Scaled Time
GTS	Toth and Vigo (2003)	0	Pentium 200 MHz	38	Fortran 77	17.55	0.03
GA	Prins (2004)	5	Pentium 1 GHz	250	Delphi 5	67.40	0.89
D-Ants	Reimann et al. (2004)	3	Pentium 900 MHz	190	C	49.33	0.48
VRTR	Li et al. (2005)	0	Athlon 1GHz	500	-	1.13	0.03
SEPAS	Tarantilis (2005)	2	Pentium 400 MHz	60	Visual C++	45.48	0.14
TS+PR	Ho and Gendreau (2006)	0	Pentium 4 2.53 GHz	1190	C++	39.93	2.41
VNS	Kytöjoki et al. (2007)	1	Athlon 64 3000+	1000	Visual C++ 6.0	0.02	0.001
VLNS	Pisinger and Ropke (2007)	1	Pentium 4 3 GHz	1573	-	10.80	0.86
Memetic SOM	Créput and Koukam (2008)	0	Athlon 2GHz	849	Java	39.06	1.68
TACO	Ours	1	Athlon 2500+	900	BCB 5	21.90	1

*: The number of solutions which are equal to the BKS.

5. CONCLUSIONS

This research proposed a novel two-stage ACO (TACO) for the Vehicle Routing Problem. Different from conventional ACO algorithms, the TACO adopts a two-stage solution construction rule. The first stage of solution construction rule is to decompose a difficult and large problem (VRP) into several simple and small sub-problems (TSP). Then these sub-problems are solved by an IACS process in the second stage of the rule. Two groups of benchmark instances are taken to test the performance of the TACO. For instances of small and medium size, the computational results show that our TACO has the ability to find good results with an average gap of 0.33% by one setting of parameters. On the other hand, the results also show that the performance of TACO for large scale instances is competitive when compared with other well-known meta-heuristic algorithms in terms of solution quality. Thus, we can indicate that our TACO is effective to solve the VRP and LSVRP. Concerning computational speed, the TACO is not a fast algorithm, but its run times are reasonable. In the future, the same idea of TACO could be extended to solve other variants of VRP, such as Vehicle Routing Problem with time windows (VRPTW), Vehicle Routing Problem with Backhauls (VRPB) and Multi-depot Vehicle Routing Problem (MDVRP).

REFERENCES

Alfa, A. S., Heragu, S. S. and Chen, M. (1991) A 3-opt based simulated annealing algorithm for vehicle routing problems, **Computers & Industrial Engineering, Vol. 21**, 635-639.

Baker, B. M. and Ayechew, M. A. (2003) A genetic algorithm for the vehicle routing problem, **Computer & Operations Research, Vol. 30**, 787-800.

Barbarosoglu, G. and Ozgur, D. (1999) A tabu search algorithm for the vehicle routing problem, **Computer & Operations Research, Vol. 26**, 255-270.

Blum, C. (2005) Ant colony optimization: Introduction and recent trends, **Physics of Life Reviews, Vol. 2**, 353-373.

Blum, C., Roli, A. and Dorigo, M. (2001) HC-ACO: The hyper-cube framework for the ant colony optimization, Proceedings of MIC'2001-Metaheuristics International Conference, Vol. 2, 399-403.

Breedam, A. V. (1995) Improvement heuristics for the vehicle routing problem based on simulated annealing, **European Journal of Operational Research, Vol. 86**, 480-490.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999a) Applying the ant system to the vehicle routing problem, In Voss, S., Martello, S., Osman, I. H. and Roucairol, C. (eds.), **Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization**. Dordrecht, Netherlands, Kluwer, Academic Publishers, 285-296.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999b) A new rank based version of the ant system - A computational study, **Central European Journal of Operations Research, Vol. 7**, 25-38.

Bullnheimer, B., Hartl, R. F. and Strauss, C. (1999c) An improved ant system for the vehicle routing problem, **Annals of Operations Research, Vol. 89**, 319-328.

Chen, C. H. and Ting, C. J. (2006) An improved ant colony system algorithm for the vehicle routing problem, **Journal of the Chinese Institute of Industrial Engineering, Vol. 23**, 115-126.

Créput, J.-C. and Koukam, A. (2008) The memetic self-organizing map approach to the vehicle routing problem, **Soft Computing, Vol. 12**, 1125-1141.

Dongarra, J. (2008) Performance of Various Computers Using Standard Linear Equations Software, Report CS-89-85, University of Tennessee, U.S.A.

Dorigo, M. and Gambardella, L. M. (1997a) Ant colonies for the traveling salesman problem, **BioSystems, Vol. 43**, 73-81.

Dorigo, M. and T. Stützle, (2004) **Ant Colony Optimization**. Bradford Books, MIT Press.

Dorigo, M., Maniezzo, V., Colorni, A. (1996) Ant system: Optimization by a colony of cooperating ggents, **IEEE Transactions on Systems, Man and Cybernetics Part B, Vol. 26**, 29-41.

Gambardella, L. and Dorigo, M. (1995) Ant-Q: A reinforcement learning approach to the TSP, 12th International Machine Learning Conference (ML-95), 252-260.

Gambardella, L. M., Taillard, E. and Agazzi, G. (1999) Ant colonies for vehicle routing problems, In Corne, D., Dorigo, M. and Glover, F. (eds.), **New Ideas in Optimization**. McGraw-Hill.

Gendreau, M., Hertz, A. and Laporte, G. (1994) A tabu search heuristic for the vehicle routing problem, **Management Science, Vol. 40**, 1276-1290.

Jaszkiewicz, A. and Kominek, P. (2003) Genetic local search with distance preserving recombination operator for a vehicle routing problem, **European Journal of Operational Research, Vol. 151**, 352-364.

Kytöjoki, J., Nuortio, T., Bräysy, O. and Gendreau, M. (2007) An efficient variable neighborhood search heuristic for very large scale vehicle routing problems, **Computers &**

Operations Research, Vol. 34, 2743-2757.
Li, F., Golden, B. and Wasil, E. (2005) Very large-scale vehicle routing: New test problems, algorithms, and results, **Computers & Operations Research, Vol. 32**, 1165-1179.
Maniezzo, V. (1999) Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem, **INFORMS Journal on Computing, Vol. 11**, 358-369.
Mester, D. and Bräysy, O. (2007) Active-guided evolution strategies for large-scale capacitated vehicle routing problem, **Computers & Operations Research, Vol. 34**, 2964-2975.
Osman, I. H. (1993) Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem, **Annals of Operations Research, Vol. 41**, 421-451.
Prins, C. (2004) A simple and effective evolutionary algorithm for the vehicle routing problem, **Computers & Operations Research, Vol. 31**, 1985-2002.
Rego, C. and Roucairol, C. (1996) A parallel tabu search algorithm using ejection chains for the vehicle routing problem, In Osman, I. H. and Kelly, J. P. (eds.), **Meta-heuristic: Theory and Applications**. Kluwer, Boston, MA, 667-675.
Rego, C., (1998) A subpath ejection method for the vehicle routing problem, **Management Science, Vol. 44**, 1447-1459.
Reimann, M., Doerner, K. and Hartl, R. F. (2004) D-Ants: Savings based ant divide and conquer the vehicle routing problem, **Computers & Operations Research, Vol. 31**, 563-591.
Robuste, F., Daganzo, C. F. and Souleyrette, R. (1990) Implementing vehicle routing models, **Transportation Research Part B, Vol. 24**, 263-286.
Rochat, Y. and Taillard, E. (1995) Probabilistic and intensification in local search for vehicle routing, **Journal of Heuristics, Vol. 1**, 147-167.
Stützle, T. and Hoos, H. H. (2000) MAX-MIN ant system, **Future Generation Computer System, Vol. 16**, 889-914.
Taillard, E. (1993) Parallel iterative search methods for vehicle routing problems, **Networks, Vol. 23**, 661-673.
Tarantilis, C. D. (2005) Solving the vehicle routing problem with adaptive memory programming methodology, **Computers & Operations Research, Vol. 32**, 2309-2327.
Toth, P. and Vigo, D. (2003) The granular tabu search and its application to the vehicle-routing problem, **INFORMS Journal on Computing, Vol. 15**, 333-346.
Wang, C.-H. and Lu, J.-Z. (2009) A hybrid genetic algorithm that optimizes capacitated vehicle routing problem, **Expert Systems with Applications, Vol. 36**, 2921-2936.
Xu, J. and Kelly, J. P. (1996) A network flow-based tabu search heuristic for the vehicle routing problem, **Transportation Science, Vol. 30**, 379-393.